Art and Artificial Intelligence: generating realistic paintings with GANs.

Bocconi University

Submitted in fulfillment of the requirements for the MSc in

Data Science and Business Analytics

Advisor: Carlo Lucibello

Author: Carlotta Bonanni

April 2022

Dedica e ringraziamenti

I would like to thank Professor Carlo Lucibello. Without his support, his patience and availability, I would have never been able to concretely shape the many ideas that are at the basis of this work.

I want to dedicate this thesis to all the people who have constantly encouraged me to think freely, to never stop being curious and to believe in myself when I didn't have the strength to do it on my own.

Contents

Introduction

In October 2018, an unexpected event occurred at the prestigious auction house Christie's. A work destined to cause much discussion was sold for forty times its originally estimated value: the "Portrait of Edmond Belamy" by the french collective Obvious. It portrays a gentleman dressed in black, with an elegant white collar and a slightly melancholic or pensive air. On the bottom of the painting, a formula is slightly visible: the value function optimized by the algorithm that generated the painting itself. This event was one of the first manifestations of the introduction of artificial intelligence AI in the art world and market. As of today, many advancements have been made in the field, resulting in a wide territory rich in opportunities.

This is the background in which the current investigation takes place. Indeed, with the aim of exploring the potential of the intriguing possibilities found at the intersection of AI and art, this research attempts to produce realistic paintings with generative adversarial networks.

In order to do so, the work will be organized as follows. In the first section of the thesis, a review of the related literature about art and artificial intelligence is presented, allowing the reader to better understand the wide context of AI art.

The second chapter of this research is instead dedicated to a deep and detailed analysis of the central technology that works behind most of the artistic works produced with artificial intelligence: generative adversarial networks (GANs). A special attention is given to this part being GANs the

main instruments employed in the various later experiments performed.

The following section provides notions and explanations regarding the other tools used in the experimental part of this research. Some specific implementations of GANs - namely StyleGAN2-ADA recently developed by NVIDIA [43] and pix2pixHD [84] - are studied after thoroughly discussing a common framework for training these model that is transfer learning. Massive computational and time resources are required when performing computer vision tasks, and image generation is no exception. Using a transfer learning approach could potentially lead to both increased performances and a reduction in the resources needed to efficiently complete the task. Finally, an overview about Holistically Nested Edge detection is given as a method to obtain edge maps of input images.

The fourth chapter of the thesis focuses on the development of an experiment aimed at identifying the best method to generate realistic paintings in the GAN setting. The proposed models are evaluated in quantitative terms and in qualitative ones with the aid of a survey. The latter is built with the purpose of understanding whether people can tell that the paintings were made by machines and in what measure people like the generated paintings.

Finally, the section regarding the conclusions of the work summarizes the contents and the key findings that emerge within this study, before providing some considerations of possible research developments around the topic of AI art.

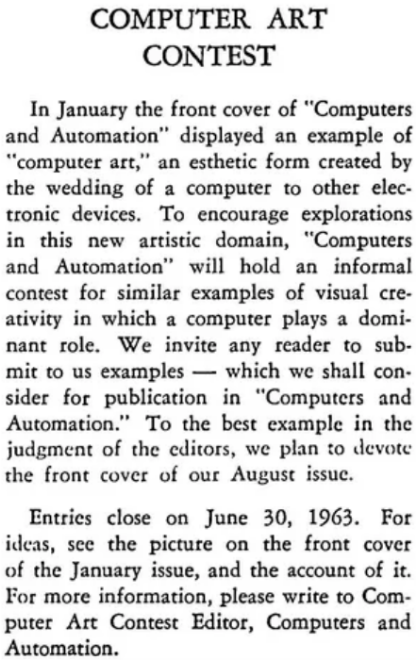
Chapter 1

Related work on art and artificial intelligence

The following chapter aims at illustrating advances and developments regarding the role of artificial intelligence in the art world. Since the current research concerns the generation of paintings, the following sections, and generally throughout the whole paper, the focus regarding the art domain will be limited on visual arts. Nevertheless it should be mentioned that research's interest is steadily growing also in different forms of art, as performing arts.

Thanks to the incredible technological advancement of recent years in the field of machine learning, the applications of the latter are constantly growing and ranging towards new horizons. The use of artificial intelligence in the art world, although existing for quite some time, is one of the most exemplary cases of how these intersections are gaining ground and improving with remarkable speed.

The idea of using computers, albeit for different purposes, in artistic domains is not as recent as one might think, though. One of the first times in which the expression "*computer art*" was used happened in 1963. Indeed, the magazine Computers and Automation held a contest with computer-generated arts, featuring them on the magazine's cover [83] as shown in



COMPUTER ART
CONTEST

In January the front cover of "Computers and Automation" displayed an example of "computer art," an esthetic form created by the wedding of a computer to other electronic devices. To encourage explorations in this new artistic domain, "Computers and Automation" will hold an informal contest for similar examples of visual creativity in which a computer plays a dominant role. We invite any reader to submit to us examples — which we shall consider for publication in "Computers and Automation." To the best example in the judgment of the editors, we plan to devote the front cover of our August issue.

Entries close on June 30, 1963. For ideas, see the picture on the front cover of the January issue, and the account of it. For more information, please write to Computer Art Contest Editor, Computers and Automation.

Figure 1.1: The announcement of the first "Computer Art Contest" in the February 1963 issue of Computers and Automation

figure 1.1. The first prize was won by the United States Army Ballistic Research Laboratories (BRL) with the The Splatter Pattern illustrated in figure 1.2. Although these early works turn out to be very distant, even conceptually, from what current and more recent research is concerned with, they show a clear sign of the direction that was being taken: to expand the purpose of the computer to domains far more extensive than those for which it was born. During the following years many advancements have been made in the field, partly thanks to the continuously expanding access to computers. These developments in computer art will not be covered as both the technology and the techniques used do not represent well contemporary methods. Nonetheless it is important to keep in mind that the intersection between art and machines has been existing for quite a long time. The recent acceleration of interest in artificial intelligence led to redefining the

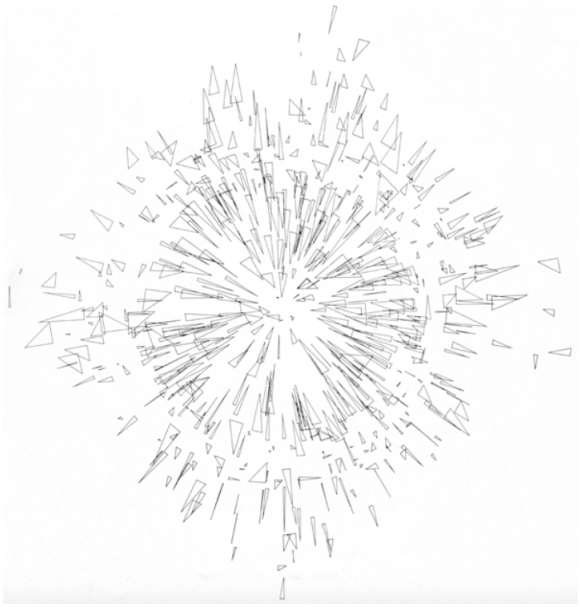


Figure 1.2: Splatter Diagram, 1963, United States Government, Ballistic Research Laboratories, Aberdeen, Maryland, computer-generated.

possibilities of computers and algorithms in artistic domains. Following the breakdown proposed by Cetinic *et al.* in their recent review paper [12], the role of AI in art can be divided into two major macrocategories: the use of AI to process and analyze preexisting art and the use of AI as a tool to produce art. The second macrocategory will be the main object of this section as the current research focuses on it, but a brief overview on the first will be outlined to better appreciate the potential of using AI in the art domain.

1.1 AI for processsing and analyzing existing art

The use of artificial intelligence to understand existing artworks can take a variety of forms. The most common use is probably to be found in automated classification purposes. Indeed, as Cetinic *et al.* report, these task

represent "one of the central challenges of computational art analysis over the last decade"[12]. The earlier researches on classification of artworks based on artist's name [47], style [76] or genre [2], focused on using machine learning algorithms on handmade image features previously extracted. The true revolution for art classification happened thanks to the introduction of Convolutional Neural Networks (CNNs). In 2013 Karayev *et al.* proposed a novel way of classifying artworks by utilizing pretrained CNN layers as features for style classification [75]. Later on, Girshick *et al.* expanded this obtaining higher performances by fine tuning the pretrained network on the target dataset [28]. These techniques, along with many others such as the ones presented in [81], [72], [92] confirmed that transfer learning, meaning the use of knowledge gained from a certain task onto a different one, can lead to dramatic improvement in model's performances. For a more complete outline of contemporary studies about artwork classification along with a novel method for classifying paintings through iconographic elements, the reader can refer to [55].

Another frequent application of AI, and in particular of neural nets, is to understand and explore the content of images by performing object recognition tasks. In the artistic domain, this translates into studies that focus on retrieving paintings with certain objects depicted in them [17], find out the position of certain objects in the artworks [31], along with identifying content with the aim of to uncovering recurring patterns in collections [48]. The practical application of this exercise is to be found in retrieval systems which enable the organization and analysis of massive collections of paintings in a systematic and efficient way [50]. Aside from inspecting just visual similarities, a recent new wave of studies is focusing on exploring both visual and textual information of artworks. The general purpose of these kind of analysis is to build multimodal retrieval systems. About this matter, Cetinic *et al.*[12], report as some of the most used dataset for the mentioned purpose SemArt, presented by Garcia and Vogiatzis [26], and Artpedia, introduced

by Stefanini *et al.*[78]. Both the first and the latter proposed methods that generally aim at mapping images and their description in a joint embedding space. The most recent and powerful model built in this domain is arguably Contrastive Language-Image Pre-Training (CLIP), developed by Radford *et al.* at OpenAI [65]. It is a neural network that has been trained on text-image pairs and that, as reported in the original paper, "can be instructed in natural language to perform a great variety of classification benchmarks". Practical applications of it include predicting the most relevant text snippet, given an image. As explained by Gonsalves [29]:

> "The goal of the training is to have the encoded images match the encoded words. Once trained, the image encoder system converts images to embeddings, lists of 512 floating-point numbers that capture the images' general features. The text encoder converts a text phrase to similar embedding that can be compared to image embeddings for a semantic search."

Another major contribution of deep learning to the art world is the possibility of implementing a quantitative method for analysing theoretical notions pertinent to art history. Perhaps, the most compelling effect of employing a quantitative perspective in the analysis of vast art datasets, is the opportunity to identify high-level features that correspond to abstract concepts of art comprehension. An interesting example is the work carried out by Deng *et al.* [18], where they proposed the concept of "representativity" to assess artworks. It designate the degree to which an artwork is deemed typical within the realm of an artist's œuvre, employing deep neural networks to extract style-enhanced image representations.

Finally, one of the most interesting topics in the area of computational analysis of paintings relates to perception. Even if psychologists have been studying aspects of visual perception for many years, it was not up until recently that this subject has became of great interest in the deep learning and computer vision domain. As reported in [12], computational aesthetics is

a developing research area focused on proposing computational approaches that can predict aesthetic judgments or emotional cues, in a human-like fashion. An example of this kind of research can be appreciated in Zhao *et al.*[93]. Indeed, they propose an end-to-end structure for representation learning of image composition and their possible use in aesthetic prediction. As for sentiment analysis purposes, a great step forward has happened thanks to Mohammad and Kiritchenko [58] with the introduction of WikiArt Emotions, a dataset of paintings annotated for various emotional cues provoked in the viewer. Similarly, Achlioptas *et al.*[1], presented ArtEmis, a large-scale dataset focused on "the affective experience triggered by visual artworks". The crucial innovation of this dataset is that they asked the annotators to indicate not only the prevailing emotion they felt for a certain painting/artistic photograph, but to also provide an explanation for their emotion choice. Eventually, they developed trained some captioning systems able to express and explain emotions from images. Of course, this area, as well as the entire intersection of art and artificial intelligence, is often criticized by experts in both fields, as these are domains where the "human vs AI" struggle becomes prominent, and more often won by humans. Indeed, the current techniques are limited, but they surely express a great potential for future research.

1.2 Generating Art

Art has always been a fundamental pillar of human societies for various reasons. In its most immediate meaning, art is the aesthetic expression of the human interiority. By reflecting the opinions, feelings and thoughts of the artist in the social, moral, cultural, ethical or religious context of his historical period, it bears many implications on a historical and sociological level. With that in mind, it is not surprising that interest in the intersection of art and computers has been growing in recent years. Artificial intelligence-based

technologies are now part of everyday life, and the attempt to investigate their potential for the pursuit of artistic expression becomes, under these lenses, just a natural consequence of the societal evolution. For a deeper comprehension of the power that these technologies hold with regard to art production, the following section aims at presenting the path outlined by the research of recent years towards the realization of different forms of art through artificial intelligence.

The major challenge in computer art, lies in the need for extensive human intervention, both in terms of programming and production of the work. Attempts to optimize this aspect can be identified in the use of genetic algorithms. The latter, generate samples in a iterative fashion, evaluating them trough a fitness function. Additional adjustments are then made to enhance performances in the next iteration. This approach bears some issues as it is difficult to define a suitable fitness function "that makes aesthetic sense" [19]. Therefore, these techniques do not seem to be appropriate for the task of autonomous art generation. The following image by Cetinic *et al.*[12], defines a clear outline of technological milestones in the AI art world.

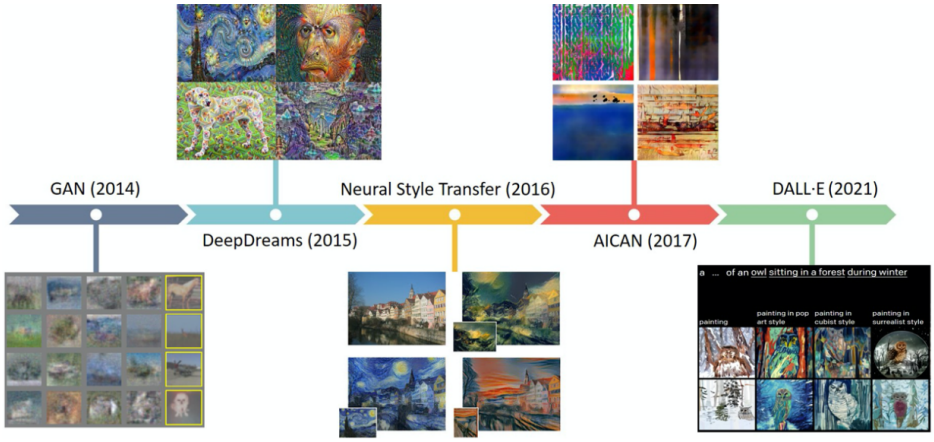


Figure 1.3: Timeline of the most important technological milestones that led to the current AI Art production. Figure from [12]

The first huge breakthrough in the domain of AI art happened in 2014

with the invention of Generative Adversarial Networks (GAN) by Ian Goodfellow [34]. These models, allowed the generation of realistic images through a nearly automatic procedure. Compared to previous image generation methods, these networks require little human intervention, leaving room for the machine to act almost autonomously. As an extensive explanation of these models is presented in chapter 2, they will not be treated in the current section. For the remaining part of the chapter, the following definition given by Goodfellow in [33] should be enough to get the idea of the basic structure of these models.

> "Generative adversarial networks are based on a game theoretic scenario in which the generator network must compete against an adversary. The generator network directly produces samples. Its adversary, the discriminator network, attempts to distinguish between samples drawn from the training data and samples drawn from the generator."

Especially in their first years, GANs, and deep learning models in general, were not directly used for art generation. As an example, take DeepDreams. It was firstly introduced by Mordvintsev *et al.* in 2015 [59] with the scope of visualizing and better understanding what each layer of a neural network has learned. The image produced presented a quite psychedelic and hallucinatory effect, that drew curiosity within the art world. Subsequently, it became a tool for creating digital art.

Another turning point that set off the use of AI for artistic purposes was Neural Style Transfer (NST). This approach has been presented in the seminal work of Gatys *et al.*[27] that proved the power of convolutional neural networks (CNNs) in generating artistic images by splitting and recombining image content and style. After this innovation, many new research application followed. Generally they involved the use of GANs architectures to perform NST. For instance, Lie *et al.* proposed a conditional network that generates painted version of given sketches. The cGAN was trained on a

dataset composed by pairs of sketches and true colored images of minions and anime. In this framework, they utilized a U-net generator that helps in avoiding information loss that could happen when employing a straight encoder-decoder network [52]. The U-net architecture for the generator was firstly proposed by Isola *et al.* in 2017 [39] for the pix2pix model. Basically it means that between encoding and corresponding decoding layer, skip-connections are added, forming a U-shape. The discriminator architecture instead is composed by an encoder units needed to discriminate between the 'real' input sketch-image pair and the 'fake' ones. The evaluation method for the results was mainly qualitative, with 55 volunteers that were asked to choose their most and least liked samples from four different models. The presented model outperformed the three preexisting ones.

More recently, Liu *et al.* presented an interesting approach for generating fully detailed art-stylized images also from sketches [51]. The GAN based original solution allows the user to pick a style for the generated image by specifying the artist or the artistic style. For evaluation of the results, they used both quantitative metrics such as the Fréchet Inception Distance (FID), and qualitative ones by asking 100 participants to choose between the proposed model's generated images and baseline models' ones. The proposed model outperfomed existing ones according to both types of evaluation.

A couple of years after the birth of GANs, Elgammal *et al.*[20] presented an implementation on GANs that was aimed at making the whole generative process more creative. The model proposed indeed, does not only learn to produce images from current art, but by learning styles it deviates from them, to produce novel artworks. This was attained by modifying the objective function so that the generation problem tries not only to minimize the deviation from the true artworks distribution but also to maximize the deviation from the style norms it has learned. Afterwards, they showed the images to human subjects who were unable to recognize AICAN - the name they gave the model from Artificial Intelligence Creative Adversarial Networks -

generated images from real paintings by contemporary artists.

Figure 1.4: Images generated by CAN that ranked high in "likeness" according to human subjects. Image from [20]

The final step further in AI art generation, has been taken thanks to the rapid development of multimodal learning. The first seminal work on text-to-image synthesis has been carried out by Reed *et al.* [69] in 2016. Research has come a long way since their first attempt, which resulted in a conditional GAN able to generate images based on whole sentence embeddings obtained from a pre-trained text encoder. A natural evolution in this context was expanding the conditioning set, by adding extra information. Recent examples of what these extra information are include multiple captions [15], dialogue [23] or layout - intended as bounding boxes and class label annotations - [36], [79]. For a more comprehensive review of text-to-image generation methods, the reader can refer to [22].

The most notable innovation in this field perhaps, has been introduced by the OpenAI team at the beginning of 2021 with DALL·E [67]. As the authors discuss, it is a model "based on a transformer that autoregressively models the text and image tokens as a single stream of data". Since its publication, many people have tried to replicate DALL·E's approach to image genera-

tion, with a particular interest specifically in art generation. A fascinating example has been presented by Rashad [68]. He implemented a combination of existing models, specifically the aforementioned CLIP technology [64] and the VQGAN - by Esser *et al.* [21]- obtaining some quite astonishing results, very similar to what DALL·E delivers. The resulted framework is available to use at the website Text2Art. The image showed in figure 1.5, which has been generated on Text2Art, gives an idea of the power of these models and the artistic opportunities associated with their development. As an example of possible and realistic applications, just recently the collective Bored.Ai is using this technology to let members self-generate their personal NFTs.

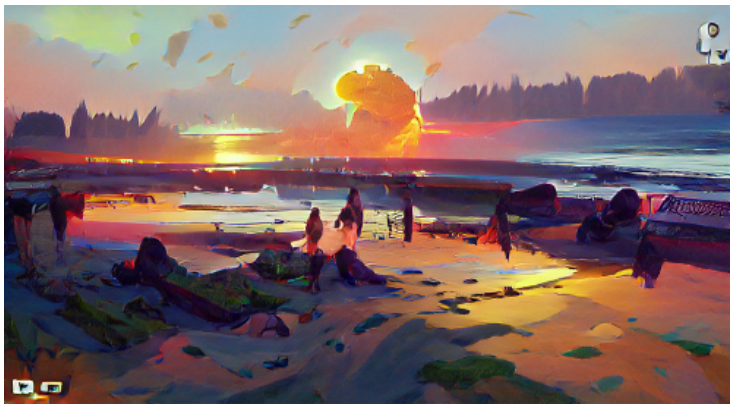


Figure 1.5: Image generated on the website Text2Art from the input sentence "Painting of beach sunset #artstation"

This chapter has shown many of the possible use of AI technologies in artistic domains, and how research is moving in this context. A problem that should be evident at this point, is the imbalance between the vast opportunities that the use of artificial intelligence offers in the artistic field, and the research interest, albeit growing, in this regard. In fact, many of the methods and models presented, rarely originated for pure artistic purposes, and it is more incidentally that they are being used for this.

Starting from this scenario, the present investigation is indeed aimed at performing a more consistent and targeted analysis of the use of artificial

intelligence to generate art. In order to do so, three different techniques will be implemented with the goal of producing realistic paintings of impressionist landscapes. The results obtained will then be evaluated using both a quantitative approach and a turing test. The following chapters will discuss in depth the models and techniques used to implement the three methods to generate the paintings.

Chapter 2

Fundamentals of Generative Adversarial Networks

Generative Adversarial Networks have been proposed in 2014 by Goodfellow *et al.* [34]. They are a framework in which the estimation process happens by training two adversarial networks: a generator G, and a discriminator D against each other. As reported in Creswell *et al.* [16] it useful to think about the two networks as an art forger and an art expert. The forger, G, creates the counterfeit and does it in a way in which it resembles the original images as much as possible. The aim of the expert, D, is to recognise whether the image is a forgery or not. It is important to mention that the expert is the only one in possession of the original images. Thus, the only way the forger can improve its performance is through the expert feedback, which simply consists in discerning whether the image is real or forged. Just as in this forger/expert parallel, the two models, G and D are trained simultaneously against each other. The competition between the two models leads them to improve their technique until the counterfeits are almost indistinguishable from the original images.

Among some of the most typical applications of GANs one can find up-scaling (improving the resolution of images), removing objects from images,

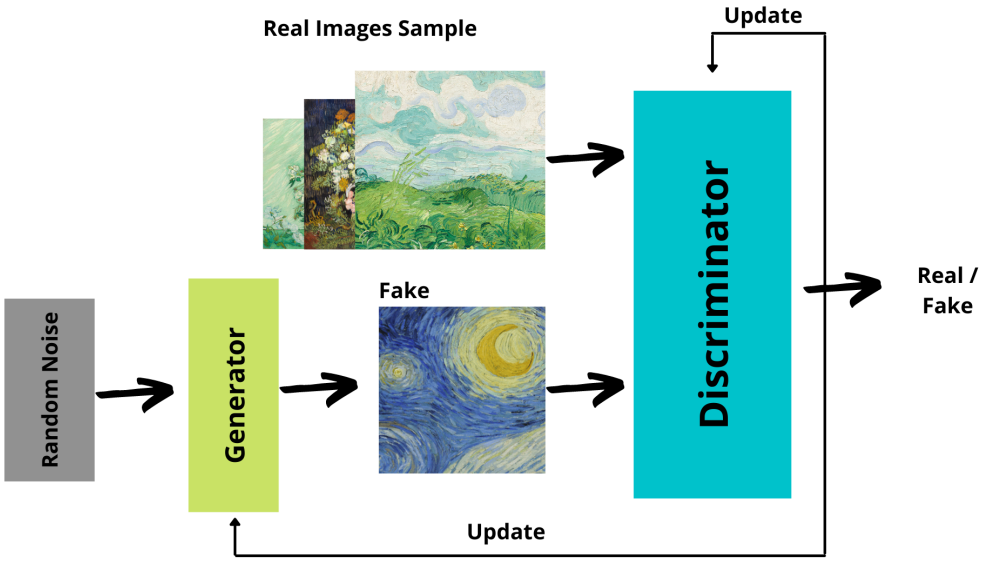


Figure 2.1: Overview of a GAN basic structure

converting audio/text to images, generating novel datasets for training (as Goodfellow proposed in his original paper [34]), semantic image to photo translation [84], 3D object generation [24] and many more. When combined with other technologies such as convolutional neural networks (CNNs), they have been widely used to capture and reproduce a certain style. In their breakthrough 2016 paper , Gatys *et al.* [27] have proposed a way of separating content and style of an image. The idea has been further developed by training GANs to learn from a complete set of pictures instead of a single image as originally proposed.

More recent - and in some measure preoccupying - evolutions of GANs implementation include deepfakes. Indeed, GANs are used to modify the original content of a source image/video by overlapping it with some novel content. This in principle is a harmless and somehow fun activity, but is easy to imagine the potentially dangerous usage of the technology.

In the next sections some major issues of this architecture and other commonly implemented structures of GANs will be discussed, starting from

its original framework as it was presented in Goodfellow *et al.*,2014 [34].

2.1 Vanilla GAN structure, Goodfellow *et al.* (2014)

In the original paper, the two models constituting the generator and the discriminator were implemented both as multilayer perceptrons [34]. Most of the successive developments of GANs focused on changing this architecture, by using different kinds of networks both for the generator and for the discriminator.

In the basic version of the model, the two networks work as follows: given a latent space $\mathbf{z}$ having an a priori distribution $\mathbf{p}_z(\mathbf{z})$, the generator represents a differentiable function $\mathbf{G}(\mathbf{z},\theta_g)$ that outputs the new data according to a certain distribution $\mathbf{p}_g$, where θ_g are the parameters of the generative model. The discriminator represents a differentiable function $\mathbf{D}(\mathbf{x},\theta_d)$, where θ_d are the parameters of the discriminative model, which outputs the probability that $\boldsymbol{x}$ comes from the distribution of the training data $\mathbf{p}_{data}$. The aim is to obtain a generator that is a good estimator of $\mathbf{p}_{data}$. When this happens, the discriminator is "tricked" and can no longer distinguish samples coming from $\mathbf{p}_{data}$ and samples coming from $\mathbf{p}_g$. The key to obtain this is adversarial training of the models. While D is trained to maximize the probability of giving the correct label to both training examples and fake generated samples from G, the latter is simultaneously trained to minimize log(1-D(G(z))). As explained in the original paper, the generator and the discriminator nets play a two-player minimax game with *V(G,D)* as value function of the form:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \tag{2.1}$$

Since the generator and the discriminator are given enough capacity, the training criterion allows the data generating distribution to be recovered. The

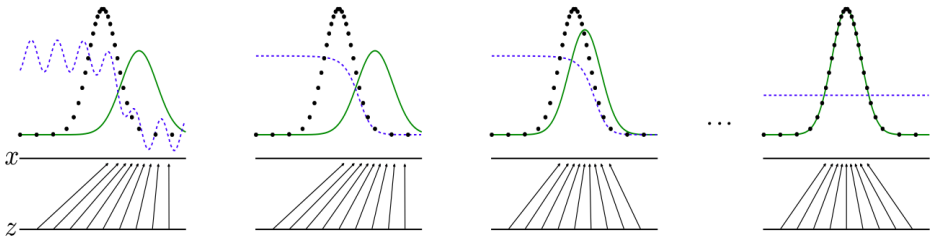


Figure 2.2: Overview of a GAN training - Goodfellow *et al.* (2014) [34]

procedure is implemented with a numerical and iterative approach explained in a less formal way in figure 2.2. The discriminative distribution (D) is represented by the blue dashed line. The data generating distribution is the black dotted line $\mathbf{p}_x$ and the generative distribution $\mathbf{p}_g$ (G) is the green solid line. The network is trained by updating at the same time D so that it is able to discriminate between samples from $\mathbf{p}_x$ and $\mathbf{p}_g$. The lower line z and x are respectively the domain from which z is sampled (here uniformly) and part of the domain of x. The arrows represent the mapping $\mathbf{x}$=G($\boldsymbol{z}$) that gives the non-uniform distribution $\mathbf{p}_g$ on the transformed samples. The figure, presented in the orginal paper, is described as follows:

> "G contracts in regions of high density and expands in regions of low density of $\mathbf{p}_g$. (first image). Consider an adversarial pair near convergence: $\mathbf{p}_g$ is similar to $\mathbf{p}_{data}$ and D is a partially accurate classifier. (second image) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to
>
> $$D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$
>
> (third image) After an update to G, gradient of D has guided G($\boldsymbol{z}$) to flow to regions that are more likely to be classified as data. (fourth image) After several steps of training, if G and D have enough capacity, they will reach a point at which both

cannot improve because $\mathbf{p}_g = \mathbf{p}_{data}$. The discriminator is unable to differentiate between 1 the two distributions, i.e. $\mathrm{D}(\boldsymbol{x}) = 2$."

2.1.1 Optimal solution and convergence of Vanilla GANs

The results presented by Goodfellow *et al.* (2014) are obtained in a non-parametric setting. They indeed "represent a model with infinite-capacity by studying convergence in the space of probability density functions" [34]. The basic algorithm proposed to optimize eq. (2.1) is illustrated in figure 2.3.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure 2.3: Algorithm proposed in the original GAN paper. [34]

To prove that Algorithm 1 from figure 2.3 optimizes Equation (2.1), Goodfellow *et al.* proceeded with the following approach.

Firstly, they considered the optimal discriminator D for any given generator G.

Proposition 1.(Goodfellow *et al.* 2014 [34], Proposition 1). *For G fixed, the*

optimal discriminator D is

$$D_G^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2.2}$$

Proof. [34] The aim of the discriminator D during training, for any generator G, is to maximize the quantity *V(G,D)*

$$\begin{aligned} V(G,D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_{\boldsymbol{z}} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned} \tag{2.3}$$

The function $y \to a\log(y) + b\log(1-y)$ reaches its maximum in $[0,1]$ at $\frac{a}{a+b}$ for any $(a,b) \in \mathbb{R}^2 \backslash \{0,0\}$ and the discriminator does not have to be defined outside of $\text{Supp}(p_{\text{data}}) \cup \text{Supp}(p_g)$. This proves Equation(2.2). □

The minimax game in Eq. (2.1) can now be reformulated as:

$$\begin{aligned} C(G) &= \max_D V(G,D) \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} [\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{x}}} [\log(1 - D_G^*(G(\boldsymbol{z})))] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} [\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g} [\log(1 - D_G^*(\boldsymbol{x}))] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g} \left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] \end{aligned} \tag{2.4}$$

Theorem 1.(Goodfellow *et al.* 2014 [34], Theorem 1). *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data.}$ At that point, $C(G)$ achieves the value $-\log 4$.*

Proof. [34] Inserting in Eq.(2.3), D^* when is at its optimum as in Eq.(2.2):

$$
\begin{aligned}
C(G) &= \int_x \left(p_{data}(\boldsymbol{x}) \log\left(D^*(\boldsymbol{x})\right) + p_g(\boldsymbol{x}) \log\left(1 - D^*(\boldsymbol{x})\right)\right) d\boldsymbol{x} \\
&\overset{(2.2)}{=} \int_x \left[p_{data}(x) \log\left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}\right) + p_g(\boldsymbol{x}) \log\left(\frac{p_g(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right)\right] d\boldsymbol{x} \\
&= \int_x \left[p_{data}(\boldsymbol{x}) \log\left(\frac{p_{data}(\boldsymbol{x})}{\frac{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}{2}}\right) + p_g(\boldsymbol{x}) \log\left(\frac{p_g(\boldsymbol{x})}{\frac{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}{2}}\right)\right] d\boldsymbol{x} \\
&= \int_x \left[p_{data}(\boldsymbol{x}) \log\left(\frac{p_{data}(\boldsymbol{x})}{\frac{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}{2}}\right) + p_g(\boldsymbol{x}) \log\left(\frac{p_g(\boldsymbol{x})}{\frac{p_{data} + p_g(\boldsymbol{x})}{2}}\right)\right] d\boldsymbol{x} \\
&\quad + \log\left(\frac{1}{2}\right) + \log\left(\frac{1}{2}\right) \\
&= \int_x \left[p_{data}(\boldsymbol{x}) \log\left(\frac{p_{data}(\boldsymbol{x})}{\frac{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}{2}}\right) + p_g(\boldsymbol{x}) \log\left(\frac{p_g(\boldsymbol{x})}{\frac{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}{2}}\right)\right] d\boldsymbol{x} - \log(4) \\
&\overset{(a)}{=} \mathrm{KL}\left(p_{data}(\boldsymbol{x}) \| \frac{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}{2}\right) \\
&\quad + \mathrm{KL}\left(p_g(\boldsymbol{x}) \| \frac{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}{2}\right) - \log(4) \qquad (2.5)
\end{aligned}
$$

where (a) is because of the definition of Kullback-Leibler (KL) divergence. The Jensen-Shannon Divergence (JSD) can be recognized in this last step. Indeed, it is defined as (Nielsen, 2010 [60]):

$$
\mathrm{JSD}(P \| Q) := \frac{1}{2}\mathrm{KL}\left(P \| \frac{1}{2}(P + Q)\right) + \frac{1}{2}\mathrm{KL}\left(Q \| \frac{1}{2}(P + Q)\right),
$$

with P and Q being probability densities. The difference with KL divergence is that JSD is symmetric. The last C(G) obtained can be therefore reformulated as:

$$
C(G) = 2\,\mathrm{JSD}\left(p_{\mathrm{data}}(x) \| p_g(x)\right) - \log(4)
$$

Since, according to Eq.(2.1), the generator minimizes $C(G)$ and by definition

the JSD is non-negative, the above loss function is minimized if:

$$\mathrm{JSD}\left(p_{\mathrm{data}}\left(\boldsymbol{x}\right)\|p_{g^*}(\boldsymbol{x})\right) = 0 \Longrightarrow p_{\mathrm{data}}\left(\boldsymbol{x}\right) = p_{g^*}(\boldsymbol{x})$$

□

To prove the convergence of GAN they continue with:

Proposition 2.(Goodfellow *et al.* 2014 [34], Proposition 2). *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}\left[\log D_G^*(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log\left(1 - D_G^*(\boldsymbol{x})\right)\right]$$

then p_g converges to p_{data} .

Proof. The KL divergences in Eq. (2.5) are convex functions w.r.t. p_g. When the updates of $p_g(x)$ are sufficiently small, it converges to $p_{data}(x)$. Eq. (2.5), holds if Eq. (2.2) holds, meaning that the discriminator is allowed to reach its optimum value.

□

In conclusion, adversarial nets represent through the function $G\left(\boldsymbol{z};\theta_g\right)$ a subest of p_g distributions. During training one optimizes θ_g instead of p_g directly.

2.2 Common problems of GANs

In practice, GANs tend to suffer some problems. Especially in their first formulation, these were the major issues:

1. **Non-convergence**. The model parameters oscillate, are unstable and do not converge.

2. **Mode collapse**. This is one of the major issues in GANs training. It happens when the generator collapses and produces limited varieties of samples.

3. **Vanishing gradient**. If the discriminator gets too good at its jobs, the generator can fail as it is not provided enough information to make progress.

4. **Evaluating GANs**. A still partially unsolved problem with GANs relies in the difficulty of finding a commonly accepted metric to evaluate model's performances.

2.2.1 Non-convergence

To fully understand why the problem of non convergence arises, it is useful to take a deeper look at the training algorithm of GANs. Since a GAN is composed by two distinct networks, its training process must tackle two issues: it must deal with two different kind of training, one for the generator and another one for the discriminator, and it must identify convergence, that often is not reached easily. As G and D have different training methods, it is common practice to alternate the training of the two networks. This means that when the discriminator is trained, the generator is kept constant. The discriminator in its training phase has to learn how to properly distinguish real images from fakes. Likewise, the discriminator is kept constant while the generator is training, or else the latter would be trying to hit a moving target and it would be difficult to converge. This two steps are repeated in alternating periods. Thanks to this back and forth operation GANs are able to address the complications that arise with generative problems.

As G's performances improve, D's get worse. If the first succeeds perfectly the latter reaches a 50% accuracy, meaning it is no longer able to tell apart the true images from the generated ones, and its prediction becomes a like a coin toss. This constitute an issue for convergence of the network as a whole.

The generator receives an increasingly less meaningful feedback over time. If the training continues beyond the point where the discriminator prediction is at random, generator's quality may collapse too as it is training on a poor feedback.

This leads to a definition of convergence in GANs training that is often not stable and more transient.

Researchers have implemented different kind of regularization to ensure or at least improve the probability of convergence. Typical solutions include adding noise to the inputs of the discriminator [3] or penalizing its weights [70].

2.2.2 Mode collapse

GANs often suffer from mode collapse, also known as the Helvetica scenario. In this event the generator fails to generalize correctly. It learns indeed how to generate samples from a restricted subset of the modes of the data distribution. This leads the network to missing entire modes from the underlying distribution, even if they occur throughout the training data. [13, 71]. An intuitive explanation for mode collapse is that every iteration of genera-

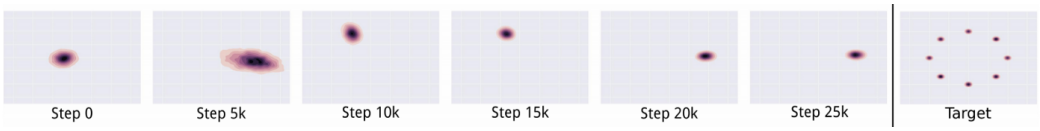


Figure 2.4: Example of mode collapse problem on a 2D toy dataset. Target distribution is a mixture of Gaussians in 2D space and is represented in the last image. During the training, G never converges to a fixed distribution, and only ever assigns significant probability mass to a single data mode at once. (Figure from [54])

tor over-optimizes for a particular discriminator, and the discriminator never manages to learn its way out the cat and mouse game and keep on incorrectly classifying fake samples as real. As a result the generator rotate through a small set of output types that are able to fool the discriminator.

A straightforward solution to this issue has been proposed by Salimans *et al.* (2016) [71] and it consists of minibatch discrimination. The idea is to feed real and generated images into the discriminator in separate batches and calculate the similarity of the images in the same batch. Then, the computed similarity is appended in one of the dense layers of the discriminator. If mode collapse happens, the similarity will increase and the discriminator will more easily be able to recognize fake images. This in turn will penalize the generator if mode is collapsing. Other well-known proposed solutions to this problem will be illustrated later on as they tackle more generally all issues encountered in GANs training.

2.2.3 Vanishing gradients

Another problem related to the aforementioned ones occurs when the discriminator gets too good: the gradient will be vanishingly small, effectively preventing the generator to update its weights and eventually stop learning. For the sake of comprehension, imagine a chess game between a grandmaster and a beginner. The grandmaster will not be challenged enough and the beginner will never be able to learn anything as the grandmaster will always beat him to fast without giving any meaningful feedback. In the original paper [34]this problem was already mentioned. In particular it was noted that it arises from the minimax game as formulated in the first section of this chapter (2.4). The latter derives from the zero-sum game where $\mathrm{J}^{(G)} =$ - $\mathrm{J}^{(D)}$ and $\mathrm{J}^{(G)}$ and $\mathrm{J}^{(D)}$ are respectively the generator's and the discriminator's costs. The paper therefore suggests to turn the game into a heuristic, non-saturating version of it by modifying the cost for the generator. Instead of obtaining it by flipping the sign on the discriminator's cost, they proposed to flip the target used to build the cross-entropy cost. The newly obtained generator cost is:

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}}[\log D(G(\boldsymbol{z})) \tag{2.6}$$

The generator now maximizes the log-probability of the discriminator being wrong. As stated in Goodfellow 2016 [32]:

> "This version of the game is heuristically motivated, rather than being motivated by a theoretical concern. The sole motivation for this version of the game is to ensure that each player has a strong gradient when that player is "losing" the game."

This is not the only suggestion proposed to solve this issue, but rather the first and most straightforward one.

2.2.4 Evaluating GANs

Another infamous issue is connected to the very simple question: how can one evaluate a GAN's performance? Indeed, there is no way to objectively evaluate training progress and the relative or absolute quality of the model from loss alone. The two main roads to follow are quantitative VS qualitative methods of evaluation, but in this framework, both present some issues. For instance, being able to trick a viewer into believing that the generated image is a real one at first glance seems like a good option. In contrast this kind of measure could lead to prefer models that focus on a limited subsample of the data incurring in problems such as overfitting, mode collapse, low diversity. On the other hand, while quantitative measures are surely less subjective, they may not translate directly how humans perceive the quality of generated images. In the context of art generation this becomes very clear. A detailed and extensive review of methods of evaluation has been carried out by Borji 2018 [7], and recently updated in [8]. A summary of common GAN metrics presented in the first paper can be found in table A.1 of the appendix.

In the following section the most common GAN architectures developed after their first formulation [34] will be outlined. Most of them have been designed

with the aim of overcoming, or if not at least diminishing, the aforementioned GAN specific problems.

2.3 Development of GANs architecture

Some natural evolutions of the model and clever solutions to the aforementioned issues can be found in:

- **cGAN** (Mirza *et al.* 2014 [56]). The idea of a conditional GAN was already suggested by Goodfellow in his original paper [34]. These models, indeed, are a variant of a simple GAN in which both the dicriminator and the generator are conditioned during training on auxiliary data such as a class label. As Goodfellow reported in [32]:

  > "Using labels in any way, shape or form almost always results in a dramatic improvement in the subjective quality of the samples generated by the model."

- **DCGAN** (Radford *et al.* 2015 [66]). Another natural extension of GAN framework is proposed with Deep Convolutional GANs (DCGAN). This architecture refers to models which use batch normalization layers in both the generator and the discriminator, replace any pooling layer with strided convolutions (D) and fractional strided convolutions (G), remove fully connected hidden layers, use ReLU activation of all of G's layers except for the output which uses tanh, use LeakyReLU activation for all of D's layers.

- **Unrolled GANs** (Metz *et al.* 2016 [54]). They represent an attempt to solve mode collapse and stability problems. The intuition behind their formulation is the following: whenever someone is playing a game, she anticipates her opponent's next moves and consequently plans her next moves. With this architecture the generator is given the opportunity to do so by unrolling k steps on how the discriminator will behave.

G is then updated through backpropagation with the cost that has been computed in the final k step. Looking ahead hinders the generator to exploit local optima to fool the discriminator and fall in the Helvetica scenario.

- **Wasserstein GAN** (Arjovsky *et al.* 2017 [4]). In this extension, the discriminator is replaced by a critic that scores the realness vs fakeness of the given image. The aim is to reformulate the model's loss functions to make it represent more directly the minimization of the distance between two probability distributions. The Wasserstein loss is defined to overcome the problems of the original GAN that arose from the fact that the loss was defined as a zero-sum (minimax) game. In practice, the critic tries to maximize the difference between its output on real instances and its output on fake instances. The losses derive from the Earth Mover's Distance between real and generates distributions, that in turn leads to an extra advantage of obtaining a true metric compared to cross entropy.

- **AdaGan** (Tolstikhin *et al.* 2017 [80]). This formulation proposes an iterative procedure inspired by boosting algorithms. A new component is added into a mixture model by running the GAN on a reweighted sample. Thanks to boosting, many weak learners are aggregated greedily to become a strong composite one. This procedure helps in leading the model to converge and solves the problem of mode collapse.

- **Progressive GANs** (Karras*et al.* 2017 [42]).Progressive Growing of GANs are an extension of GAN models that allow generation of high resolution images, speeding up and making the training process more stable. In practice, the size of the model is incrementally increased during training. Initially the model is trained to produce very low resolution images. When it converges, new layers are added and the output resolution doubles. The process goes on until the target resolu-

tion is obtained.

- **Transformers GAN** (recent examples can be found in Jiang *et al.* 2021 [40], or in Hudson *et al.* 2021 [38]). State-of-the-art GANs have included in various ways the promising Transformer architecture [82] leading to excellent results.

Please beware that this list is by no means exhaustive as it would be out of scope for the current project to investigate the massive variety of GAN architectures any further, but it is useful to have an idea of both the first and the more recent directions taken by researchers to overcome GANs related issues. Curious readers may refer to [87] for a broader and more cohesive taxonomy of GANs, of which a summarized outline is illustrated in the figure below.

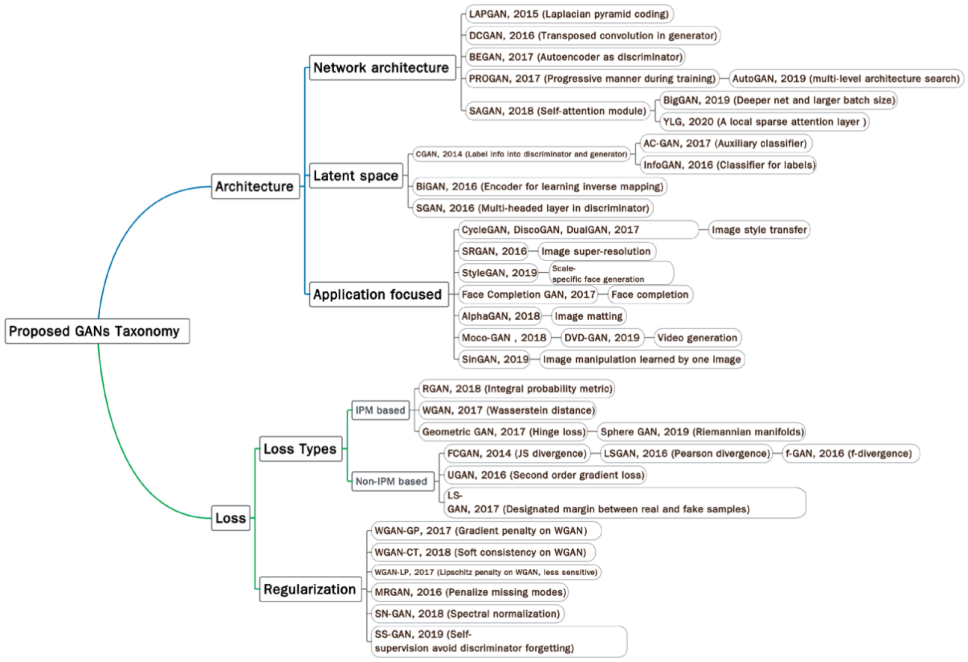


Figure 2.5: A recent GAN taxonomy. Figure from [87]

Chapter 3

Methods and tools for art generation

The following chapter aims at providing a comprehensive insight on the models and techniques that have been employed in the experimental part of the current research. In particular, attention will be focused firstly on transfer learning, secondly on StyleGAN2 model, later on Holistically-Nested Edge Detection (HED) and finally on the pix2pixHD model.

3.1 Transfer Learning

The first studies about transfer learning date back to 1976, when scientists Stevo Bozinovski and Ante Fulgosi published a first paper [10][9], in which they explicitly discussed transfer learning in the process of training neural networks. Within the article a first mathematical and geometrical model of transfer learning was provided.

The term transfer learning refers to an advanced method of machine learning in which a model that has been pre-trained to perform a specific task is reused as a starting point for the development of a model intended to perform a second different task. This practice is inspired by the fact that people can

intelligently apply previously learned knowledge to solve new problems more quickly or with improved solutions.

In the deep learning context indeed, it is common practice to use pre-trained models as a departure point for new models. This technique is used notably for computer vision and natural language processing tasks for one main reason: the development of neural network models, especially for such areas, requires the deployment of vast computational resources and time; this problem is solved through transfer learning, which allows the reuse of already set models.

Typically, this system is used within a context of supervised learning, in which the input is the same but the objective to be achieved varies. Transfer learning, therefore, can be very useful in the case in which there is a similarity in the structure of the new problem to be solved.
The goal, in simple words, is to exploit the knowledge gained from the first task to extract information that may be useful during the learning phase of the second task or to make new predictions in a second setup of the model.

The following section proposes to give a formal overview of transfer learning techniques. For a more comprehensive and recent review on transfer learning the reader can refer to [94].

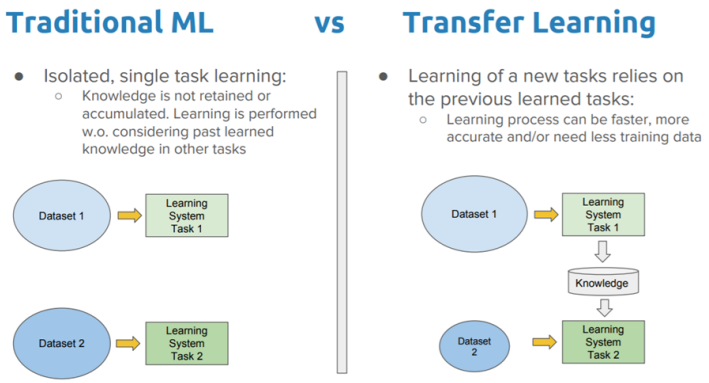


Figure 3.1: Traditional machine learning VS Transfer Learning. Figure from [73]

3.1.1 Definition and categorization of transfer learning techniques

To obtain a more formal definition, it is necessary to define firstly what a domain and what a task are. Following Weiss *et al.* [88], a domain $\mathcal{D}$ can be defined by two components: a feature space $\mathcal{X}$ and a marginal probability distribution $\mathrm{P(X)}$, where $\mathrm{X} = \{\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}\} \in \mathcal{X}$. Generally, two different domains, could have different feature spaces or different marginal probability distributions. For a given domain $\mathcal{D}$, a task $\mathcal{T}$ consists of two parts, a label space $\mathcal{Y}$, and a predictive function $f(\cdot)$, which is learned from the feature vector and label pairs $\{x_i, y_i\}$ where $x_i \in X$ and $y_i \in \mathcal{Y}$. Therefore, by the above definitions, a domain $\mathcal{D} = \{\mathcal{X}, \mathrm{P(X)}\}$ and a task $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$. The source domain data $\mathrm{D_S}$, is defined $\mathrm{D_S} = \{(\mathrm{x_{S1}}, \mathrm{y_{S1}}) \ldots, (\mathrm{x_{Sn}}, \mathrm{y_{Sn}})\}$, where $\mathrm{x_{Si}} \in \mathcal{X}_S$ is the ith data instance of $\mathrm{D_S}$ and $\mathrm{y_{Si}} \in \mathcal{Y}_S$ is the corresponding class label for $\mathrm{x_{Si}}$. Similarly, $\mathrm{D_T}$ is the target domain data defined as $\mathrm{D_T} = \{(\mathrm{x_{T1}}, \mathrm{y_{T1}}) \ldots, (\mathrm{x_{Tn}}, \mathrm{y_{Tn}})\}$, where $\mathrm{x_{Ti}} \in \mathcal{X}_\mathcal{T}$ is the ith data instance of D_T and $y_{Ti}, \in \mathcal{Y}_T$ is the corresponding class label for x_{Ti}. Further, the source task is notated as $\mathcal{T}_\mathcal{S}$, the target task as $\mathcal{T}_\mathcal{T}$, the source predictive function as $\mathrm{f_S}(\cdot)$, and the target predictive function as $\mathrm{f_T}(\cdot)$. Having all the needed elements,

a formal definition of transfer learning can be obtained. The following is borrowed from Pan and Yang (2010)[63].

Definition 1. *Given a source domain* $\mathrm{D_S}$ *and learning task* $\mathcal{T}_\mathcal{S}$*, a target domain* $\mathrm{D_T}$ *and learning task* $\mathcal{T}_\mathcal{T}$*, transfer learning aims to help improve the learning of the target predictive function* $\mathrm{f_T}(\cdot)$ *in* $\mathrm{D_T}$ *using the knowledge in* $\mathrm{D_S}$ *and* $\mathcal{T}_\mathcal{S}$*, where* $\mathrm{D_S} \neq \mathrm{D_T}$ *, or* $\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$*.*

Now, focusing on the last part of the definition, a categorization of different methods for implementing transfer learning can be drawn out. In principle, one can identify four different scenarios:

- $\mathcal{X}_\mathcal{S} \neq \mathcal{X}_\mathcal{T}$: the feature spaces of the source and target domains are different. (*heterogeneous transfer learning*)
- $\mathrm{P}(\mathcal{X}_\mathcal{S}) \neq \mathrm{P}(\mathcal{X}_\mathcal{T})$: the marginal probability distributions between domain data are different.
- $\mathcal{Y}_\mathcal{S} \neq \mathcal{Y}_\mathcal{T}$: the label spaces between the source and target tasks are different.
- $\mathrm{P}(\mathcal{Y}_\mathcal{S}|\mathcal{X}_\mathcal{S}) \neq \mathrm{P}(\mathcal{Y}_\mathcal{T}|\mathcal{X}_\mathcal{T})$: the conditional probability distribution of source and target tasks are different.

As Pan and Yang point out in their review [63], when practicing transfer learning it is essential to tackle the following research issues: (a) *what* to transfer, (b) *when* to transfer and (c) *how* to transfer.

(a) "What to transfer": this is the first and most crucial phase in the entire procedure. In order to increase the performance of the target task, it is essential to figure out which parts of the knowledge may be transferred from the source to the target. When addressing this issue, one has to try determining which aspects of knowledge are unique to the source and which are shared by the source and the target.

(b) "When to transfer": in some cases, transmitting knowledge just for the sake of doing it might make things worse rather than better (*negative transfer*). Transfer learning should be used to improve rather than deteriorate target task results. Some attention has to be paid when deciding whether it actually makes sense to transfer.

(c) "How to transfer": after the first two questions are answered, attention has to be paid on the actual possible ways of transferring knowledge across domains/tasks. This entails modifying current algorithms and employing other strategies.

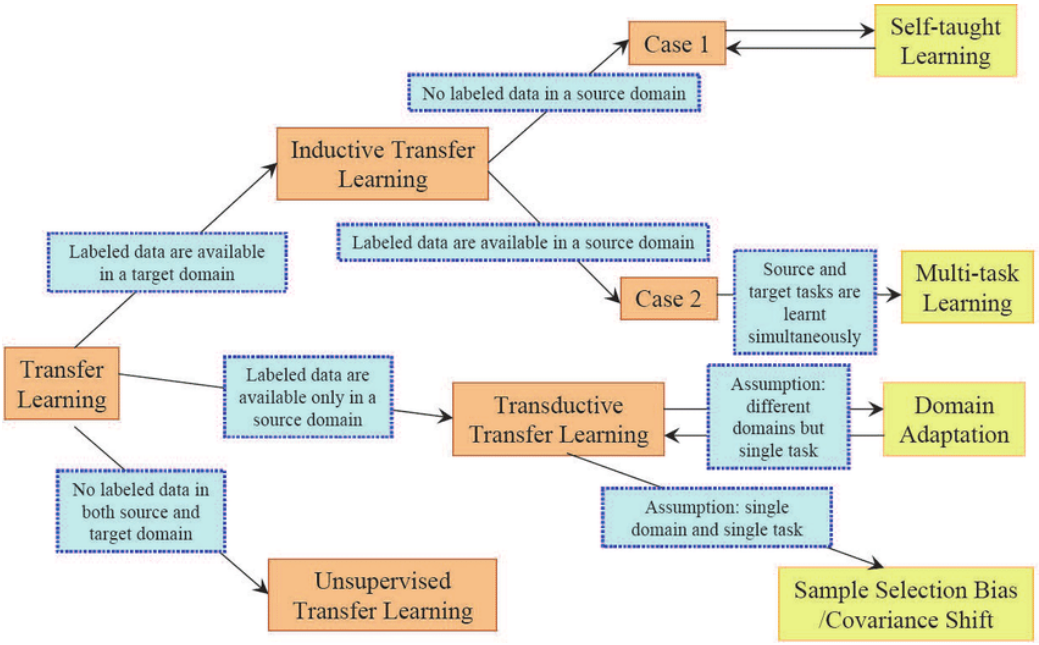


Figure 3.2: Overview of Transfer Learning settings presented by Pan and Yang [63]. Figure from [90]

To answer the when to transfer question, it is useful to identify the possible different scenarios. By doing this, three main settings emerge: inductive transfer learning, transductive transfer learning and unsupervised transfer learning.

Inductive Transfer Learning refers to the situation in which the source and target domains are the same, while the source and target tasks are not. The aim of transfer learning in this case is to improve the target task by using the source domain's inductive biases. This category can be further divided into two subsettings, comparable to multitask learning and self-taught learning, depending on whether the source domain contains labeled data or not.

Transductive Transfer Learning relates to the case where the source and target tasks are similar, but the associated domains are distinct. The source domain presents a lot of labeled data in this situation, but the target domain has none. This can be further divided into subcategories, which refer to situations in which the feature spaces or marginal probabilities are different.

Unsupervised Transfer Learning is similar to inductive transfer but focuses on unsupervised tasks in the target domain. The tasks are different, but the source and target domains are similar. Labeled data is not available in either domain in this case.

Figure 3.2 summarizes clearly the proposed classification of transfer learning methods according to the different scenarios.

As previously mentioned, another crucial question to ask is what to transfer.

- **Instance transfer**: In most circumstances, the data from the source domain cannot be directly reused. Instead, certain instances from the source domain can be combined with target data to improve performance, by reweighting. The two major methods in this circumstance: instance reweighting and importance sampling.

- **Feature-representation transfer**: This method is used to identify good feature representations that can be used from the source to the target domains, minimizing domain divergence and error rates. For feature-representation-based transfers, supervised or unsupervised approaches may be used depending on the availability of labeled data.

- **Parameter transfer**: This method works under the assumption that some parameters or prior distributions of the hyperparameters of the models are shared by the source and target tasks. Knowledge is transferred by being encoded into these priors or hyperparameters.

- **Relational knowledge-transfer**: This approach seeks to handle non independent and identically distributed data. These are cases where each data point has a relationship with other data points; As a result, the knowledge to be transferred is the relationship between data itself.

The table in figure 3.3, illustrates the possible transfer learning approaches depending on the different settings.

	Inductive Transfer Learning	Trasductive Transfer Learning	Unsupervised Transfer Learning
Instance Transfer	✔	✔	
Feature-representation transfer	✔	✔	✔
Parameter transfer	✔		
Relational knowledge-transfer	✔		

Figure 3.3: Different approaches used in different settings. Figure adapted from [63]

The final question to be answered is how to transfer knowledge. In practice, there are two major strategies to follow: feature extraction or fine tuning.

Feature extraction. Deep learning models are layered architectures that learn hierarchical representations of features. To obtain the final output, the layers of the network are finally connected to a last layer (usually a dense one). This layered design allows using a pre-trained network for various tasks without having to use its final layer as a fixed feature extractor. Instead it is common practice to use the representations learned by a given model by retraining only the last layers of the network. The main concept here is to extract features using the weighted layers of the pre-trained model rather than updating the weights of the model's layers during training with fresh

data for the new task. This technique is often referred to as "Off-the-shelf feature extraction" or "Cut-and-paste".

Fine-tuning. A more involved strategy for transfer learning consists in deliberately retraining some of the preceding layers in addition to replacing the final one. Saying that deep neural nets learn in a hierarchical fashion come from the fact that the first layers of neural nets have been seen to capture generic traits, whilst the subsequent and deeper layers of the net are more focused on the specific task at hand. This structure allows to freeze certain layers of the network while fine-tuning the others to match certain needs. In this kind of approach, knowledge is intended as the overall architecture of the network and it is used as a departure point for the re-training phase. As a result, when fine-tuning pretrained models, higher performance may be achieved with less training time.

3.1.2 Transfer Learning for GANs

When it comes to training generative adversarial networks (GANs) on high-quality images datasets, it is quite common to encounter issues in terms of the significant amount of computer power needed. Another problem relies in the difficulty of retrieving a dataset of suitable quality and size apt to training the model from scratch. Therefore, it is clear how transfer learning in this context can become particularly appealing. While this may sound quite immediate, understanding how to actually implement it is more challenging. Indeed, training of GANs is arguably still an open research question, and this reflects on how to perform transfer learning with these particularly complex architectures. Some of the most common techniques used within this framework will now be presented.

Fine-tuning [86]: Fine-tuning is the most intuitive and successful method of knowledge transfer and consists of setting the parameters of target models to the source models' pre-trained weights. Fine-tuning both the generator and the discriminator, according to the authors, yields the best results. This

technique, on the other hand, is prone to overfitting, therefore sufficient regularization could be required.

Scale/shift [61]: Considering that naive fine-tuning can lead to overfitting, researchers have proposed a new strategy that focuses on the batch statistics parameters scale and shift, of the generator's hidden layers. Indeed, only those parameters are updated, while the other weights are kept fixed. However, because of its limitations, this method frequently produces poor results, particularly when the source and target distributions differ significantly.

Generative latent optimization (GLO) [5]: This method arises from the idea that basing GANs loss on the discriminator's loss, through adversarial training, can be unreliable for limited data. Indeed, the authors proposed fine-tuning the generator through supervised learning, in a framework where the loss consists of the sum of the L1 loss and the perceptual loss. To prevent overfitting, GLO optimizes both the generator and the latent codes. While GLO enhances stability, because to the lack of adversarial loss (and previous knowledge of the source discriminator), it tends to create fuzzy pictures.

MineGAN [85]: Within this approach, authors present a novel architecture that includes a miner network that turns a multivariate normal distribution into a distribution on the input space of the pretrained GAN in such a way that the produced images mimic those of the target domain. When the source and target distributions share support, this importance-sampling-like technique can be successful, but it may not be applicable when their supports are disconnected.

FreezeD [57]: This recent technique for transferring knowledge in the context of GANs training builds on fine-tuning. In this case though, lower discriminator's layers are freezed. Intuitively, this works as the discriminator's bottom layers learn general image features, while the top layers learn to categorize the image as real or false depending on the extracted features, and are more specific to the task at hand.

3.2 StyleGAN2-ADA

In 2018, Karras *et al.* proposed a new GAN architecture, StyleGAN [45], that lead to tremendous increases in model's performances. While the continuous research improvements in GANs architectures were leading to images with an increasing high quality, controlling and regulating their output was still considered an issue. Take, for instance, the generation of images representing faces. Generating realistic faces was possible thanks to advancements in GAN literature, but doing so while also controlling for explicit features such as the shape of the face, the pose, hairstyles was still regarded as a fairly complex challenge. This is Karras *et al.* point of departure. Borrowing adaptive instance normalization from the literature on style transfer[37], they have redefined the generator architecture to obtain new ways for controlling the image generation process.

3.2.1 First formulation

The model builds upon their precedently proposed method of Progressive growing of GANs [42], allowing StyleGAN to generate the simulated image sequentially, originating from a low resolution and enlarging it to higher ones (1024×1024). The first change with respect to traditional GAN architecure lies in the network input. Indeed, usually the generator is fed directly a random vector z through the input layer. In StyleGAN, the fixed input consists of a d-dimensional independent Gaussian random vector with zero mean and variance one. The latter is firstly mapped to an intermediate latent space W through an 8-layer MLP. The learned affine transformations then specialize w to styles. The input latent space of traditional generators must follow the probability density of the training data. This, according to Karras *et al.*, leads to some degree of entanglement. This style space W instead, avoids at least partially this entanglement as it has different features of the image encoded along different orthogonal dimension in it. The style space is

then normalised through adaptive instance normalisation (AdaIN)[37], and fed at each convolution layer of the generator network.(A operations in figure 3.4). The AdaIN oprations is defined as:

$$\text{AdaIN}\left(\mathbf{x}_i, \mathbf{y}\right) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu\left(\mathbf{x}_i\right)}{\sigma\left(\mathbf{x}_i\right)} + \mathbf{y}_{b,i}$$

Where $\mathbf{y} = (\mathbf{y}_s, \mathbf{y}_b)$ are a pair of styles values, $\mathbf{x}_i$ is a feature map and μ and σ are its mean and the standard deviation respectively.

The other change to the discriminator is the injection of noise at each resolution to insert stochastic variation in the images. (B operations in figure 3.4). The following figure from [45] shows in detail the differences between traditional GAN generator and StyleGAN one.

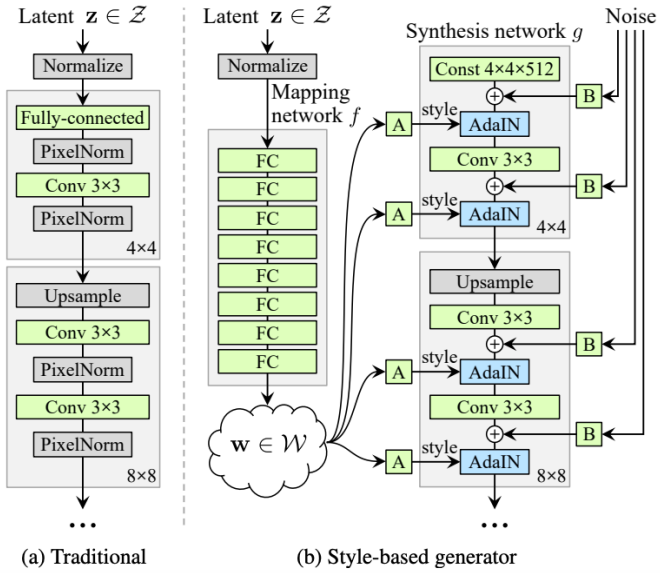


Figure 3.4: Traditional generator architecture (left), StyleGAN generator architecture (right). Figure from [45]

Other variations from the traditional model include:

- The replacement of nearest-neighbor up/downsampling with bilinear sampling in both the generator and the discriminator.

- Mixing regularization. Two random latent codes are used instead of one to generate a given percentage of images. This regularization method ensures that the network does not assume that adjacent styles are correlated.

3.2.2 Improvements: StyleGAN2

Despite the fact that StyleGAN outperformed the existing models in terms of performance, quality and resolution of the generated images, it still produced some common blob-like artifacts in the output. NVIDIA engineers then proposed an upgrade of the model with StyleGAN2 to address these issues [46]. Firstly, they identified the origin of the noticeable artifacts with the AdaIN operation. By normalizing the mean and variance of each feature map individually, it could potentially eliminate any mutual information contained in the features relative to each other. When the normalization is removed, the water like droplets indeed disappear from the generated images. AdaIN is therefore replaced with "weight demodulation". To understand this type of normalization, recall that each style block as presented in figure 3.5 is composed by modulation, convolution and normalization. Instead of scaling each feature map of the convolution, one can alternatively scale the convolution weights:

$$w'_{ijk} = s_i \cdot w_{ijk}$$

where w and w' are the original and modulated weights, respectively, s_i is the scale of the i-th feature map, j enumerates the output feature maps and k the spatial footprint of the convolution. To achieve the same goal of instance normalization, they start by noting that the output activations after modulation and convolution are scaled by the L2 norm of the corresponding weights at they have a standard deviation of:

$$\sigma_j = \sqrt{\sum_{i,k} {w'_{ijk}}^2}$$

The new weights are then normalized with the above standard deviation leading to:

$$w''_{ijk} = w'_{ijk} / \sqrt{\sum_{i,k} {w'_{ijk}}^2 + \epsilon}$$

where ϵ is a small constant value used to prevent some numerical issues. This operation is denoted as weight demodulation and by normalizing the output feature map to have a unit standard deviation, it reaches similar goals as other standardization methods as stable training. Other modifications include:

- Simplify the input constant by removing the application of bias, noise, and normalization.
- Moving the noise broadcast outside the style block.

The final revised architecture with a zoom on weight demodulation is shown in the following figure:

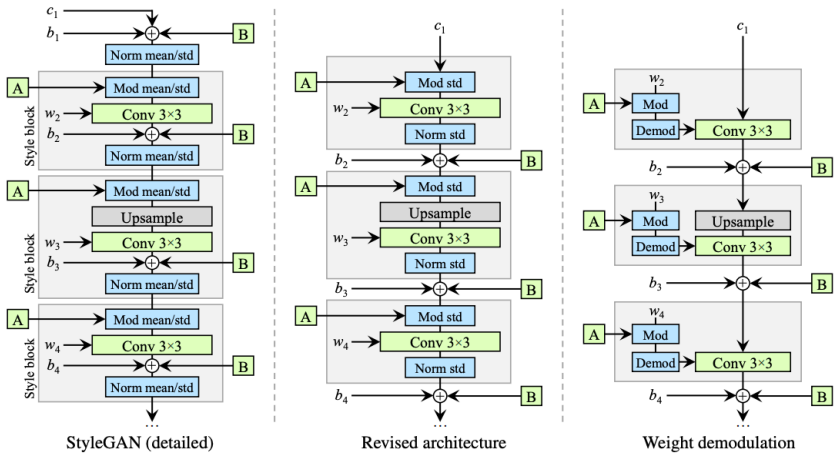


Figure 3.5: StyleGAN generator detailed architecture (left). StyleGAN2 revised architecture (center). StyleGAN2 weight demodulation operation (right) Figure from [46]

Finally, they removed progressive growing, as it caused artifacts by leading excessively high frequency in intermediate layers. Instead, they changed

G and D architectures and used a skip generator and a residual discriminator.

3.2.3 Final formulation: StyleGAN2-ADA

In 2020, another improvement of this model was presented. The purpose of the new changes was to allow obtaining good results in terms of quality of images generated by StyleGAN2 even when training with limited datasets. The proposed model is StyleGAN2-ADA[43], that stands for "Adaptive Discriminator Augmentation". Training GANs with limited dataset typically leads to a discriminator that quickly overpowers the generator, resulting in overfitting and difficulty in reaching convergence. This kind of issue, is generally resolved in other areas of deep learning through data augmentation. The problem is that these kind of operations when training GANs would lead to the generator reproducing the augmentations performed on the input dataset in the generated images, and this "leaking" of augmentations is highly undesirable. Karras *et al.* propose as a techinque to solve this issue stochastic discriminator augmentation. An intuitive explanation of this approach is the idea of putting distorting goggles on the discriminator and asking the generator to produce images that if viewed through these goggles will not be distinguished form the ones belonging to the training set. Indeed, augmentation is performed only on the images shown to the discriminator. This method works only when the distortions applied are obtained through an invertible transformation of the probability distributions over the data space. The following example taken from the original paper [43] helps in understanding what is meant by invertible:

> "It is crucial to understand that this does not mean that augmentations performed on individual images would need to be undoable. For instance, an augmentation as extreme as setting the input image to zero 90% of the time is invertible in the probability distribution sense: it would be easy, even for a human, to reason about the original distribution by ignoring black images

> until only 10% of the images remain. On the other hand, random rotations chosen uniformly from 0°, 90°, 180°, 270° are not invertible: it is impossible to discern differences among the orientations after the augmentation. The situation changes if this rotation is only executed at a probability $p < 1$: this increases the relative occurrence of 0°, and now the augmented distributions can match only if the generated images have correct orientation."

Ideally, if the augmentation operator T is "invertible", there should be no leaks in generated images. In practice though, very high values of p (parameter that controls the augmentation strength) lead to increment leaks in the generated images. The authors also show that the optimal value of p is highly sensitive to the size of the dataset. They tackle the issues of doing an expensive grid search on p by making the process adaptive. Their aim was avoiding manual tuning of p by controlling it dynamically based of overfitting. Denoting the discriminator output as D_{train}, they use the following overfitting heuristic:

$$r_t = \mathbb{E}\left[\text{sign}\left(D_{\text{train}}\right)\right]$$

It estimates the fraction of the training set that obtains positive discriminator outputs. When r=1 there is complete overfitting. They controlled p by initializing it to zero and adjusting it every four minibatches based on the value of r_t. If it is too high/low, p is incremented/decremented by a fixed amount. Through this adaptive discriminator augmentation, training is stabilized and the quality of images generated when training with limited dataset is considerably improved.

3.3 Holistically-nested Edge Detection

Edge detection is a technique used to detect points in a digital image where the light intensity changes abruptly. Sudden changes in the properties of a digital image are usually a manifestation of important events or shifts in physical representation of the image. These changes can be for example: discontinuity of depth, discontinuity of surface orientation, change in material properties, and changes in illumination from the surrounding environment. Edge recognition is a research field of image processing and computer vision, particularly the branch of feature extraction that is focused on identifying and depicting these changes. One of the commonly methods employed for this purpose is Holistically-nested Edge Detection (HED), presented by Xie and Tu in 2015 [89]. Specifically, it is a deep learning model apt to obtain image-to-image predictions through fully convolutional neural nets and deeply-supervised networks. The literature concerning edge detection methods is quite rich, but the main aim of the authors with HED was to address two issues:

- Holistic image training and prediction, using fully convolutional nets.
- Lerning features in a nested multi-scale fashion, through deeply supervised networks.

The technique proposed is eventually able to take an image as an input and directly outputs the corresponding edge map.

3.3.1 Network architecture

The choice for the architecture of this model arises from meeting two main needs. Firstly, the model has to be deep in order to generate in an efficient manner perceptually multi-level features. Secondly, to reproduce the intrinsic scales of edge maps, it needs to have multiple steps with distinct strides. A model that generally meets these criteria and achieves state-of-the-art

performances is VGGNet [77]. Indeed, it is quite deep (16 conv layers), it presents multiple stages (five 2-stride downsampling layers) and great density (stride-1 convolutional kernels). The authors added these modifications with respect to the original model:

- In each layer stage, the side output layer is connected with the last convolutional one. The conv layers present a receptive field size equal to the corresponding side-outpput layer.
- The last stage of VGGNet, 5th pooling layer and fully connected ones included, are removed.

The final HED model architecture is a VGGNet in which 5 stages that have respectively stride 1, 2, 4, 8, 16, with distinct receptive field sizes, have been nested. These architectural choices lead to a technique for edge detection that is both accurate and computationally efficient. Figure 3.6 shows an example taken from the original paper of an edge map produced by HED.

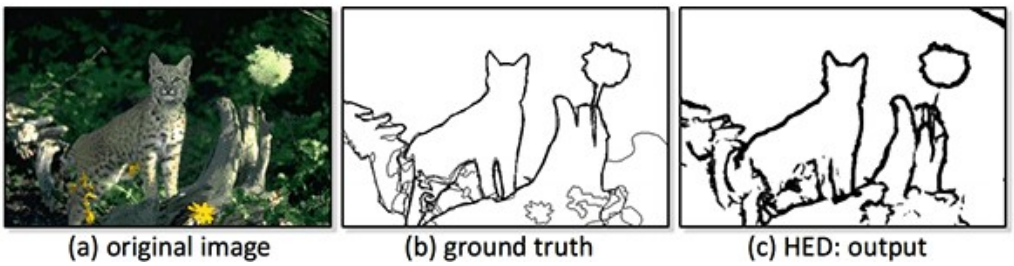


Figure 3.6: (a) a sample test image in the BSD500 dataset [53]; (b) corresponding edges as annotated by human subjects; (c) the HED results. Figure from [89]

3.4 pix2pixHD

Image to image translation is one of the many applications of generative adversarial networks. As mentioned in section 1.2, one of the most used

methods to implement it is pix2pix, proposed by Isola *et al.* [39]. In the current section, an updated version of this framework will be presented.
pix2pixHD was proposed by Ph.D. researchers from the University of California, Berkeley in coordination with NVIDIA Corporation in 2017 [84]. The main reason behind the proposed enhancements with respect to the precedent version of the method are two. The first is that the framework ideated by Isola *et al.* [39] was not able to generate high resolution images. Chen and Koltun [14] tried to overcome this issue by adopting a modified perceptual loss to synthesize images. The resulted images, although in high resolution, presented defects in both details and realistic textures. This is indeed the second motivation that pushed the authors of pix2pixHD to improve the pre-existing techniques.

The original pix2pix framework is a conditional GAN (cGAN) consisting of a generator G and a discriminator D operating in a supervised setting. The objective of G is to convert semantic label maps into realistic-looking pictures, while D tries to discriminate the real images from the translated ones. The training set is composed by a set of paired images $\{(s_i, x_i)\}$, where s_i is the semantic map and x_i is the corresponding real image. cGANs' goal is to model the conditional distribution of the real images given the maps of input semantic labels through a modified version of the aforementioned minimax game from Eq.2.1 in Section 2.1, illustrated here below.

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D), \tag{3.1}$$

where $\mathcal{L}_{\text{GAN}}(G, D)$ corresponds to:

$$\mathbb{E}_{(\mathbf{s},\mathbf{x})}[\log D(\mathbf{s}, \mathbf{x})] + \mathbb{E}_{\mathbf{s}}[\log(1 - D(\mathbf{s}, G(\mathbf{s}))].$$

The architecture used are a U-net generator and a patch-based fully convolutional network discriminator. Starting from this point the authors of pix2pixHD added some modifications.

3.4.1 Coarse-to-fine generator

Firstly the authors of pix2pixHD, split the generator in two networks: a global generator G_1 which is wrapped around by a local enhancer G_2. The G_1 network outputs images with a resolution of 1024 $\times$ 512, and the G_2 network operates at 4$\times$ the output size of the previous one (2$\times$ along each image dimension). The architecture of the global generator is inspired by the one proposed by Johnson *et al.* [41] and consists of three elements:

- Convolutional front-end $G_1^{(F)}$
- Set of residual blocks $G_1^{(R)}$
- Transposed convolutional back-end $G_1^{(B)}$

The semantic label map is passed to all three components of G_1 sequentially. Similarly, the local enhancer network comprises 3 components:

- Convolutional front-end $G_2^{(F)}$
- Set of residual blocks $G_2^{(R)}$
- Transposed convolutional back-end $G_2^{(B)}$

To facilitate the integration of global information from G_1 to G_2, the set of residual block of the local enhancer $G_2^{(R)}$ is fed with a element-wise sum of the output feature map of $G_2^{(F)}$ and the last feature map of $G_1^{(B)}$. Training starts from the global generator and then continues with the local enhancer. The networks are subsequently fine-tuned together. Thanks to this architecture, global and local information are both taken into account when generating the image. Figure 3.7 illustrates the coarse-to-fine $G = \{G_1, G_2\}$ architecture.

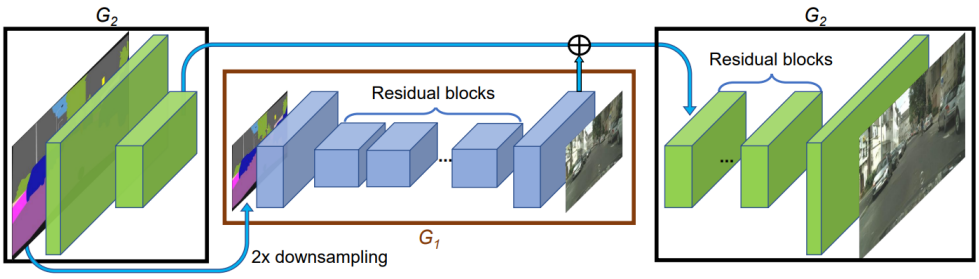


Figure 3.7: Network architecture of the coarse-to-fine generator. Figure from [84]

3.4.2 Multi-scale discriminators

When generating high resolution images with GANs, the discriminator has to have a large receptive field [84]. Since common solutions to this issue lead to overfitting, the authors proposed using 3 discriminators (D_1, D_2, D_3) that operate at different image scales while maintaining the same structure. After downsampling by a factor of 2 and 4 the real and generated images to obtain an image pyramid of 3 scales, each scale is fed to one of the three discriminators. This operation, by letting the discriminator look at the image through different levels of receptive fields, allows the generator to learn the details of an image at different levels. The learning problem from Eq.3.1, becomes:

$$\min_{G} \max_{D_1,D_2,D_3} \sum_{k=1,2,3} \mathcal{L}_{\text{GAN}}\left(G, D_k\right). \tag{3.2}$$

3.4.3 Improved adversarial loss

The standard GAN loss, as presented in Eq. 3.1, does not pay attention to the intermediate representations. In principle, allowing this could enable the generation of more realistic textures and details representation during downsampling. The authors proposed as a solution to this problem a modified version of the loss, specifically a feature matching loss based on the

discriminator. In practice, they learn to match intermediate representations extracted from multiple layers of the discriminator from the real and generated images.

The feature matching loss $\mathcal{L}_{\text{FM}}(G, D_k)$ becomes:

$$\mathcal{L}_{\text{FM}}(G, D_k) = \mathbb{E}_{(\mathbf{s},\mathbf{x})} \sum_{i=1}^{T} \frac{1}{N_i} \left[\left\| D_k^{(i)}(\mathbf{s}, \mathbf{x}) - D_k^{(i)}(\mathbf{s}, G(\mathbf{s})) \right\|_1 \right]. \tag{3.3}$$

Where $D_k^{(i)}$ denotes the ith-layer discriminator's feature extractor, T is the number of layers and N_i is the number of elements per each layer. The authors finally illustrate how the full objective includes the GAN loss and the feature matching loss:

Our full objective combines both GAN loss and feature matching loss as:

$$\min_G \left(\left(\max_{D_1,D_2,D_3} \sum_{k=1,2,3} \mathcal{L}_{\text{GAN}}(G, D_k) \right) + \lambda \sum_{k=1,2,3} \mathcal{L}_{\text{FM}}(G, D_k) \right). \tag{3.4}$$

The improved architecture of pix2pixHD led to amazing results both in terms of resolution and level of detail of the images produced. The following figure presents an example of synthesized image by the presented network.

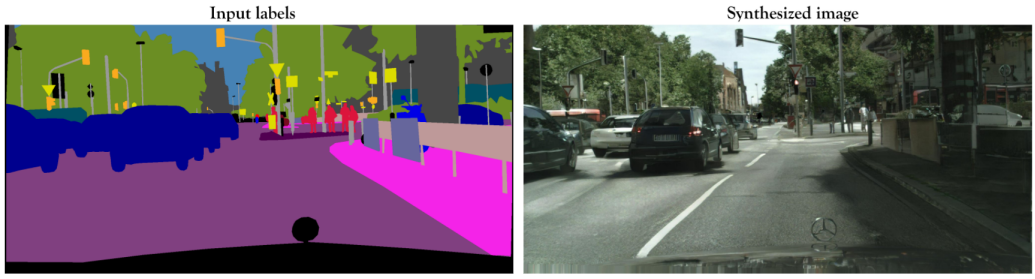


Figure 3.8: Sample result of a translation from semantic label to cityscape street view implemented with pix2pixHD. Figure from [84]

Chapter 4

Experiments: generating realistic paintings with GANs

As discussed in the previous chapters, the potential of using artificial intelligence for the production of visual arts is constantly growing. Indeed, thanks to recent technological advances in machine learning, it is possible to generate high-quality images. However, research has only partially focused on developing these models for artistic purposes. This is the starting point of the present research, whose aim is to investigate which are the best models and techniques to generate realistic paintings using AI.

Firstly, the dataset used will be presented, specifically it consists of 5000 impressionistic landscape paintings. Afterwards, the following models will be developed:

- A simple DCGAN as a baseline.
- StyleGAN2-ADA trained from scratch.
- StyleGAN2-ADA used with transfer learning in two versions: fine-tuning of all layers and freezing 13 layers of the discriminator (FreezeD version).

- A composite model arising from 3 steps:
  - Getting sketches of the paintings using HED to obtain edge maps of the paintings present in the dataset.
  - StyleGAN2-ADA trained from scratch to generate new sketches.
  - Pix2pixHD to transform the generated sketches into paintings.

Subsequently, the outputs produced by these models will be evaluated both in quantitative terms through the calculation of the Frechét Inception Distance (FID), and in qualitative terms through a Turing test.

Finally, the different results obtained will be compared in order to define which is the most suitable method to generate realistic paintings using AI. All the experiments carried out for this research have been run on the Google Colab platform, and the linked code and notebooks can be found at the Github repo of the project[6].

4.1 Dataset overview

Choosing the dataset for this experiment is one of the most crucial steps in the entire research. Partial results of the one of the various tests with different datasets that have been carried are available in the appendix [A.2]. However, the final in-depth analysis has been performed on one dataset. Specifically, it consists of 5000 impressionistic landscape paintings and it was retrieved on Kaggle [30]. The choice of this art style and genre is motivated by the ease of retrieving an adequate number (5000) of high quality (1024x1024px) images of paintings of this type.

4.1.1 Data gathering and preprocessing

The procedure described in this section, has been carried by the author of the dataset Robert Gonsalves [30]. Nevertheless, for all the dataset retrieved for

experiments, the approach followed has been the same. The first step was to use a custom Python script to scrape landscape paintings from WikiArt.org. The script alphabetizes all of the artists on the site. It checks to see whether the artist was born after 1800 and died before 1950, and if they were a member of the "Impressionism" art movement. Subsequently, the script loops over all of the artists' works, seeking for ones that are in the public domain and are in the "landscape" genre. Finally, the script downloads each qualifying picture, naming the file after the artist and artwork name. About 5,300 artworks fulfilled these criteria, according to the algorithm.

The images retrieved had various aspect ratios. For efficiency reasons though, it is common practice to feed deep learning models square images with dimensions corresponding to powers of two. For this reason a preprocessing step had to be performed. There are three main methods for making the modifications, each with its own set of benefits and drawbacks. Figure 4.1 depicts an example of each of the three main approaches for resizing the painting.

Figure 4.1: *L'Etang de Chevreuil* by Alfred Sisley and the three resized versions. The image on the top row is the original painting. In the bottom row from the left: letter box format, center crop, squeezed version.

The image in the "letterbox" format preserves the image uncut and unsqueezed, but it is squared by adding black bars above and below. This is an issue as the part of the resolution of the image is lost and the black sections of the image are "wasted" during training.
The picture in the "center cut" format is cropped to maintain only the image's square center. The problem with this format is that it loses a lot of the image content on the left and right sides.
The third version presents a "squeezed" format. It has been resized to the desired height and width, by stretching and compressing the original ones. In this case there is no loss of content, but the image is visibly deformed. Since the images present in the dataset are all in different aspect ratios, each of them will be squeezed by a different amount.

The author of the dataset, came up with a hybrid approach for resizing the images, obtaining a good trade-off in terms of not losing too much imagery

and of resolution.
To begin, he calculated the average aspect ratio of all the original paintings, which in this case is 1.27:1. The images were then cropped with this 1.27:1 aspect ratio, extracting a central rectangle. Subsequently they have been compressed horizontally into a square format with a size of 1024 $\times$ 1024 pixels. The resulted images are the ones in the final training format. Thanks to this technique, when the GAN will generate square output pictures, they will simply be rescaled to 1.27:1 to match the original format. Ultimately, the outcome of the models, with this method, will be distortion-free synthetically generated pictures. The final version of the sample painting shown above is presented in Figure 4.2.

Figure 4.2: *L'Etang de Chevreuil* by Alfred Sisley in the obtained hybrid format.

4.1.2 CLIP-based representativeness selection

Scrolling through the images fetched by the scraper, it is easy to notice how some of them do not fully represent a painting of an impressionistic landscape. For this reason, the author of the dataset went through one more

final filtering step. He used the aforementioned technology CLIP [65], by comparing the embedding for the phrase "impressionist landscape painting" to the embeddings from the 5300 paintings.

Figure 4.3: Sample images with the lowest CLIP score for the input phrase *"impressionist landscape painting"*

Thanks to this procedure, he was able to identify the images that best fit the sentence. He eventually ended up with a dataset of 5000 impressionistic landscape paintings, that is the training dataset utilized in the current research. The following figure depicts some examples of images present in the final dataset.

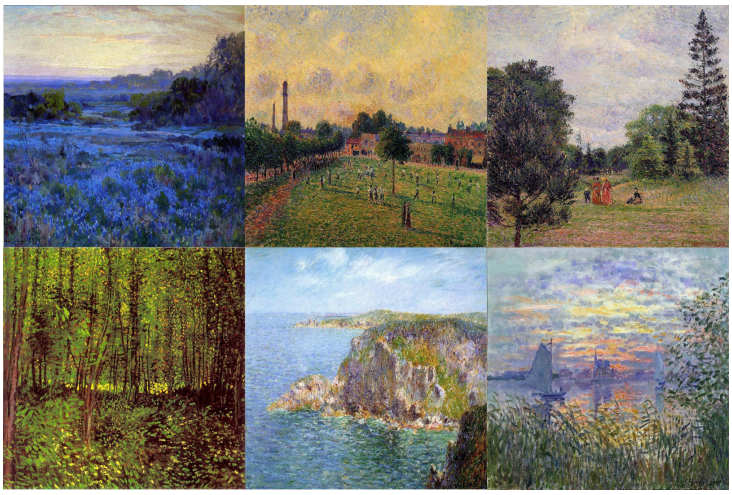


Figure 4.4: Sample images from the training dataset

4.2 A GAN baseline

The first experiment carried out, is aimed at getting more of an idea of the results obtainable from one of the first formulations of adversarial networks: a baseline DCGAN.

4.2.1 Baseline GAN

As the baseline model is a GAN that produces 128x128 images, an ulterior pre-processing step has to be performed. Specifically the 1024x1024 images in the dataset have to be resized to 128x128 pixels for training. After this step the generator and discriminator networks have been defined.

Generator. The generator network takes as an input a noise vector. For this reason the fist layer is a fully connected dense layer of size 4096. Therefore, the function of the generator itself takes as input the noise size and channel number (in the case of this experiment, being the images RGB, it equal to 3). The next layer is a Reshape one that transforms the shape of the dense layer to a (4 * 4 * 256) layer (for this reason the dense one was defined with a shape of 4096 = 4 * 4 * 256). The next layers are block composed by an upsampling layer, a convolutional layer, batch normalization and activations function relu. This blocks are stacked to obtain a final output of shape 128x128 - that is the training data shape. This in turn means that this network takes random noise as and input and outputs a 128x128 image. A summary of the generator architecture is presented in the appendix [A.3].

Discriminator. The discriminator takes as input either training images or generated ones. Therefore, the function is designed to take the image shape as a parameter. Inside the function, a Sequential model from keras is initialized to build linear stacks of layers. Firstly there is a convolutional layers of shape 32, with kernel size 3 and stride of 2 with padding "same". In simpler terms: the conv layer has a filter size of 3x3 that strides over the image data. Moreover, as it is a first layer it contains the input shape

parameter. Since the padding is set to "same", there is no additional padding. Following, a LeakyRely layer as activation function and a dropout layer to prevent overfitting are added. The next layers blocks are all built with a conv layer, batch normalization, LeakyRelu activation and dropout. Since our discriminator's job is to classify whether the given image is fake or not, it is a binary classification task and sigmoid is an activation that squeezes every value to values between 0 and 1. Because the aim of the discriminator is to classify as real or fake the image it receives as an input, this translates into a binary classification task. As a consequence, the final layer of the discriminator network is a fully connected one with a sigmoid activation function that squeezes output values between 0 and 1.

Adversarial Training. After initializing the two networks, a helper function was designed to save images after some iterations. The function takes as input cnt, and noise. The first parameter represents the number of iterations to be performed in order to save a partial output, while noise is the input vector to generate the images. The final step to build the GAN is combining the generator and discriminator networks through adversarial training. Firstly, the generator model is compiled by passing as input the random noise vector. Then, the discriminator is compiled with a loss function, an optimizer and a performance metric. Specifically, following the original formulation of GANs [34], the chosen loss is binary cross entropy, the selected metric is accuracy and the optimizer is Adam [49], a popular extension to stochastic gradient descent. The latter is initialized with a learning rate of 1.5e-4 and a beta1 equal to 0.5. At this step, the discriminator weights are frozen. Indeed, as mentioned in section 2.2.1, while the generator is training, the discriminator must be kept constant. If this step is skipped, not only the generator will update its weights to better fool the discriminator, but also the discriminator weights will be adjusted to increase the ability of being fooled. The two models are then combined and compiled with the aforementioned loss, optimizer and performance metric.

While training, the discriminator is trained separately on real and fake images, and its final metric is computed as the average of the two corresponding performance metric. For a deeper insight on the functioning of this network the reader is invited to check out the code through which this model was implemented on the github repo of this project [6].

Results. The design of this baseline model is quite a simple one. Indeed, the output produced is rather poor, but it still provides a support ground to better grasp the power of the models that would later be developed. Around 10,000 epochs the training seems to stabilize, with a discriminator that overpowers the generator, bringing the final image to be fairly scarce in terms of quality. Some sample synthesized images are illustrated in figure 4.5. In the case of GAN, visual inspection of the quality of the output produced is the first step in evaluating the model's ability to generated images. Mind that producing visually satisfying outputs is a necessary but not sufficient condition for evaluating the model performance. Since in this case the generated paintings do not reach adequate results in terms of visual image quality, the baseline model will not be further investigated.

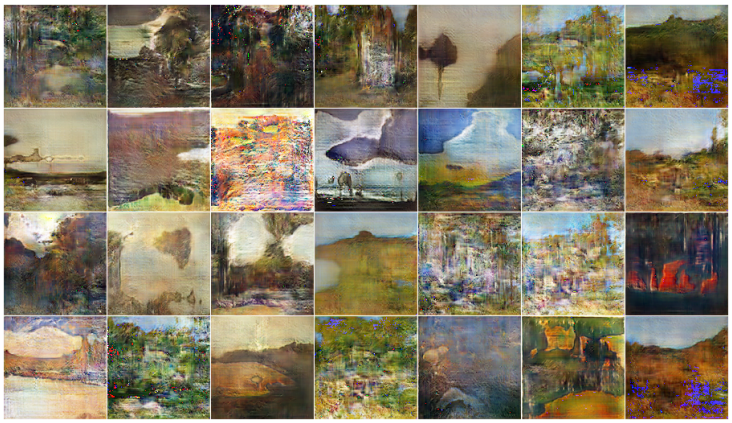


Figure 4.5: Sample results of the baseline GAN model.

4.3 Model 1: StyleGAN2-ADA trained from scratch

The experiment continues with the use of state-of-the-art StyleGAN2-ADA developed by NVIDIA engineers [43]. For a thorough explanation of the model's architecture the reader can refer to section 3.2 of the current research. The source code of the model can be found at StyleGAN2-ADA github repo[44], but the current experiment has been performed relying mostly on Derrick Schultz's tensorflow implementation[74].

According to its authors, the main contribution of this model is its ability to obtain state-of-the-art results even when trained on limited datasets. As the training data for the current project consist of 4500 images (500 images are left out as a test set), this network should in theory yield a satisfying outcome.

Model setup. The first step to proceed with training the model is change the format of the data. StyleGAN models indeed, take as input datasets stored as multi-resolution TFRecords. After this operation training begins. Different combination of parameters have been tested, but here presented is the one that lead to the preferred results. In addition to specifying the path to the data folder, output directory and how often a .pkl file should be created and saved along with a sample of the generated images, the following parameters are set:

- *augpipe = 'bgc'*. Augmentation pipeline used. 'bgc' means that pixel blitting, geometric transformations and color transformations are applied. This is the principal parameter changed during the various experiments, but the 'bgc' configuration seemed to lead to the best results.
- *mirror = True*. Performs x-flips of images for augmentation.
- *mirrory = False*. Does not perform y-flips of images for augmentation.

It could be useful to compute some pre-defined metrics while training, but when this was tried through the parameter *metrics*, the Google Colab session interrupted due to memory usage limitations, and is therefore set to 'None' throughout all the following experiments. Training lasted around 90 hours ($\sim$ 4 days). The choice of this training time arose from limited computational and time resources, and seemed like a good compromise in terms of resources used vs results obtained. Indeed, as it is mentioned in the official github repo [44], StyleGAN2-ADA trained with one GPU on a 1024x1024 resolution image dataset, typically needs around 25000kimg to be shown to the discriminator too reach convergence (a bit more than 40 days of training), but reasonable results are expected to be obtained already at 5000kimg. As ColabPro runtime session times out easily, this operation took around one week. Every time the session timed out, the training could be easily restarted from the last .pkl file saved through the parameter *'resume'*. Figure 4.6 presents some examples of the paintings generated by this model.

Figure 4.6: Sample paintings generated with StyleGAN2-ADA trained from scratch for around 90 hours.

4.4 Model 2: Transfer learning with StyleGAN2-ADA

As widely discussed in section 3.1 of the current research, transfer learning can lead to incredible improvements in terms of efficiency while performing a certain task. This experiment aims at assessing whether transferring the

knowledge acquired while training on photographic images with very different subjects from the ones present in the impressionistic landscape dataset leads to better performances and faster times for convergence.

The StyleGAN2-ADA model enables transfer learning by specifying the same parameter mentioned in the previous section *'resume'*. By passing 'ffhq1024' to this parameter, the training on the novel dataset resumes from the network pre-trained on the 'Flickr-Faces-HQ Dataset', originally presented in the StyleGAN paper [45]. This dataset is composed by 70.000 high-quality PNG images at 1024×1024 resolution representing human faces. Another parameter that refers to transfer learning is *freezed*. It refers to the number of discriminator lower layers to be frozen. The authors of the model added this feature following the prominent results obtained by Mo *et al.*[57]. The default setting is with *freezed = 0*, but in line with Karras *et al.* findings in [43], another test with *freezed = 13* (meaning that all layers at the four highest resolutions are frozen) was carried out. The first experiment will be referred as *fine-tuning*, while the latter as *freeze-D*. As for the remaining parameters, the same configuration that was used while training the model from scratch was adopted.

Freeze-D. Unlike the expected outcome for this technique, the results obtained by freezing the first 13 layers of the discriminator while transferring knowledge are quite poor. The images produces are blurry and the subjects are not well defined. Visual artifacts are evident and overall the quality of the image is not the desired one even after a couple of days of training time.

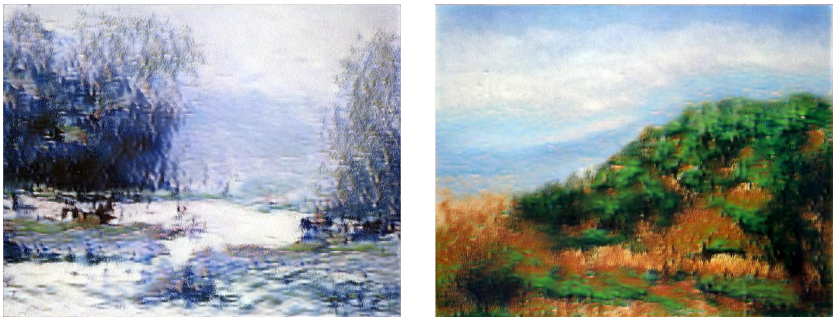


Figure 4.7: Sample paintings generated with transfer learning with the Freeze-D configuration.

As the resulted outcome does not satisfy the expected visual quality of the paintings this configuration will not be further investigated.

Fine-tuning. The second transfer learning configuration was set following a fine-tuning approach. No layers of the discriminator were frozen. The remaining parameters are again set with the same configuration used when training the model from scratch.
Fine-tuning the pretrained model led to excellent results in terms of reduction of training time. For better comparison though, the model has been trained for 90 hours just as the one trained from scratch. Figure 4.8 presents examples for both partial (12h) and final (90h) results.

(a) (b)

Figure 4.8: Partial and final results obtained by transfer learning performed with a fine-tuning approach. (a) Images produced after 12 hours of training. (b) Images produced after 90 hours of training.

4.5 Model 3: 3 steps GAN

This last formulation follows the idea proposed by Xue in [91]. In practice, it mimics the process that human artist usually go through. First they draw a sketch, then they paint. In the generative models domain this translates into defining what is a sketch, generating the defined sketch and finally transforming it into a painting. In this experiment this procedure takes form in three main steps:

1. Obtaining sketches of the 5.000 paintings dataset.
2. Generating novel sketches with StyleGAN2-ADA.
3. Transforming the sketches into paintings with pix2pixHD.

4.5.1 HED maps as sketches

The first step consists in obtaining some form of sketch for the dataset at hand. One direct way to achieve this is to perform edge detection and consequently obtain edge maps of the original image. As discussed in section 3.3 of the present work, to which the reader can refer to for an in-depth explanation of this method, HED is a commonly used technique to implement edge detection. Figure 4.9 presents an example of the edge map obtained with this approach.

Figure 4.9: HED edge map obtained for Claude Monet's *Field of poppies*.

With this technique, sketches for all of the 5.000 paintings constituting the dataset have been produced.

4.5.2 Generating novel sketches

The newly obtained sketch dataset has been subsequently used to train StyleGAN2-ADA to obtain novel sketches. The parameters configuration in this case has slightly changed with respect to the one used when training on the real paintings. Specifically, the augmentation pipeline did not include color transformations.

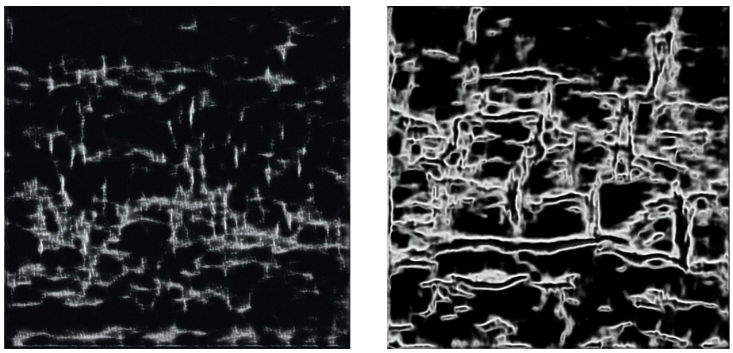


Figure 4.10: Samples of HED edge maps sketches generated by StyleGAN2-ADA. On the left a sample produced after 90 hours of training and on the right one produced after 30 hours.

The model, even after many iterations, did not seem to be able to learn to produce realistic sketches. Different parameters configurations have been tested but all leading to scarce results. Some of the common issues of GANs were encountered during training, such as mode collapse. From a technical point of view, this failure was quite unexpected. For this reason a more in-depth investigation about what caused the failure while training will be performed outside of the context of this research. Figure shows some of the sketches generated by the model. Nevertheless, the experiment with Model 3 continued. The reason behind this choice is mainly curiosity. Indeed, the context for this research is the intersection of art and artificial intelligence. The idea is to explore the possible outcome more in artistic and less technical ones.

4.5.3 From novel sketches to novel paintings

The novel sketches obtained, probably due to failure in convergence during training, are more abstract in nature rather than resembling a landscape painting sketch. This experiment seeks to investigate the possible outcome of passing these abstract sketches in a pix2pix model to obtain the corresponding painted version of them.

In order to do so, a pix2pixHD model has been trained over the whole dataset, meaning the 5000 pairs of HED generated sketches and corresponding real painting. The model has been trained for 88 epochs, leading to a total of 440.000 steps performed (1 epoch corresponds to iterating over the whole dataset). An example of what the model was able to produce around this epoch is presented in figure 4.11.

(a) input sketch (b) real image (c) synthesized image

Figure 4.11: Sample output of pix2pixHD trained for around 86 epochs compared with the real image that originally generated the HED sketch.

After this step, some of the generated sketches in section 4.5.1 were passed through the model to obtain their painted version. Some curious results emerges, but as expected, the paintings are abstract in nature. Figure 4.12 presents two version of painted sketches that have been previously generated from different training epochs of the StyleGAN2-ADA model. Following the line of thought introduced in the previous section, the results of Model 3, even if poor when observed under technical lenses, could potentially have interesting implications when showed to human subjects. For this reason, a quantitative performance metric will not be calculated, but a painting generated with this model will be presented for qualitative evaluation by human subjects.

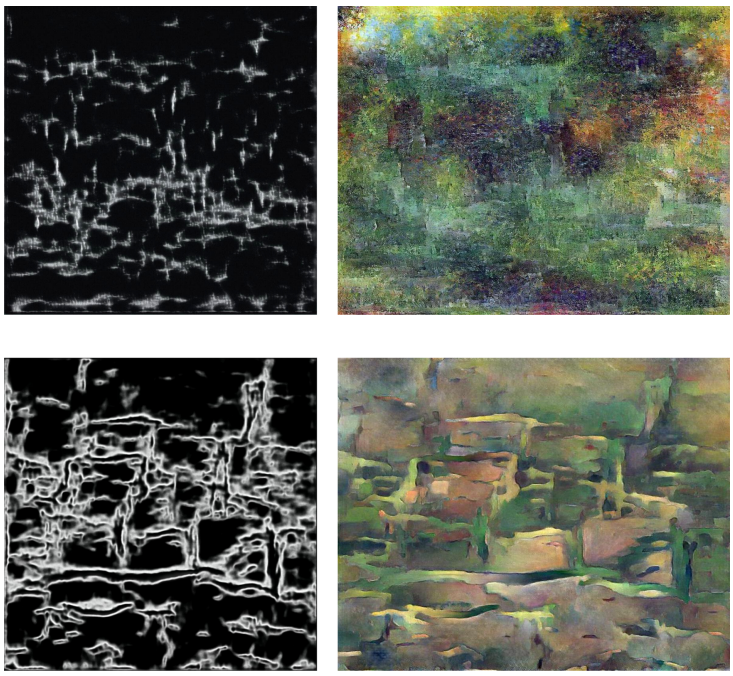


Figure 4.12: Different results obtained from passing sketches generated by StyleGAN2-ADA trained for 90 hours (top) and trained for around 30 hours (bottom).

4.6 Best performing models evaluation

4.6.1 Quantitative metric: Fréchet Inception Distance

A commonly used metric for GAN evaluation is the Fréchet Inception Distance FID [62], presented for the first time by Heusel *et al.* in their 2017 paper named "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." [35]. It was proposed as an improvement with respect to the widely used Inception Score IS [71]. Indeed, the latter estimates the quality of a set of generated images without capturing how these synthetic images compare to real ones. The FID takes it one step further providing a measure of similarity between the real and generated images.

In more formal terms, the FID computes the squared Wasserstein distance between two multivariate Gaussians: $\mathcal{N}(\mu_G, \Sigma_G)$, the distribution of the extracted features of the generated images and $\mathcal{N}(\mu_R, \Sigma_R)$, the distribution of the extracted features from the real images. To do it, the activation of an intermediate layer of Inception V3 model pretrained on Imagenet is used. Hence, the FID can be calculated from the mean and covariance of the activations computed when the generated and the real images are fed to the Inception network, as:

$$\text{FID} = \|\mu_G - \mu_R\|_2^2 + \text{Tr}\left(\Sigma_G + \Sigma_R - 2\left(\Sigma_G^{1/2}\Sigma_R\Sigma_G^{1/2}\right)^{1/2}\right)$$

where G and R are the fake and real embeddings (activation from the Inception model) assumed to be two multivariate normal distributions, μ_G and μ_R are the magnitudes of the vectors G and R, Tr is the trace of the matrix and Σ_G and Σ_R are the covariances of the vectors.

The FID has been calculated for Model 1 and the fine-tuning version of Model 2. As Model 3 presented clear issues while training, and generated visibly different images, it is considered to be pointless to measure the FID for this model.

	Model 1	**Model 2**
FID	31.55	**25.96**

Table 4.1: FID calculated for the different models. Model 2 (transfer learning with fine tuning) leads to the lowest FID score.

The calculated FID are presented in Table 4.1. Now, a remark should be stated. The above measured metrics, are computed against the whole dataset, including training images. This approach is commonly advised against, as one should not evaluate performances on the training dataset, but as it often happens in computer vision tasks, the test dataset at hand was too small. Indeed, to calculate the Gaussian statistics necessary to com-

pute FID, a minimal test size should be of at least 2048 images, as it should be greater than the dimension of the coding layer that for the Inception V3 pooling layer is of 2048. If this is not the case, the covariance is not full rank resulting in complex numbers and nans when calculating the square root.

Having this remark in mind, it is clear how the computed metrics are not completely reliable on their own, and are probably underestimating the real FID, but they can help in giving a general idea of which model produces the images that are most similar to the real ones.

For the aforementioned reasons, model performance evaluation will be integrated with the findings of the survey described in the following section.

4.6.2 Turing test and results

A common evaluation method for AI generated art in current literature [25] [51], consists in what could be considered a Turing test. The main idea is showing participants in the chosen type of experiment, such as a survey, a quiz or real life exposition, both AI generated artworks and human created paintings. The participant is then asked whether she/he believes that the artwork was produced by a human or a computer.
As the current research investigates the ability of GANs to produce realistic paintings, this kind of evaluation approach suits well as a qualitative performance metric.

Similarly to the previous section, the models under analysis are Model 1 (StyleGAN2-ADA trained from scratch) and Model 2 (StyleGAN2-ADA with transfer learning). In addition, a painting produced by Model 3 is also displayed in the survey. The model definitely encountered problems in training and did not lead to the desired results. However, following the perspective of this research in which the artistic potential of artificial intelligence is investigated, it would be interesting to study what participants thought of the output produced. When looking at the generated image through a more artistic and less technical lens, one could also consider these errors as

territory with a promising exploratory prospective.

Survey Design

The questionnaire was distributed as a scored quiz on various social platforms and can be accessed with the following link: **AI-art QUIZ**.

To be able to present different outputs of the various models without making the quiz longer, two similar versions of the quiz were built and the access to each of them was randomized. The only difference between the two versions is the actual paintings showed to the participant but the structure is the same and can be summarized as follows.

The first screen that appears upon opening the quiz states: "I am going to show you some paintings. Can you guess which of them have been painted by a human and which have been generated by a computer? At the end of the quiz you will see how well you did."

Subsequently, seven paintings are showed to the participant. Two paintings for Model 1, two paintings for Model 2, one painting for Model 3 and finally two real paintings. Particular care was paid while choosing the latter since showing popular paintings might have led to a bias in the reliability of the answers. The chosen paintings can be found in the appendix. [A.4],[A.5],[A.6],[A.7].

For each painting, the three following questions are asked:

1. *Human or computer?* Multiple choice questions with one possible answer between "Human" or "Computer".

2. *How confident are you in your answer from 0 to 5?* Slider question on a scale from 0 to 5 with 0 meaning "Not confident at all" and 5 meaning "Very confident".

3. *Do you like this painting?* Slider question with star rating answers from 0 to 5 stars, with 0 stars meaning "No, not at all" and 5 stars meaning "I love it!".

After answering the questions about the paintings, the user is asked the following question: "Do you study/work in arts? If you don't work or study in this field, but you are really passionate and consider yourself to have a solid knowledge in the arts please click on the "Yes" box". This question was asked since an in depth knowledge of the arts could lead to biased results.

Results

A total of 408 responses were collected over the course of about two days. This high level of participation is already an indicator of how relevant the topic of art and AI is. In addition, many respondents were intrigued by the images displayed in the survey and asked for more information about it.

The results that will be presented in this section are obtained by averaging over the different images displayed for each model. The following example should help in better clarifying the procedure followed to obtain the results for each model. Respondent X was shown version 1 of the survey and the table presented in figure 4.13 illustrates her answers.

Respondent X answers			
Reference image	Human or computer?	How confident you are in your answer?	Do you like this painting?
Model 1 - Painting 1	Human	2	5
Model 1 - Painting 2	Computer	5	5
Model 2 - Painting 1	Computer	4	3
Model 2 - Painting 2	Computer	5	5
Model 3 - Painting 1	Computer	5	3
Real painting 1	Human	5	4
Real painting 2	Human	4	5

Figure 4.13: Survey sample response.

Take the answers regarding Model 1 as an example. Respondent X misclassified one of the two paintings, hence her final score in terms of correct answer for Model 1 will be 0.5. The same is done in terms of confidence in the given answer and likeability of the painting. Respondent X's averaged

answers over the 3 models and the real samples, obtained in the described way are illustrated in the next table.

Reference model	Correct classification	Confidence	Likeability
Model 1	0.5	3.5	5
Model 2	1	4.5	4
Model 3	1	5	3
Real	1	4.5	4.5

Table 4.2: Sample answers of respondent X averaged over the different images belonging to the same model.

The described procedure was applied for each respondent and then a final average over the 408 observations was taken.

The first result that emerges is that on average an art expert guesses correctly 4.2 out of 7 paintings, in contrast to non art experts that correctly classify 3.7 out of 7 paintings. Figure 4.14 illustrates clearly the tendency of art experts to answer correctly more than non-experts.

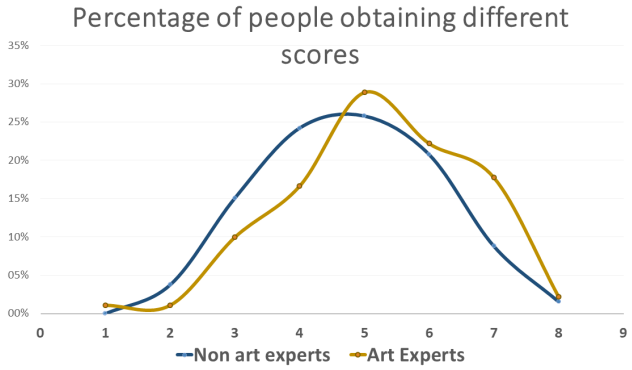


Figure 4.14: Comparison of the distributions of correctly classified images by non art experts and art experts.

When looking at the single models another trend becomes evident. The paintings that were more difficult to correctly classify are the ones produced by Model 1. Indeed, as shown in figure 4.15, paintings generated with Model

1 are wrongly attributed as painted by humans 54% of the time. With regards to the pictures produced by Model 2, one out of two are classified as being made by a human. Model 3 paintings are the ones that are most easily identified as computer-generated, with and average of 45 % attributions to a human.

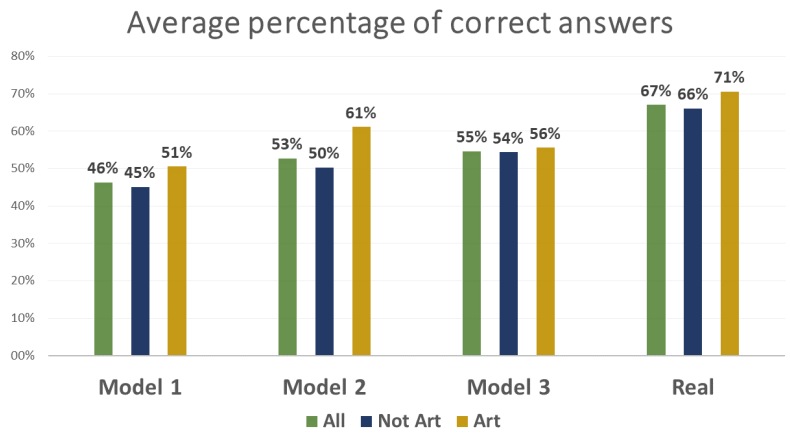


Figure 4.15: Percentage of correctly classified paintings by model.

Another interesting aspect emerges when looking at how confident the participants were in their classification. As illustrated by figure 4.16, although participants correctly classified the pictures generated by Model 3, they were not very convinced of their response, with an average confidence while answering of 3 out of 5. Additionally, it is clear that on average art experts tend to feel less sure about their classification.

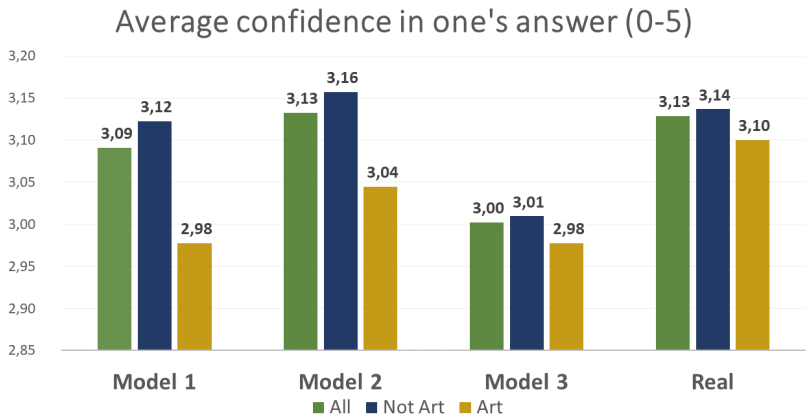


Figure 4.16: Average confidence expressed when answering the Human or Computer questions for each model. The confidence scale was set from 0 to 5 with 0 meaning not confident at all and 5 very sure.

The last point to consider is how much the participants like the paintings. This characteristic will be referred to as 'likeability' for the remaining part of the research. Looking at figure 4.17, the first important evidence emerges: the paintings that received the highest rating in terms of likeability are the ones produced by Model 1, with an average rating of 3.3 on a 0-5 scale. Following in order: Model 2 (average score of 3.19), the real paintings (average score of 3.01) and finally the paintings generated from the Model 3 (average score 2.40).

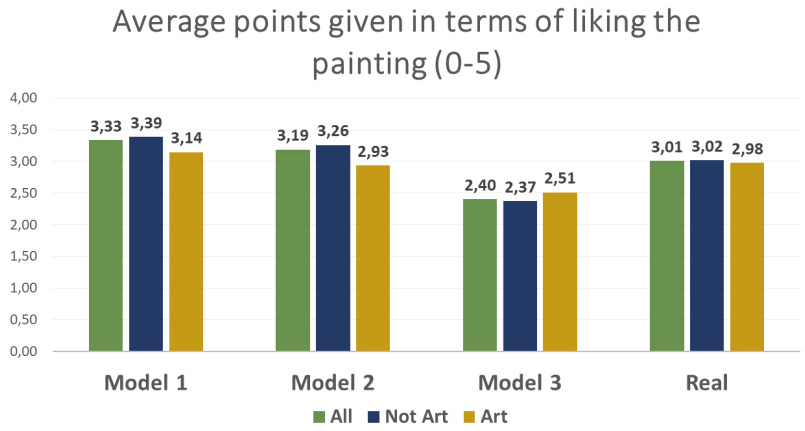


Figure 4.17: Average rating assigned to the paintings produces by each model and the real paintings.

Finally, the rating of the painting was cross-analysed with the attribution of the painting to a human or a computer. The idea behind this investigation is assess whether there is any kind of correlation between the attribution of the painting to either human or computer and the rating assigned to it. In this regard, an interesting point stands out: the attribution of the painting to a computer is correlated with lower rating scores. On the other hand, the correlation between human attribution and related rating is positive. These two evidences could be interpreted as a negative bias towards the general perception of computer-generated art. Nevertheless, it is not easy with the data at hand to disentangle the direction of this correlation, and a further analysis on this regard would definitely be very useful in the context of exploring AI art. In spite of this negative bias, it is worth noting once again that the paintings that have been appreciated the most by participants are the computer generated ones (excluding those produced by Model 3).

Concluding the experiment led in this research, generating realistic paintings with GANs is not only possible, but can lead to the production of artworks that are generally appreciated. According to the evidences emerged in this study, using state-of-the-art StyleGAN2-ADA model is the best way

to produce generative art, especially if trained from scratch. This definitely increases the computational and time resources needed, but allows the generation of paintings that not only are difficult to classify as computer generated, but are also liked more than the real paintings.

Conclusions and future work

The nature of creative processes is changing dramatically as a result of new technologies such as artificial intelligence. Computers are becoming increasingly important in creative fields including fine arts. This shift in how computers are typically viewed is the main driver behind this study, which aimed at investigating the usage of adversarial networks in the context of art generation.

Indeed, after providing a review on current related literature and a thorough description of the methods and models used in the empirical part of the research, an experiment designed to assess which is the best technique to generate realistic paintings was carried-out. The chosen dataset to develop the models comprised 5000 impressionistic landscape paintings. After testing a baseline GAN, three models have been developed and compared. Model 1 consisted of StyleGAN2-ADA trained from scratch. After around 4 days of training, the model obtained good results in terms of quality of the generated output. Model 2 presented as an extension of Model 1 as it was StyleGAN2-ADA trained with transfer learning, used both with fine-tuning and freezing 13 lower layers of the discriminator. The first version generated paintings with similar visual quality compared to the ones of Model 1. The second version produced blurry images with respect to the first version, thus because of its relatively less neat visual quality, it was discarded. Ultimately, Model 3 was developed. Firstly, sketches of the paintings were obtained through holistically-nested edge detection. Afterwards, new sketches were

generated with StyleGAN2-ADA and finally they were translated into paintings with pix2pixHD, mimicking the generally followed creative process of a human artist. In this experiment, however, StyleGAN2-ADA training exhibited some issues, including noticeable signs of mode collapse. Nonetheless, the paintings obtained with this procedure, presented an interesting aesthetic, that motivated the choice of further investigating them also in the final part of this experiment.

The evaluation of the competing models was carried on along two directions: a quantitative one with the calculation of the FID metric and a qualitative one through the distribution of a survey that acted as a Turing test. The latter consisted, in fact, in a scored quiz in which participants were shown various images of paintings (both real and generated by the developed models) and were asked to classify them as human or computer generated. Moreover, they were asked to express the level of confidence in their answer and to rate how much they liked the displayed painting. A total of 408 individual responses was collected. The main findings include:

- On average, participants were able to correctly guess 3.8 out of 7 times. This average increased when looking at the fraction of respondents who declared to be working/studying in the arts.

- Model 1's paintings are not only the ones that people wrongly label as being painted by humans the most, but they're also the ones that people like more than any other, beating even real paintings.

- Model 3 paintings instead, are the ones that people identify mostly as computer generated but, interestingly, they are the ones about which participants were most unsure about.

- Evidence has emerged that the attribution of the painting to a computer is correlated with lower rating scores. On the other hand, the correlation between human attribution and related rating resulted pos-

itive. These two facts could be interpreted as a negative bias towards the general perception of computer-generated art.

The experiment thus ended with the conclusion that the best method to generate realistic paintings is the one used in Model 1, i.e., training StyleGAN2-ADA from scratch. Nevertheless, it is worth keeping in mind the benefits in terms of time and computational resources required that a transfer learning approach can bring.

Regardless of the method to generate it, one thing is certain: AI art is a rapidly expanding field. The recent NFT market surge in the context of generative art [11], for instance, is just one of the many evidences of the potential of using AI for artistic purposes, and for this reason research should continue along several directions.

During the development of this project in fact, various hints emerged, upon which it would certainly be worthwhile conducting further in-depth studies. In the first place, a natural evolution of this research would be the deepening of generative art in a framework of text-to-image models, given the tremendous current developments in this field. Secondly, in light of the intriguing findings that emerged from the study of the responses to the survey, it begs the question of what new evidences might arise from a more targeted study regarding the perception and consequent reception of AI art. On a final note, one cannot avoid mentioning the general concept of creative autonomy articulated in the AI domain. The models developed in this study, as well as in most of the literature, should be considered more as artists' tools rather than as artists per se. However, considering the rapid pace of technological advancements in machine learning, it is plausible to imagine future scenarios in which the creative process is delegated in its entirety to machines. This in turn, will pave the way for a plethora of studies and research opportunities, not just in the field of data science, but also and notably in philosophical, artistic, and sociological contexts.

Appendix A

Appendix

Figure A.2: Sample output of StyleGAN2-ADA trained for around one day and a half on a dataset composed by Ukiyo-e style prints.

	Measure	Description
Quantitative	1. Average Log-likelihood [18, 22]	• Log likelihood of explaining realworld held out/test data using a density estimated from the generated data (*e.g.* using KDE or Parzen window estimation). $L = \frac{1}{N}\sum_i \log P_{model}(\mathbf{x}_i)$
	2. Coverage Metric [33]	• The probability mass of the true data "covered" by the model distribution $C := P_{data}(dP_{model} > t)$ with t such that $P_{model}(dP_{model} > t) = 0.95$
	3. Inception Score (IS) [3]	• KLD between conditional and marginal label distributions over generated data. $\exp\left(\mathbb{E}_{\mathbf{x}}\left[\mathbb{KL}\left(p\left(\mathrm{y} \mid \mathbf{x}\right) \parallel p\left(\mathrm{y}\right)\right)\right]\right)$
	4. Modified Inception Score (m-IS) [34]	• Encourages diversity within images sampled from a particular category. $\exp(\mathbb{E}_{\mathbf{x}_i}[\mathbb{E}_{\mathbf{x}_j}[(\mathbb{KL}(P(y\|\mathbf{x}_i)\|P(y\|\mathbf{x}_j))]])$
	5. Mode Score (MS) [35]	• Similar to IS but also takes into account the prior distribution of the labels over real data. $\exp\left(\mathbb{E}_{\mathbf{x}}\left[\mathbb{KL}\left(p\left(y \mid \mathbf{x}\right) \parallel p\left(y^{train}\right)\right)\right] - \mathbb{KL}\left(p\left(y\right) \parallel p\left(y^{train}\right)\right)\right)$
	6. AM Score [36]	• Takes into account the KLD between distributions of training labels vs. predicted labels, as well as the entropy of predictions. $\mathbb{KL}(p(y^{\text{train}}) \parallel p(y)) + \mathbb{E}_{\mathbf{x}}\left[H(y\|\mathbf{x})\right]$
	7. Fréchet Inception Distance (FID) [37]	• Wasserstein-2 distance between multi-variate Gaussians fitted to data embedded into a feature space $FID(r, g) = \|\mu_r - \mu_g\|_2^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}})$
	8. Maximum Mean Discrepancy (MMD) [38]	• Measures the dissimilarity between two probability distributions P_r and P_g using samples drawn independently from each distribution. $M_k(P_r, P_g) = \mathbb{E}_{\mathbf{x},\mathbf{x}'\sim P_r}[k(\mathbf{x},\mathbf{x}')] - 2\mathbb{E}_{\mathbf{x}\sim P_r,\boldsymbol{y}\sim P_g}[k(\mathbf{x},\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{y},\boldsymbol{y}'\sim P_g}[k(\boldsymbol{y},\boldsymbol{y}')]$
	9. The Wasserstein Critic [39]	• The critic (*e.g.* an NN) is trained to produce high values at real samples and low values at generated samples $\hat{W}(\mathbf{x}_{test}, \mathbf{x}_g) = \frac{1}{N}\sum_{i=1}^N \hat{f}(\mathbf{x}_{test}[i]) - \frac{1}{N}\sum_{i=1}^N \hat{f}(\mathbf{x}_g[i])$
	10. Birthday Paradox Test [27]	• Measures the support size of a discrete (continuous) distribution by counting the duplicates (near duplicates)
	11. Classifier Two Sample Test (C2ST) [40]	• Answers whether two samples are drawn from the same distribution (*e.g.* by training a binary classifier)
	12. Classification Performance [1, 15]	• An indirect technique for evaluating the quality of unsupervised representations (*e.g.* feature extraction; FCN score). See also the GAN Quality Index (GQI) [41].
	13. Boundary Distortion [42]	• Measures diversity of generated samples and covariate shift using classification methods.
	14. Number of Statistically-Different Bins (NDB) [43]	• Given two sets of samples from the same distribution, the number of samples that fall into a given bin should be the same up to sampling noise
	15. Image Retrieval Performance [44]	• Measures the distributions of distances to the nearest neighbors of some query images (*i.e.* diversity)
	16. Generative Adversarial Metric (GAM) [31]	• Compares two GANs by having them engaged in a battle against each other by swapping discriminators or generators. $p(\mathbf{x}\|y=1; M_1^*)/p(\mathbf{x}\|y=1; M_2^*) = \big(p(y=1\|\mathbf{x}; D_1)p(\mathbf{x}; G_2)\big)/\big(p(y=1\|\mathbf{x}; D_2)p(\mathbf{x}; G_1)\big)$
	17. Tournament Win Rate and Skill Rating [45]	• Implements a tournament in which a player is either a discriminator that attempts to distinguish between real and fake data or a generator that attempts to fool the discriminators into accepting fake data as real.
	18. Normalized Relative Discriminative Score (NRDS) [32]	• Compares n GANs based on the idea that if the generated samples are closer to real ones, more epochs would be needed to distinguish them from real samples.
	19. Adversarial Accuracy and Divergence [46]	• Adversarial Accuracy. Computes the classification accuracies achieved by the two classifiers, one trained on real data and another on generated data, on a labeled validation set to approximate $P_g(y\|\mathbf{x})$ and $P_r(y\|\mathbf{x})$. Adversarial Divergence: Computes $\mathbb{KL}(P_g(y\|\mathbf{x}), P_r(y\|\mathbf{x}))$
	20. Geometry Score [47]	• Compares geometrical properties of the underlying data manifold between real and generated data.
	21. Reconstruction Error [48]	• Measures the reconstruction error (*e.g.* L_2 norm) between a test image and its closest generated image by optimizing for z (*i.e.* $min_{\mathbf{z}}\|G(\mathbf{z}) - \mathbf{x}^{(test)}\|^2$)
	22. Image Quality Measures [49, 50, 51]	• Evaluates the quality of generated images using measures such as SSIM, PSNR, and sharpness difference
	23. Low-level Image Statistics [52, 53]	• Evaluates how similar low-level statistics of generated images are to those of natural scenes in terms of mean power spectrum, distribution of random filter responses, contrast distribution, etc.
	24. Precision, Recall and F_1 score [23]	• These measures are used to quantify the degree of overfitting in GANs, often over toy datasets.
Qualitative	1. Nearest Neighbors	• To detect overfitting, generated samples are shown next to their nearest neighbors in the training set
	2. Rapid Scene Categorization [18]	• In these experiments, participants are asked to distinguish generated samples from real images in a short presentation time (*e.g.* 100 ms); *i.e.* real v.s fake
	3. Preference Judgment [54, 55, 56, 57]	• Participants are asked to rank models in terms of the fidelity of their generated images (*e.g.* pairs, triples)
	4. Mode Drop and Collapse [58, 59]	• Over datasets with known modes (*e.g.* a GMM or a labeled dataset), modes are computed as by measuring the distances of generated data to mode centers
	5. Network Internals [1, 60, 61, 62, 63, 64]	• Regards exploring and illustrating the internal representation and dynamics of models (*e.g.* space continuity) as well as visualizing learned features

Figure A.1: A summary of common GAN metrics. (Figure from [7])

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 4096)              507904
_________________________________________________________________
reshape_1 (Reshape)          (None, 4, 4, 256)         0
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 8, 8, 256)         590080
_________________________________________________________________
batch_normalization_13 (Batc (None, 8, 8, 256)         1024
_________________________________________________________________
activation_1 (Activation)    (None, 8, 8, 256)         0
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 16, 16, 256)       0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 16, 16, 256)       590080
_________________________________________________________________
batch_normalization_14 (Batc (None, 16, 16, 256)       1024
_________________________________________________________________
activation_2 (Activation)    (None, 16, 16, 256)       0
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 32, 32, 256)       0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 32, 32, 256)       590080
_________________________________________________________________
batch_normalization_15 (Batc (None, 32, 32, 256)       1024
_________________________________________________________________
activation_3 (Activation)    (None, 32, 32, 256)       0
_________________________________________________________________
up_sampling2d_4 (UpSampling2 (None, 64, 64, 256)       0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 64, 64, 256)       590080
_________________________________________________________________
batch_normalization_16 (Batc (None, 64, 64, 256)       1024
_________________________________________________________________
activation_4 (Activation)    (None, 64, 64, 256)       0
_________________________________________________________________
up_sampling2d_5 (UpSampling2 (None, 128, 128, 256)     0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 128, 128, 256)     590080
_________________________________________________________________
batch_normalization_17 (Batc (None, 128, 128, 256)     1024
_________________________________________________________________
activation_5 (Activation)    (None, 128, 128, 256)     0
=================================================================
Total params: 3,463,424
Trainable params: 3,460,864
Non-trainable params: 2,560
_________________________________________________________________
```

Figure A.3: Summary of the architecture of the generator used as a baseline.

(a) (b)

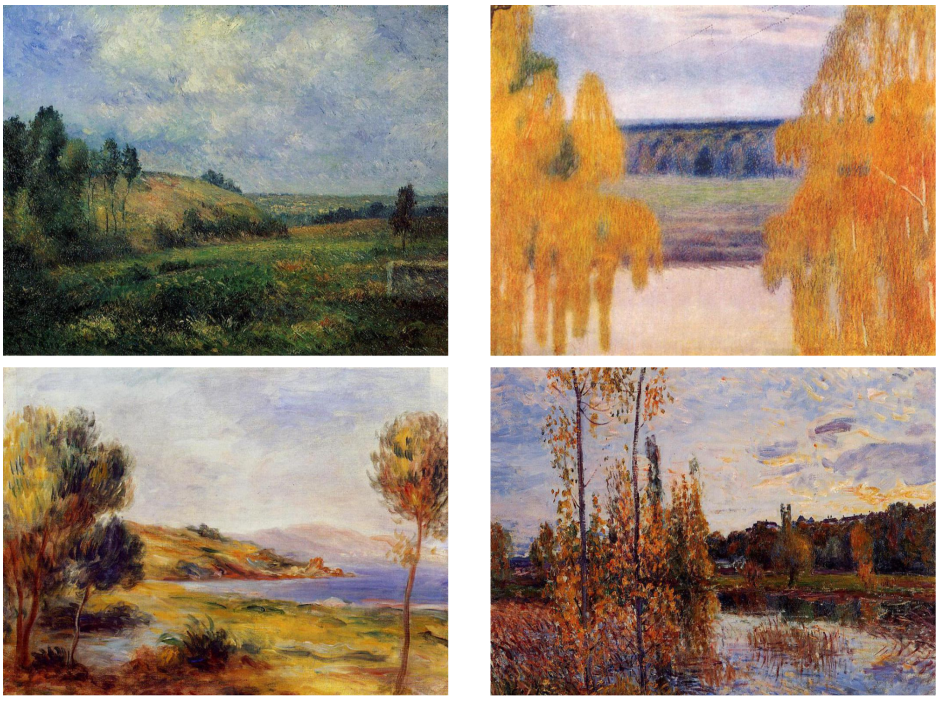
Figure A.4: Paintings produced by Model 1 (StyleGAN2-ADA trained from scratch) displayed in the survey. (a) Paintings showed in version 1 of the quiz. (b) Paintings showed in version 2 of the quiz.

(a) (b)

Figure A.5: Paintings produced by Model 2 (StyleGAN2-ADA trained with transfer learning) displayed in the survey. (a) Paintings showed in version 1 of the quiz. (b) Paintings showed in version 2 of the quiz.

(a) (b)

Figure A.6: Paintings produced by Model 3 (3 step GAN) displayed in the survey. (a) Painting showed in version 1 of the quiz. (b) Painting showed in version 2 of the quiz.



(a) (b)

Figure A.7: Real paintings displayed in the survey. (a)Paintings showed in version 1 of the quiz. Top image: Camille Pissarro "*Landscape near Pontoise, 1880*". Bottom image: Pierre Auguste Renoir "*The Bay*". (b)Paintings showed in version 1 of the quiz. Top image: Victor Borisov-Musatov "*Autumn Song, 1905*". Bottom image: Alfred Sisley "*L'Etang de Chevreuil, 1888*".

Bibliography

[1] Panos Achlioptas, Maks Ovsjanikov, Kilichbek Haydarov, Mohamed Elhoseiny, and Leonidas J Guibas. Artemis: Affective language for visual art. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11569–11579, 2021.

[2] Siddharth Agarwal, Harish Karnick, Nirmal Pant, and Urvesh Patel. Genre and style based painting classification. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 588–594. IEEE, 2015.

[3] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[5] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*, 2017.

[6] Carlotta Bonanni. Ai art. `https://github.com/bubibon/AI_ART`, 2022.

[7] Ali Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[8] Ali Borji. Pros and cons of gan evaluation measures: New developments. *Computer Vision and Image Understanding*, 215:103329, 2022.

[9] Stevo Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, 44(3), 2020.

[10] Stevo Bozinovski and Ante Fulgosi. The influence of pattern similarity and transfer of learning upon training of a base perceptron b2.(original in croatian: Utjecaj slicnosti likova i transfera ucenja na obucavanje baznog perceptrona b2). In *Proc. Symp. Informatica*, pages 3–121, 1976.

[11] Henrique Centieiro. The power of generative art in the nft space, 2021.

[12] Eva Cetinic and James She. Understanding and creating art with ai: Review and outlook. *arXiv preprint arXiv:2102.09109*, 2021.

[13] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*, 2016.

[14] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1511–1520, 2017.

[15] Jun Cheng, Fuxiang Wu, Yanling Tian, Lei Wang, and Dapeng Tao. Rifegan: Rich feature generation for text-to-image synthesis from prior knowledge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10911–10920, 2020.

[16] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[17] Elliot J Crowley and Andrew Zisserman. The state of the art: Object retrieval in paintings using discriminative regions. 2014.

[18] Yingying Deng, Fan Tang, Weiming Dong, Chongyang Ma, Feiyue Huang, Oliver Deussen, and Changsheng Xu. Exploring the representativity of art paintings. *IEEE Transactions on Multimedia*, 23:2794–2805, 2020.

[19] Steve DiPaola and Liane Gabora. Incorporating characteristics of human creativity into an evolutionary art algorithm. *Genetic Programming and Evolvable Machines*, 10(2):97–110, 2009.

[20] Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. *arXiv preprint arXiv:1706.07068*, 2017.

[21] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883, 2021.

[22] Stanislav Frolov, Tobias Hinz, Federico Raue, Jörn Hees, and Andreas Dengel. Adversarial text-to-image synthesis: A review. *Neural Networks*, 144:187–209, 2021.

[23] Stanislav Frolov, Shailza Jolly, Jörn Hees, and Andreas Dengel. Leveraging visual question answering to improve text-to-image synthesis. In *Proceedings of the Second Workshop on Beyond Vision and LANguage: inTEgrating Real-world kNowledge (LANTERN)*, pages 17–22, 2020.

[24] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. In *2017 International Conference on 3D Vision (3DV)*, pages 402–411. IEEE, 2017.

[25] Harsha Gangadharbatla. The role of ai attribution knowledge in the evaluation of artwork. *Empirical Studies of the Arts*, page 0276237421994697, 2021.

[26] Noa Garcia and George Vogiatzis. How to read paintings: semantic art understanding with multi-modal retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[27] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[28] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[29] Robert A. Gonsalves. Ganshare: Creating and curating art with ai for fun and profit, 2021.

[30] Robert A. Gonsalves. Impressionist-landscapes-paintings dataset, version 1, 2021.

[31] Nicolas Gonthier, Yann Gousseau, Said Ladjal, and Olivier Bonfait. Weakly supervised object detection in artworks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[32] I Goodfellow. Nips 2016 tutorial: Generative adversarial networks. arxiv 2016. *arXiv preprint arXiv:1701.00160*, 2016.

[33] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[35] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[36] Tobias Hinz, Stefan Heinrich, and Stefan Wermter. Semantic object accuracy for generative text-to-image synthesis. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[37] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[38] Drew A Hudson and C Lawrence Zitnick. Generative adversarial transformers. *arXiv preprint arXiv:2103.01209*, 2021.

[39] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[40] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two pure transformers can make one strong gan, and that can scale up. *Advances in Neural Information Processing Systems*, 34, 2021.

[41] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[42] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[43] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020.

[44] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Stylegan2-ada. `https://github.com/NVlabs/stylegan2-ada`, 2021.

[45] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[46] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[47] Daniel Keren. Painter identification using local features and naive bayes. In *Object recognition supported by user interaction for service robots*, volume 2, pages 474–477. IEEE, 2002.

[48] Diana Kim, Jason Xu, Ahmed Elgammal, and Marian Mazzone. Computational analysis of content in fine art paintings. In *ICCC*, pages 33–40, 2019.

[49] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[50] Sabine Lang and Björn Ommer. Attesting similarity: Supporting the organization and study of art image collections with computer vision. *Digital Scholarship in the Humanities*, 33(4):845–856, 2018.

[51] Bingchen Liu, Kunpeng Song, Yizhe Zhu, and Ahmed Elgammal. Sketch-to-art: Synthesizing stylized art images from sketches. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[52] Yifan Liu, Zengchang Qin, Tao Wan, and Zhenbo Luo. Auto-painter: Cartoon image generation from sketch by using conditional wasserstein generative adversarial networks. *Neurocomputing*, 311:78–87, 2018.

[53] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–549, 2004.

[54] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[55] Federico Milani and Piero Fraternali. A dataset and a convolutional model for iconography classification in paintings. *Journal on Computing and Cultural Heritage (JOCCH)*, 14(4):1–18, 2021.

[56] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[57] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Freeze the discriminator: a simple baseline for fine-tuning gans. *arXiv preprint arXiv:2002.10964*, 2020.

[58] Saif Mohammad and Svetlana Kiritchenko. Wikiart emotions: An annotated dataset of emotions evoked by art. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*, 2018.

[59] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015.

[60] Frank Nielsen. A family of statistical symmetric divergences based on jensen's inequality. *arXiv preprint arXiv:1009.4004*, 2010.

[61] Atsuhiro Noguchi and Tatsuya Harada. Image generation from small datasets via batch statistics adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2750–2758, 2019.

[62] Eric J Nunn, Pejman Khadivi, and Shadrokh Samavi. Compound frechet inception distance for quality assessment of gan created images. *arXiv preprint arXiv:2106.08575*, 2021.

[63] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[64] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[65] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021.

[66] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[67] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[68] Fathy Rashad. How i built an ai text-to-art generator, 2021.

[69] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International conference on machine learning*, pages 1060–1069. PMLR, 2016.

[70] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. *arXiv preprint arXiv:1705.09367*, 2017.

[71] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242, 2016.

[72] Catherine Sandoval, Elena Pirogova, and Margaret Lech. Two-stage deep learning approach to the classification of fine-art paintings. *IEEE Access*, 7:41770–41781, 2019.

[73] Dipanjan Sarkar. A comprehensive hands-on guide to transfer learning with real-world applications in deep learning, 2018.

[74] Derrick Schutlz. Stylegan2-ada. `https://github.com/dvschultz/stylegan2-ada`, 2021.

[75] Aaron Hertzmann Sergey Karayev, Holger Winnemoeller, Aseem Agarwala, and Trevor Darrell. Recognizing image style. *CoRR*, abs/1311.3715, 2013.

[76] Lior Shamir, Tomasz Macura, Nikita Orlov, D Mark Eckley, and Ilya G Goldberg. Impressionism, expressionism, surrealism: Automated recognition of painters and schools of art. *ACM Transactions on Applied Perception (TAP)*, 7(2):1–17, 2010.

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

[78] Matteo Stefanini, Marcella Cornia, Lorenzo Baraldi, Massimiliano Corsini, and Rita Cucchiara. Artpedia: A new visual-semantic dataset with visual and contextual sentences in the artistic domain. In *International Conference on Image Analysis and Processing*, pages 729–740. Springer, 2019.

[79] Tristan Sylvain, Pengchuan Zhang, Yoshua Bengio, R Devon Hjelm, and Shikhar Sharma. Object-centric image generation from layouts. *arXiv preprint arXiv:2003.07449*, 1(2):4, 2020.

[80] Ilya Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. *arXiv preprint arXiv:1701.02386*, 2017.

[81] Nanne Van Noord and Eric Postma. Learning scale-variant and scale-invariant features for deep image classification. *Pattern Recognition*, 61:583–592, 2017.

[82] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[83] James Vincent. A look back at the first computer art contests from the'60s: Bulletricochetsandsinecurveportraits, 2015.

[84] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[85] Yaxing Wang, Abel Gonzalez-Garcia, David Berga, Luis Herranz, Fahad Shahbaz Khan, and Joost van de Weijer. Minegan: effective knowledge transfer from gans to target domains with few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9332–9341, 2020.

[86] Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring gans: generating images from limited data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 218–234, 2018.

[87] Zhengwei Wang, Qi She, and Tomas E Ward. Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

[88] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[89] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.

[90] Wen Xu, Jing He, and Yanfeng Shu. Transfer learning and deep domain adaptation. In *Advances and Applications in Deep Learning*. IntechOpen, 2020.

[91] Alice Xue. End-to-end chinese landscape painting creation using generative adversarial networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3863–3871, 2021.

[92] Heekyung Yang and Kyungha Min. Classification of basic artistic media based on a deep convolutional approach. *The Visual Computer*, 36(3):559–578, 2020.

[93] Lin Zhao, Meimei Shang, Fei Gao, Rongsheng Li, Fei Huang, and Jun Yu. Representation learning of image composition for aesthetic prediction. *Computer Vision and Image Understanding*, 199:103024, 2020.

[94] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.