Imperial College London

Masters Thesis

Imperial College London

Dyson School of Design Engineering

MichelGANgelo: A Deep Learning Framework to Create Art, Establish Links between Movements in History and Predict & Generate *Future* Art.

Author:
Alexandre Berkovic

Supervisor:
Pietro Ferraro

Second Marker:
George Chandramohan

A thesis presented to Imperial College London in partial fulfilment of the requirements for the degree of MEng. of Design Engineering and the Diploma of Imperial College London

June 9, 2022

Abstract

This paper aims at bridging the gap between machines and mankind's unique ability to be creative; reconciling the supposed discrepancy between imagination and computation. In an effort to do so, we conceive a novel deep generative framework that we name MichelGANgelo [1]. Its goal is to generate art of a specific style or subject matter (or both at once) which it does by using an ACWGAN-GP (Auxiliary-Classified Wasserstein Generative Adversarial Network with Gradient Penalty). The generated works are enlarged with a SRGAN (Super-Resolution GAN) pretrained on high resolution images of textures to enhance visual quality. Moreover, the framework also aims to better understand the links between art movements in history which it does through a novel dual-phase CNN (Convolutional Neural Network) architecture exploiting pretrained ResNet models. Finally, the system attempts to produce *future* art by encoding the data into the latent space and using linear regression to forecast the next movement in a given sequence; a method based on geodesic interpolation with Optimal Transport is also suggested. Apart from the quantitative validation of the proposed CNN, we argue that art is intrinsically subjective which led us to conduct a set of human-centric experiments. Those show that subjects find it highly difficult to differentiate generated art from that of real artists and also that certain works produced by our model obtain subjective appreciation scores that outperform that of real paintings.

Keywords: GAN · VAE · OT · CNN · Art

[1] The repository for this project can be found here: `https://github.com/alexandreberkovic/MichelGANgelo`

Acknowledgements

"This thesis was supported by the Dyson School of Design Engineering at Imperial College London. I would like to thank my supervisor Pietro Ferraro who provided insight and expertise that greatly assisted the research.

I am also immensely grateful to Robert Shorten, Sam Cooper, Steven Kench, Isaac Squires from Imperial College London as well as Sarah Boufelja from IBM Research for their invaluable comments and contribution to the research.

I would particularly like to thank our very dear René Coty who inspired and guided me throughout this project."

Table of Contents

List of Figures

List of Tables

Nomenclature

α	Learning rate
β_1	Exponential decay rate for the first moment estimates in Adam
β_2	Exponential decay rate for the second moment estimates in Adam
Δ	Distance or dissimilarity measure
λ	Gradient penalty term
$\mathcal{C}$	Critic network of the ACGAN
$\mathcal{D}$	Set of 1-Lipschitz functions
$\mathcal{F}_C^{(k)}(\mathbf{c})$	Image probability distribution in latent space
$\mathcal{F}_X^{(k)}(x)$	Image distribution in data space
$\mathcal{L}$	Loss function
$\mathcal{L}^{(G)}$	Generator loss function
$\mathcal{T}_c(\mu, \nu)$	Transfer function with cost c
$\mathcal{V}$	Set of training vectors for the AutoEncoder
$\mathcal{X}$	Transport plan in OT-GAN or data manifold of the Encoder
$\mathcal{Y}$	Transport plan in OT-GAN
μ	Arbitrary data distribution in a Lagrangian framework
ν	Arbitrary data distribution in a Lagrangian framework
Ω	Latent manifold
ω	LeakyReLU parameter
π	Causal transport plan
$\prod(\mu, \nu)$	Set of transport plans between μ and ν
σ	Standard deviation or Sigmoid non-linearity
$\mathbf{A}$	Input image for CNN
$\mathbf{c}$	Latent codes
$\mathbf{l}$	Latent encoding vector
$\mathbf{W}_{ok}^{(m)}$	Filter parameterization matrix for CNN
$\mathbf{x}$	Network input vector
$\mathbf{z}$	Noise vector

θ	Network parameters
θ_g	Generator parameters
B	Base space in FAE
c	Arbitrary *conditioning* variable or OT cost
c_k	Label associated with a specific class k
c_x	True label associated with an image $\mathbf{x}$
d_{W_p}	Wasserstein distance
E	Error or energy function
F	Fiber space in FAE
f	Arbitrary function
f_θ	Encoding map of the AutoEncoder
G	Generator network
g	Arbitrary function
g_ξ	Decoding map of the AutoEncoder
H	AutoEncoder hidden layer size
$H(t,s)$	Categorical Cross-Entropy
I^{HR}	High-resolution image input
I^{LR}	Low-resolution image input
I^{SR}	Super-resolution image output
l^{SR}	Perceptual super-resolution loss
N	Number of images in dataset or AE input/output layer size
$N(\mu,\sigma)$	Gaussian distribution
n_c	Number of critic iterations
p_1	Arbitrary probability distribution n°1
p_2	Arbitrary probability distribution n°2
p_g	Model probability distribution
$p_z(\mathbf{z})$	Prior on input noise vector
p_{data}	Actual data probability distribution
T	Transport map
$V(D,G)$	Two-player game minimisation formulation of a GAN
ACGAN	Auxiliary Classified GAN

AE	AutoEncode
AI	Artificial Intelligence
CNN	Convolutional Neural Network
ConvLSTM	Convolutional LSTM
COT	Computational Optimal Transport
D	Discriminator network
DCGAN	Deep Convolutional GAN
DCNN	Deep Convolutional Neural Network
EMD	Earth-Mover Distance
FAE	Fibered AutoEncoder
FID	Frechet Inception Distance
GAN	Generative Adversarial Network
GP	Gradient penalty
IS	Inception Score
JS	Jensen-Shannon
K	Number of art movements in the dataset
KL	Kullback-Leibler
LSTM	Long Short-Term Memory network
M	Learned latent space of the FAE
MGP	Multi disentangled-features Gaussian Processes
MLP	Multi-Layer Perceptron
OMT	Optimal Mass Transport
OT	Optimal Transport
PCA	Principle Component Analysis
SGD	Stochastic Gradient Descent
t-SNE	t-distributed Stochastic Neighbour Embedding
VAE	Variational AutoEncoder
WAE	Wasserstein AutoEncoder
WC	Weight clipping
WGAN	Wasserstein GAN
WVAE	Wasserstein Variational AutoEncoder

1 Introduction

1.1 Motivations

Art is a realm of creativity and innovation, highlighting not only the artist's mind-set but also creating a snapshot of his time. Its development is intrinsically linked to social, economic, and political trends which makes it a phenomenal time capsule. Artificial intelligence on the contrary could be defined as a simulation of human intelligence, designed by complex computational processes. Although the advancements of machine learning have been drastic in the past 10-15 years, its use is still very much confined to specific tasks. Better understanding the *creative* capabilities of a seemingly impersonal system is at the centre of our line of work.

1.2 AI applied to art

Recently, certain research papers have shown an interest in investigating the possible applications of machine learning for tasks such as art classification or periodisation as shown by the exhaustive review in [7]. Moreover, when looking into better understanding the periodisation of art in history, [8] leverages dimensional-reduction techniques such as PCA (Principle-Component Analysis) to visualize and understand key links and artists between movements in time. Finally, fewer papers look into actually generating *art*. Relevant works consist in the generation of art from existing datasets [9] or in the *creative* generation of art by diverging from classical deep generative frameworks [10].

1.3 Gap in research

There are three main contributions of this paper to the existing research. Firstly, a deep generative framework is devised in order to produce art. Secondly, a state-of-the-art image classifier to help categorise art based on style is developed to validate the generated outputs. Thirdly, a novel methodology to interpolate, predict and generate *future* art is proposed. Moreover, this paper also has two subsidiary contributions: a novel methodology for training a super resolution framework to increase the quality of generated works as well as a holistic investigation on how humans interact with AI-generated art compared to human-created one.

1.4 Paper outline

This paper is devised as follows: Section 2 sheds some light on existing research surrounding deep learning applied to fine-arts in order to better understand potential gaps in research. Then, Section 3 lists in more details the contributions this paper aims to provide. Afterwards, Section 4 brings background information to the frameworks and techniques that are used throughout the paper. Subsequently, Section 5 develops the methods conceived for the purpose of the research proposal. As a result, Section 6 presents the results of the research followed by a discussion. Finally, Section 7 serves as a way to wrap up this paper and present future work.

2 Literature review

2.1 Convolutional Neural Networks

One of the aims of this project is to build a fine-art classifier based on style in order to validate the generated outputs from the MichelGANgelo framework. Hence, researching the existing literature surrounding CNNs and art is deemed necessary.

2.1.1 Leveraging pre-trained networks

Fine-art classification with CNNS is a relatively recurrent theme in the field of computer vision. To increase performance from architectures built from scratch, [11] uses a pre-trained CNN to classify paintings in different categories, namely style, artist and genre. The author does highlight that the task of style recognition is the hardest as it is not necessarily correlated with subject matter. Indeed, style is a hard concept to define in the context of discrete mutually exclusive classes as transitions between two movements are rarely strictly defined. To this extent, certain paintings may have elements of multiple styles which increases complexity when trying to identify a single class in which to classify them. In this paper, the authors decide to use a pre-trained CaffeNet implementation in which the last layer is modified to cater for the number of classes in the dataset. After retraining it, they achieve a style accuracy of 56.43% for 15 classes. Similarly, other research papers obtained equivalent results by using other pre-trained models such as AlexNet architecture [12] to obtain a 54.50% accuracy for 27 styles. Interestingly, [13] uses a Residual Neural Network pre-trained on ImageNet [2] and retrains the 20 last layers instead of only the last fully connected one and achieves an accuracy of 62% on 25 classes. To the best of our knowledge, this is the best performance at the time our work has been written.

2.1.2 Modifying and fine-tuning pre-trained networks

[8] seems to be the only research paper which not only retrains a pre-trained architecture but also modifies it. Indeed, the authors use a pre-trained VGG16 model and add two fully connected layers at the end (with 1024 and 512 nodes) to reduce the number of nodes, forcing the representation to have less features. The aim of this is to enable paintings across styles to be closer together depending on their similarity. This method boosts accuracy to a state-of-the-art value of 62.8%

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

with 20 classes. It must be noted that all research papers mentioned above leverage the open source `Wikiart` dataset [3] which makes their results comparable.

2.2 Deep Generative Networks

The main objective of this paper is to formulate a deep generative framework to produce art based on GANs. Hence, the literature is analysed to better understand which frameworks to leverage and whether anyone has attempted something similar.

2.2.1 Enhancing the potential of GANs with OT

First introduced by [14], Wasserstein GANs are a novel framework based on the Wasserstein-1 or Earth-Mover distance whose importance will be reviewed later. In this methodology, the EMD is minimized instead of the Jensen-Shannon (JS) divergence as the former is continuous and differentiable everywhere, allowing the Discriminator to be trained to an optimal state. The WGAN function is constructed using the Kantorovich-Rubinstein duality as shown below:

$$\min_{G} \max_{D \in \mathcal{D}} \mathbb{E}_{\mathbf{x} \sim p_data}[D(\mathbf{x})] - \mathbb{E}_{\tilde{\mathbf{x}} \sim p_g}[D(\tilde{\mathbf{x}})] \tag{1}$$

where $\mathcal{D}$ is considered as the set of 1-Lipschitz functions and p_g is the model distribution defined by $\tilde{\mathbf{x}} = G(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z})$.

Similarly, [15] generalizes WGANs by enabling the framework to *explicitly* use any transportation cost function while the model initially formalized by Arjovsky *implicitly* uses the Euclidean Wasserstein-1 distance. This enables the author to use cost functions leveraging Wasserstein-2 distance which stabilises and speeds up gradient descent.

A limitation of [14] is the use of weight clipping to enforce a Lipschitz constraint, forcing the function to have a maximum gradient. This makes the model very sensitive to the choice of clipping value and can easily lead to either very long training times or the common vanishing gradient problem. Since this formulation, [5] has introduced an Improved WGAN model that uses an alternative to the clipping method. Here, the authors penalize the gradient norm of the critic's output with respect to its input which proves to stabilise training while reducing hyper-parameter tuning. The Gradient Penalty objective formalised is as follows:

[3] `https://www.wikiart.org`, `https://github.com/cs-chan/ArtGAN/tree/master/WikiArt%20Dataset`

$$\mathcal{L} = \underbrace{\mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}[D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x}\sim p_d ata}[D(\mathbf{x})]}_{\text{Original Critic Loss}} + \underbrace{\lambda \mathbb{E}_{\hat{\mathbf{x}}\sim p_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2]}_{\text{IWGAN Gradient Penalty}} \tag{2}$$

In the domain of GANs, researchers have looked at applying Optimal Transport properties in order to create better outputs, as shown with WGANs. [16] uses OMT (Optimal Mass Transportation) to optimise the Discriminator of the framework by implementing Kantarovich potential to calculate an optimal transportation map with the Generator. This developed method renders better results than WGANs in regards of convex geometries, approximating probability distributions for numerous clusters better in low-dimensional space. Moreover, even though GANs are able to output very sharp and precise images, they do not properly estimate areas of high data density which often results in a distorted output or low modal diversity. This is due to a poor estimation of the distance metric when training the model and may be mitigated by using regularization terms. [17] proposes the Parallel Optimal Transport GAN which is "an unsupervised approach that regularizes the Generator using the Wasserstein distance in low dimensional latent representation of the data distribution and model distribution". Such framework provided meaningful improvements in terms of FID and Inception scores, demonstrating a quantitative amelioration.

2.2.2 Creativity in GANs

The question of stylized representations and artistic creation with machine learning has been recurrent within the research field. Numerous papers have addressed concepts such as style transfer [18], sketch to image modeling [19] or even painting generation by conditional brushstroke input [20]. However, these outputs cannot be coined as creative as they modify an already existing input; there is no creative generation, only a creative modification. The only paper that tackles the aforementioned problem is [10]. It proposes a novel system named Creative Adversarial Networks which generates art by increasing what is defined as *arousal potential*. The author's point in conceiving this algorithm is that GANs are limited in their creative process to what they perceive in their input dataset; this model tries to maximise deviation from existing styles while minimising deviation from art distribution. Although the present work has better performances than its baseline DCGAN, its output is very subjective and is complicated to effectively assess as such.

Another relevant research named ArtGAN [9] develops a network which increases the quality of generated images by back-propagating the gradient of the loss function with respect to the label

from the categorical Discriminator to the Generator. It uses an AutoEncoder in the Discriminator which includes additional and complimentary information to the model, allowing it to generate more diverse images as it will produce very different gradient directions within the batch.

Finally, [21] looks at leveraging cGANs and AutoEncoders in order to model and forecast art movements. However, once again, this research mostly develops future unseen art (which does not have a baseline) deprived from any genre conditioning which makes it complicated to assess. Nevertheless, despite this subjectivity, this paper still presents results that are related to the line of research that we intend to pursue, and therefore serves as inspiration for this work.

2.2.3 An implementation of two concurring frameworks

Another common framework to build deep generative networks is Variational AutoEncoders (VAEs). However, the literature concerning the dual implementation of VAEs and GANs is very limited as few researchers have achieved such architectures. [22] is the first paper to mention such a framework and shows that AutoEncoders are able to leverage the representations they learn through training to better estimate comparable features in the data space. As stated, learned feature representations in the Discriminator can be used as grounds for the VAE reconstruction objective. To do so, a VAE and a GAN architecture are combined to create an unsupervised generative model that can encode, generate and compare datasets.

[1] develops a framework that embeds the image data manifold into the latent space with an AE and then creates a semi-discrete Optimal Transport map between the uniform distribution and the latent distribution in order to generate new latent codes. The architecture is more easily understood by visualizing the diagram taken from their paper in Fig. 1. This architecture is relevant to the project as the predictive task requires us to use sampled latent codes from the latent space in order to train the generative (GAN) model.

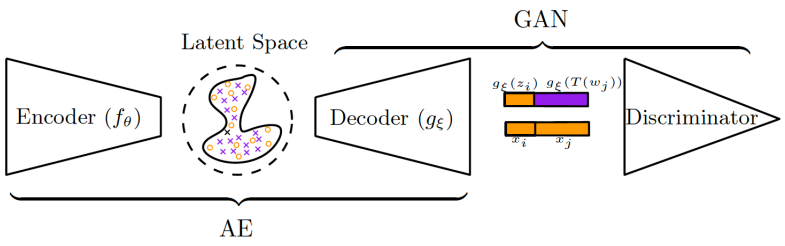


Fig. 1: AE-OT-GAN architecture [1].

2.3 Manipulating the latent space

In this paper, we also aim at developing a methodology to generate *future* art. Formulating such a predictive model requires to exploit the latent space which is why reviewing relevant literature about it is essential.

2.3.1 A latent space embedding methodology

An essential aspect of this project is being able to compress our image dataset into the latent space, for it to be efficiently used in the rest of the pipeline. If AutoEncoders have been used to encode data into lower dimensional latent spaces [23], the output often lacks in structure which makes it complicated to sample from the generated data distribution [24]. VAEs prove to be extremely relevant in the context of density modeling as they effectively introduce a regularization on the latent space coupled with an adapted training procedure. They are also able to decode the latent space to generate data; mapping the low-dimensional latent space to the observation space (defined as *data manifold*). This interpolation method has been utilized for image generation and certain frameworks even allow for the generation of annotated data by implementing a condition [25]. Hence, determining an optimal space regularization for latent space generation is of the utmost importance; methods such as Geodesic Latent Space Regularization [26] achieve great preliminary results. Interestingly enough, a few researchers managed to implement Optimal Transport cost in VAEs by incorporating Wasserstein distance in AutoEncoders (WAEs) which have shown great results in terms of generative modeling of data distribution [27] [28]. VAEs have also proved useful in Deep Clustering [29], allowing to create clustering of datasets within the latent space, which may show useful further down the line when trying to understand the way art movements are correlated. Finally, using VAEs also enables the generation of in-between images through learned latent space representation [30] which may be used in order to further understand the trends from a movement to another.

2.3.2 Travelling the latent space as a disentangled area

OT's applications have extended due to its capability of generating a map between two distributions in the latent space; circumventing the issues arising from posterior collapse. [31] implements AE-OT, an equivalent to WAE except that the Wasserstein Distance is defined in the latent space and not in the original image space. The developed architecture trains its Discriminator in a mini-

mization problem by using Brenier's theorem [32] which expresses the transport map by exploiting the gradient of the optimal Discriminator. By preserving the data manifold's structure in the latent space, AE-OT is able to obtain better multi-cluster distributions than VAE and WAE as demonstrated by dataset reconstruction results on `CelebA` or `MNIST`. However, as highlighted by [33], a common issue with AE-OT related algorithms is that they tend to perform better than conventional models in two dimensional datasets with little features but often struggle when increasing image complexity. [34] proposes a novel way to estimate the *distance* between datasets by using the Optimal Transport distances defined by Villani [35] to compare the obtained distributions. The results show a scalable and flexible way in which to compare datasets but does not take into account loss function and predictor function, whose use may give even better results.

The research in [2] aims at creating a geometric framework to build paths between samples of different conditions which they call FAE (Fibered AutoEncoder). The authors define the latent space as "a fiber bundle stratified into a *base space* encoding conditions and a *fiber space* encoding the variables with conditions". The correspondences between each *fiber* of the latent space is computed by determining their geodesic (shortest path) which is obtained by minimizing an energy functional that results in a diffeomorphic flow between them. As a point of reference, a diffeomorphic function is invertible and allows to map a differentiable manifold to another such that both the function and its inverse are differentiable. This might be of interest when trying to better understand the links between each art movement in the latent space as we will be able to determine the *flow* functions between each conditional fiber. A diagram of the transport mechanism of the method is presented in Fig. 2 where the learned latent space M is stratified into a base space B and a fiber space F.

2.3.3 Using information from the latent space to develop predictive models

Some research looks into how the latent space can be used to generate data predictively. For instance, [36] develops a GAN framework to generate sequential data by using Causal Optimal Transport. In doing so, the authors use classic OT theory alongside additional temporal causality constraints that allows them to parameterize the cost function learnt by the Discriminator as a *worst case* distance and learn time-dependant data distributions. The causal condition of the developed framework is defined for d-dimensional data (number of channels), K-long sequences so that distributions μ and ν are on the space $R^{d\times K}$ where transport plans $\mathcal{X} = R^{d\times K}$ and $\mathcal{Y} = R^{d\times K}$ make it possible to define the objective function. On $\mathcal{X} \times \mathcal{Y}$, we consider $x = (x_1, \cdots, x_K)$ and

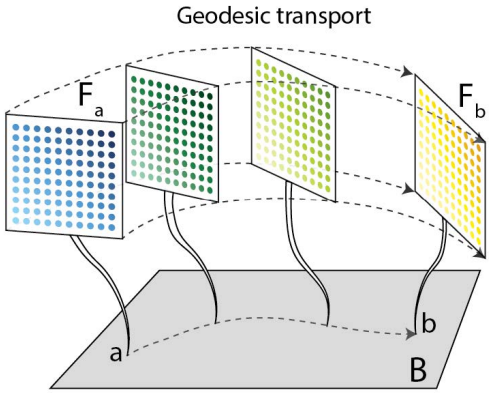


Fig. 2: Conceptual sketch of geodesic transport in the FAE methodology [2].

$y = (y_1, \cdots, y_K)$ the first and second half of the coordinates respectively. Hence, a transport plan $\pi \in \prod(\mu, \nu)$ is causal when defined by the following:

$$\pi(dy_k|dx_1, \cdots, dx_K) = \pi(dy_k|dx_1, \cdots, dx_k) \qquad \text{for all} \quad k \in \{1, \cdots, K-1\} \tag{3}$$

Other methods such as [37] leverage more conventional methods such as LSTMs. Here, a deep AutoEncoder is trained with a GAN to compress the data into the latent space and then exploit the sequential and temporal correlation in the data with a ConvLSTM to predict the latent vector representation of the future *frame*. Similarly, [4] introduces a MGP-VAE model (Multi-disentangled-features Gaussian Processes Variational AutoEncoder) which models the latent space for unsupervised learning of disentangled representations. It uses complex models that leverage fractional Brownian motions and Brownian bridges to infer on inter-image correlation as well as a geodesic loss function that accounts for the curvature of the data manifold (similar to FAEs). Although seemingly applicable to our problem, both of these methods are applied to video sequences which implies that their inputs are video frames. However, considering the nature of our dataset (art images), it might not be feasible to produce such a sequential dataset. A possible way to do so would be to find the closest images in all movements $k + n$ (with n the number of movements after k) to a *seed* image and use this sequence as our dataset.

3 Contributions

Existing research shows limitations in terms of generative frameworks when simply using VAEs or GANs. However, using OT in either models has shown significant improvements, especially since the formulation of the WGAN framework. Although frameworks combining OT with GANs or VAEs have been developed, few papers actually mention models that incorporate all three together. Hence, the main aim of this paper is to create a novel deep generative framework implementing OT, VAEs and GANs to better understand the links which exist between art movements and consequently generate art. An ambitious supplementary aim of this paper is to exploit the generative framework and its latent space in order to produce *future* art and see how it compares to existing works. Ideally, the goal is that given the data prior to a given movement (Cubism for instance), the predictive network is able to generate a painting which is closer to that movement (or any movement after it) than anything it has been trained on. As a result, this paper aims to:

- Better understand the key *features* that contribute to art movements in time by developing a state-of-the-art classifier. The classifier will build upon pretrained CNN frameworks by incorporating them in a novel architecture composed of an initial binary classifier followed by two movement-specific classifiers.
- Determine whether a machine learning algorithm can generate art by conditioning it on a specific class such as style, subject matter or both. The framework builds upon conditional GANs and the implementation of Optimal Transport in the loss function (specifically the Wasserstein distance) as a means of increased model stability and convergence.
- Implement a Super-Resolution architecture in order to increase the resolution and the quality of the generated works. The developed model is based off an existing architecture but implements a novel pre-training methodology in order to recreate more realistic textures.
- Understand whether artificial intelligence can forecast or at least mimic mankind's creative capacity; reconciling the gap between imagination and computation. The methodology leverages the low-dimensional embedding of the dataset in the latent space as a means to determine the optimal transport map between clusters (representing movements in history) and hence interpolate on the next distribution (movement) in the sequence.
- Conduct a holistic user-centred research to better grasp how subjects interact with generated art and how they compare it to existing works from similar styles or genres.

The overall flow of the MichelGANgelo framework is displayed in Fig. 3. A more detailed analysis of the different components in this diagram will be elaborated further in Section 5.

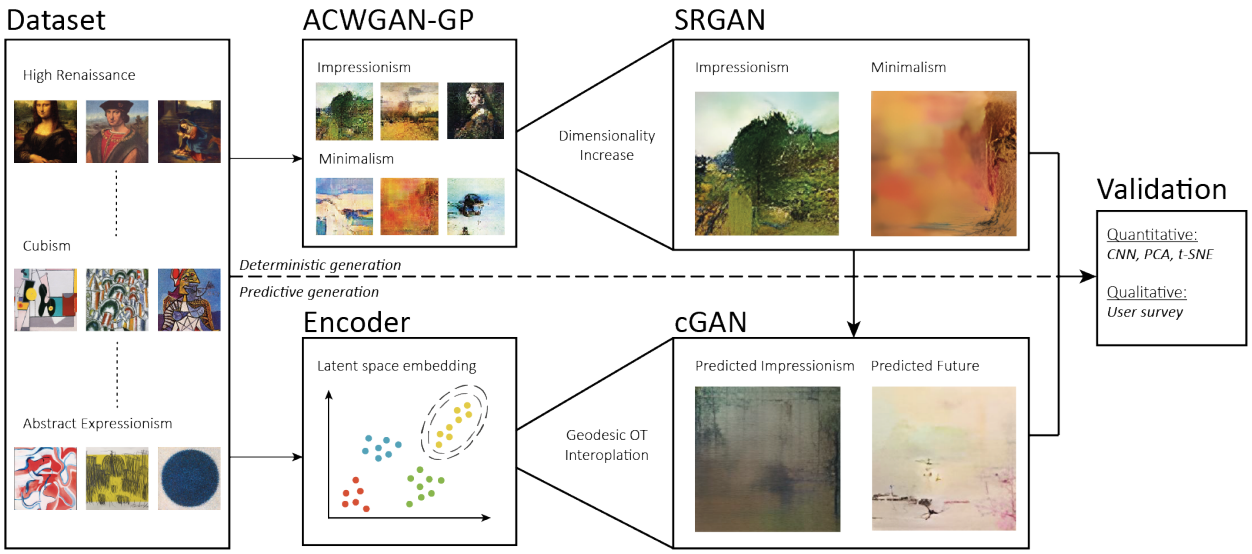


Fig. 3: Overview of the full MichelGANgelo framework.

4 Background research

4.1 Convolutional Neural Networks

4.1.1 A brief introduction to CNNs

CNNs are central to feature extraction for image classification which is why understanding their parameters is key in order to better leverage them when modifying the architecture and fine-tuning it. This type of network is composed of multiple neuron layers, each being a nonlinear operation on a linear transformation of the preceding layer's outputs. The layers mainly include convolutional and pooling layers. Abstractly, a (discrete) convolution is defined as a product of functions f and g that are objects in the algebra of Schwartz functions in $\mathbb{R}^n$, given by:

$$(f * g)[x] = \sum_{m=-\infty}^{\infty} f[m]g[x-m] \tag{4}$$

4.1.2 The layers of a CNN

A convolutional layer contains a set of filters whose parameters have to be learned. Using the calculatory methodology of [38], those filters are then convolved with an input image $\mathbf{A}^{(m-1)}$ (with K_m channels) in order to compute a new image $\mathbf{A}^{(m)}$ (with O_m *channels*) whose output at each channel generates a *feature map*. The dimensions of the filter are smaller than the input which means that the receptive field size (region in the input space that the CNN's feature is affected by) of each neuron is equal to the filter size.

$$\mathbf{A}_o^{(m)} = g_m(\sum\nolimits_k \mathbf{W}_{ok}^{(m)} * \mathbf{A}_k^{(m-1)} + b_o^{(m)}) \tag{5a}$$

$$\mathbf{W}_{ok}^{(m)} * \mathbf{A}_k[s,t] = \sum\nolimits_{p,q} \mathbf{A}_k[s+p, t+q]\mathbf{W}_{ok}[P-1-p, Q-1-q] \tag{5b}$$

where $*$ is the (2-D) convolution operation, $\mathbf{W}_{ok}^{(m)}$ is a matrix of shape $P_m \times Q_m$ and $b_o^{(m)} \in \mathbb{R}$. The matrix $\mathbf{W}_{ok}^{(m)}$ parameterizes a filter that the layer uses to detect or enhance some feature in the incoming image and that is learnt from data during the training of the network.

Another essential aspect of CNNs are its pooling layers which are incorporated between two successive convolutional layers. They are dimensionality reduction operations used to diminish the

amount of trainable parameters for the next layers, enabling them to focus on bigger areas of the input pattern. This is done by down-sampling the representation that is is being given. As per the previously used methodology, taking an image $\mathbf{A}^{(m-1)}$, a pooling layer of size $P_m, Q_m \in \mathbb{N}$ and strides $\alpha_m, \beta_m \in \mathbb{N}$ will usually implement a channel-wise operation as presented below:

$$\mathbf{A}_o^{(m)}[s,t] = \kappa \cdot (\sum\nolimits_{p,q} (\mathbf{A}_o^{(m-1)}[\alpha_m s + p, \beta_m t + q])^{\rho})^{\frac{1}{\rho}} \tag{6}$$

where $\kappa, \rho \in \mathbb{N}$ are fixed parameters. It must be noted that by setting $P_m = Q_m = \alpha_m = \beta_m$, we are inherently dividing each input channel into non-overlapping $P_m \times Q_m$ patches and replacing the values in that region by one determined by ρ and κ. There are two main pooling methods which vary depending on κ and ρ. Max pooling ($\rho =, \kappa = 1$) which is the maximum value of the values in the given patch or average pooling ($\rho = 1, \kappa = 1/PQ$) which is the average of values in the patch. It must be said that max pooling is more commonly used.

Another type of layer which is required are non-linear activation functions which define the output of a neuron given a specific input. ReLU (Rectified Linear Unity) or LeakyReLU is often a good choice as it simplifies derivations and often suppresses vanishing gradient problem because the derivative of the function when x is positive is always 1. The difference between the two methods is that when an input is negative, the derivative of ReLU will be zero (leading to dying ReLU problem) while that of LeakyReLU will be a pre-determined parameter as shown below:

$$\frac{d}{dx} ReLU(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0 \end{cases} \qquad \frac{d}{dx} LeakyReLU(x) = \begin{cases} \omega, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0 \end{cases} \tag{7}$$

Apart from layer types and non-linearities, the convolutional neural network will need to be fine tuned with parameters such as stride (determines the amount of movement of the kernel over the input), padding (amount of pixels usually set to 0 added to the input to maintain dimensionality uniformity) or learning rate (determines how quickly the model moves towards the minimum).

4.1.3 Training phase: optimizers and loss function

The learning phase occurs when the network's parameters are optimised to minimise a given error. A key element is the optimiser; Stochastic Gradient Descent and Adam being the two most popular.

On the one hand, SGD is a constrained optimization algorithm in which the gradient of the cost function of a randomly selected point is computed in order to iteratively reduce the error by modifying network parameters through backpropagation. Following the proof of [39], suppose $(\Omega, F, \mathbb{P})$ is a probability space, $L : \Omega \times \mathbb{R}^N \to \mathbb{R}$ a function depending on a random argument ω and a parameter X to be optimised, the function is defined below:

$$\mathcal{L}(X) = \mathbb{E}_\omega[L(\omega), X] \tag{8}$$

The goal of the SGD algorithm is to find a minimum of $\mathcal{L}$ and as mentioned previously, it does so in an iterative process by taking at iteration n:

- a (deterministic) learning rate ρ_n (schedule fixed *a priori*)
- a random $\omega_n \in \Omega$ independent of any other previous random variables is drawn (following $\mathbb{P}$)
- an update rule following the formula in Eq. 9

$$X_{n+1} = X_n - \rho_n \nabla_x L(\omega_n, X_n) \tag{9}$$

On the other hand, the Adam optimizer is also an iterative method which repeats *Adaptive Moment Estimation* cycles extending of SGD to solve non-convex problems faster while using less resources. Following the proof in [40], let $w \in \mathbb{R}^n$ be the weights of a function $f(w) \in C^2(\mathbb{R}^n, \mathbb{R})$, which we seek to minimize. The function $g(w) \to \nabla f(w) \in \mathbb{R}^n$ is defined as the gradient of f where the state variable of the aforementioned dynamical system is $x = (m, v, w)$. The Adam optimizer algorithm may therefore be defined recursively as showed in Eq. 10.

$$m_{t+1} \to \beta_1 m_t + (1 - \beta_1) g(w_t) \in \mathbb{R}^n \tag{10a}$$

$$v_{t+1} \to \beta_2 v_t + (1 - \beta_2) g(w_t) \otimes g(w_t) \in \mathbb{R}^n \tag{10b}$$

$$w_{t+1} \to w_t - \alpha \frac{\sqrt{1 - \beta_2^{t+1}}}{1 - \beta_1^{t+1}} (m_{t+1} \oslash \sqrt{v_{t+1} \oplus \epsilon}) \in \mathbb{R}^n \tag{10c}$$

After determining the optimiser, a loss function must also be picked. For an image classifying task in which probabilities are most likely required, cross-entropy loss is generally chosen as it outputs

a rank 1 vector with as many values as there are classes. In this case, categorical cross entropy (defined below) may be used if the labels have been one-hot encoded.

$$H(t,s) = -\sum_{i}^{K} t_i \log(s_i) \tag{11}$$

where t_i and s_i are respectively the ground truth and the CNN score for each i in K (the classes of the model). Probabilistically speaking, we can set $t_i = p(K_i)$ as the true probability distribution (as a one-hot encoded vector) and $s_i = q(K_i)$ as the model's predicted probability distribution.

4.2 A primer on Generative Adversarial Networks

4.2.1 The initial formulation of GANs

In 2014, Goodfellow introduced the concept of Generative Adversarial Networks [41] which is a usually unsupervised framework in which two neural networks (a Generator G and a Discriminator D) compete against each other in the form of a zero-sum game where G tries to fool D, and D tries to G from fooling it (Fig. 4). G produces random samples which should have the same probability distribution as the training data denoted as p_{data}, without having access to this data. D examines the outputs to verify whether they are real or not (whether they come from p_{data} or not).

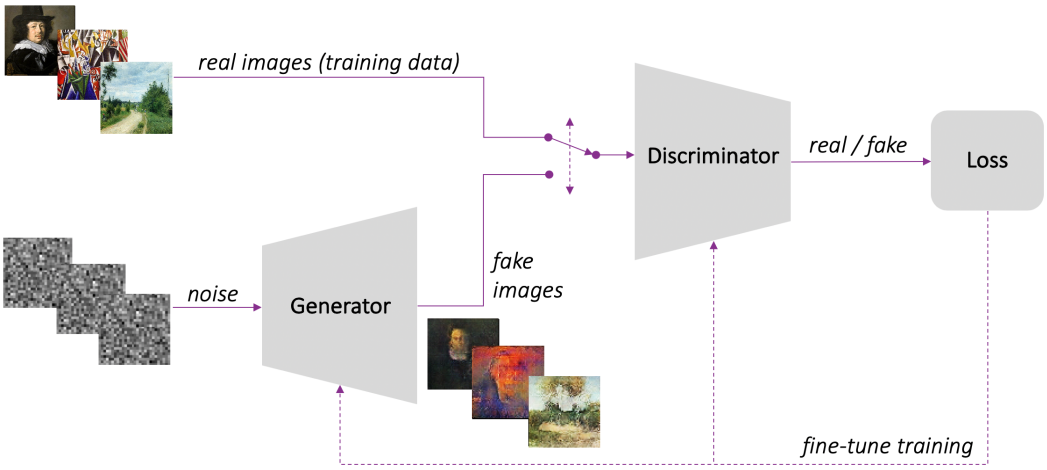


Fig. 4: Generative Adversarial Network architecture

Following the proof in Goodfellow's initial research, to learn the Generator's distribution p_g over data $\mathbf{x}$, we define a prior on input noise variables $p_z(\mathbf{z})$, then represent a mapping to data space as $G(\mathbf{z};\theta_g)$, where G is a differentiable function with parameters θ_g. We also define $D(\mathbf{x};\theta_d)$ which outputs a single scalar where $D(\mathbf{x})$ represents the probability that $\mathbf{x}$ came from the data rather than p_g. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize $\log(1 - D(G(\mathbf{z})))$. The objective function behind the aforementioned procedure is shown below:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z}[\log(1 - D(G(\mathbf{z})))] \tag{12}$$

where $\mathbf{z} \sim p_z = N(\mu, \sigma)$ is a noise vector and $\mathbf{x} \sim p_{data}$ is a real image.

4.2.2 Limitations of this framework

The training procedure of GANs as devised by Goodfellow in Alg. 1 shows that both Generator and Discriminator are trained separately which is why correctly training such a dual network is often complex as the Discriminator should not overpower the Generator and vice versa. A major issue which regularly occurs during training is known as mode collapse and takes place when the Generator is stuck into a setting where it always outputs the same result. This is due to the fact that the algorithm's purpose is for G to fool D, not to generate multiple diverse outputs.

Algorithm 1 Minibatch SGD training of GANs

for number of training iterations **do**
  for k steps **do**
    Sample m noise samples $\{z_{(1)}, \cdots, z(m)\}$ from noise prior $p_g(\mathbf{z})$
    Sample m examples $\{x_{(1)}, \cdots, x_{(m)}\}$ from data generating distribution $p_{data}(\mathbf{x})$
    Update the Discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} [\log D(\mathbf{x}^{(i)} + log(1 - D(G(\mathbf{z}^{(i)})))] \tag{13}$$

  end for
  Sample m noise samples $\{z_{(1)}, \cdots, z(m)\}$ from noise prior $p_g(\mathbf{z})$
  Update the Generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(\mathbf{z}^{(i)}))) \tag{14}$$

end for

It must be noted that one limitation of GANs' loss function is common to other neural networks and is known as the vanishing gradient. In this case, it arises from $\mathcal{L}^{(G)}$ which in turn comes from Kullback-Leibler (KL) and Jensen-Shannon (JS) divergences given below:

$$\mathcal{L}^{(G)} = 2JS(p_{data} \parallel p_g) - 2\log 2 \tag{15}$$

$$KL(p_1 \parallel p_2) = E_{x\sim p_1} \log(\frac{p_1}{p_2}) \tag{16a}$$

$$JS(p_1 \parallel p_2) = \frac{1}{2}KL(p_1 \parallel \frac{p_1+p_2}{2}) + \frac{1}{2}KL(p_2 \parallel \frac{p_1+p_2}{2}) \tag{16b}$$

The goal of the optimization procedure for Eq. 15 is to move p_g towards p_{data} in order to optimise D. JS divergence will remain constant when there is no overlap between p_{data} and p_g and the gradient is no-zero only when p_{data} and p_g significantly overlap, meaning D is close to its optimal value and G will therefore face vanishing gradient problem. This is why GANs modify the original loss function of G as the one shown below:

$$\max_G E_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(D(G(\mathbf{z})))] \tag{17}$$

This initial research spurred further advancements in the field with the implementation of CNN structures in Deep Convolutional GANs [42] as well as much more advanced concepts such as WGANs, Cycle-Consistent GANs or Progressive Growing GANs. Those novel architectures tend to stabilise the model which renders better results and reduce the chance of mode collapse.

4.2.3 A brief explanation of Conditional GANs

Although numerous GAN frameworks exist and will be used later during the project, it is of interest here to briefly mention Conditional GANs which were first introduced by [43]. Such networks take as input a conditioning variable c which instantiates a one-to-many mapping as a conditional predictive distribution. This variable represents an auxiliary source of information and allows to condition the model by feeding it to both networks of the model (D and G) as an additional input layer. Following the proof in [43], the prior input noise $p_z(\mathbf{z})$ and c are combined in joint hidden

representation in the Generator while $\mathbf{x}$ and c are presented as inputs in the Discriminator. The objective function of the GAN is hence modified as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D(\mathbf{x}|c)] + \mathbb{E}_{\mathbf{z}\sim p_z}[\log(1 - D(G(\mathbf{z}|c)))] \tag{18}$$

Using cGANs is relevant to our project as the goal is to generate images of a certain style, subject matter or both at once.

4.3 Embedding high dimensional data in the latent space

4.3.1 AutoEncoders

An AutoEncoder is a an unsupervised learning technique that consists of an Encoder and a Decoder as shown in Fig. 5. On the one hand, the Encoder is designed to compress a usually high dimensional input (like an image) into a low dimensional meaningful representation in the *latent* space. On the other hand, the Decoder is designed to reconstruct the input as closely as possible to its original input. Such networks have a wide range of applications such as dimensionality reduction, image compression, image denoising, feature extraction, image generation or anomaly detection.

As formally defined by [44] a simple AutoEncoder's architecture can be constructed considering the following: an input layer of size N, a hidden layer of size H and an output layer of size N as well as a set of input training vectors $\mathcal{V} = \{\mathbf{x}_i, \cdots, \mathbf{x}_M\}$, a function $\mathbf{B}$ from the input layer to the hidden layer and a function $\mathbf{A}$ from the hidden layer to the output layer. When being fed with an input vector $\mathbf{x}$, the network generates a hidden vector $\mathbf{h} = \mathbf{B}(\mathbf{x})$ and an output vector $\mathbf{y} = \mathbf{AB}(\mathbf{x})$. The objective of an AE is to minimize and error function E displayed below:

$$\min E(\mathbf{A},\mathbf{B}) = \min_{\mathbf{A},\mathbf{B}} \sum_{i=1}^{M} \Delta(\mathbf{x}_i, \mathbf{y}_i) = \min_{\mathbf{A},\mathbf{B}} \sum_{i=1}^{M} \Delta(\mathbf{AB}(\mathbf{x}_i), \mathbf{x}_i) \tag{19}$$

where Δ is a distance or dissimilarity measure (usually Euclidean or Hamming distances). This model assumes that the transformations $\mathbf{A}$ and $\mathbf{B}$ belong respectively to two classes of transformations A and B. The input vectors follow $\mathbb{F}$ while the hidden vectors follow $\mathbb{G}$ and in most simple cases, $\mathbb{F} = \mathbb{G}$, both being considered as fields.

In the case of our method, autoencoding is leveraged to embed the high dimensional images of paintings into the *latent* space and sampling data from it to perform feature extraction.

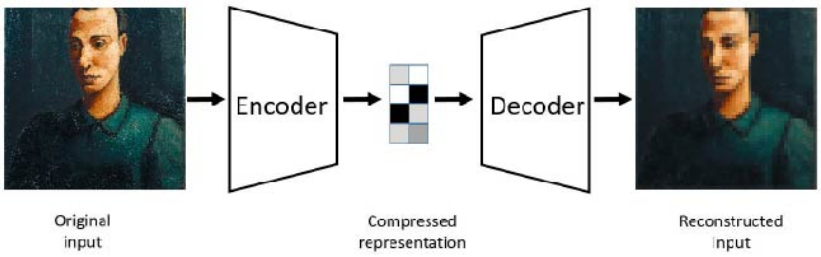


Fig. 5: Basic AutoEncoder architecture

4.3.2 Variational AutoEncoders

As mentioned, more complex AE frameworks have been designed but one is of particular interest, namely Variational AutoEncoders. A VAE's Encoder compresses an input dataset $\mathbf{x}$ defined by a probability distribution function $p(\mathbf{x})$ and a multivariate latent encoding vector $\mathbf{l}$ into a latent space $p_\theta(\mathbf{x})$ (i.e. a distribution with network parameters θ) which is used as a representation of compressed data. Then, the Decoder receives the information sampled by the Encoder to produce a dataset $\hat{\mathbf{x}}$ which aims at being as close to $\mathbf{x}$ as possible.

To better understand VAEs, we will briefly have a closer look at how exactly they function by following the proof from [45]. It states that given an observed dataset $\mathbf{X} = \{\mathbf{x}_1\}_{i=1}^N$, the defined generative model (known here as the *probabilistic Decoder*) is assumed for every $\mathbf{x}_i$ conditioned on an unobserved random *latent* variable $\mathbf{z}_i$ with θ representing the parameters that govern the generative distribution. This assumption is also reciprocally considered when defining the *probabilistic Encoder* governed by parameters ϕ and which uses an approximate posterior distribution over $\mathbf{z}_i$ given an $\mathbf{x}_i$ denoted by recognition. We also take into account a prior distribution for each and every $\mathbf{z}_i$ written as $p_\theta(\mathbf{z}_i)$. Both set of parameters θ and ϕ will need to be learned during the training process from the data. Moreover, the variables $\mathbf{z}_i$ can be viewed as a *code* which is delivered by the recognition model set as $q_\phi(\mathbf{z}|\mathbf{x})$.

One of the reasons VAEs perform better than other AutoEncoding techniques resides in their use of the marginal log-likelihood. As simply defined in [46], in Bayesian statistics, the marginal likelihood (sometimes referred to as the *evidence*), is used to evaluate model fit as it quantifies the joint probability of the data under the prior. For VAEs, this metric is represented as a sum over the individual data points $\log p_\theta(\mathbf{x}_1, \cdots, \mathbf{x}_N) = \sum_{i=1}^N \log p_\theta(\mathbf{x}_i)$ where each point is defined as Eq. 20. In this equation, the first term is the KL divergence of the approximate recognition model from

the true posterior while the second term is the *variational lower bound* on the marginal likelihood (also called ELBO) and which is a useful lower bound on the log-likelihood of the observed data (this term is defined in Eq. 21). The variational inferences that is being utilized by VAEs can be calculated by maximizing the *variational lower bound* ($\mathcal{L}(\theta, \phi; \mathbf{x}_i)$) for all data points with respect to the two sets of parameters θ and ϕ.

$$\log p_\theta(\mathbf{x}_i) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i) \parallel p_\theta(\mathbf{z}|\mathbf{x}_i)) + \mathcal{L}(\theta, \phi; \mathbf{x}_i) \tag{20}$$

$$\mathcal{L}(\theta, \phi; \mathbf{x}_i) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_i) \parallel p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i|\mathbf{z})] \tag{21}$$

Once this single-vector value is computed, determining the marginal log-likelihood lower-bound of the entire defined dataset $\mathbf{X}$ (size N) using a minibatch $\mathbf{X}^M = \{\mathbf{x}_i\}_{i=1}^M$ (size M) can relatively simply be derived as per Eq. 22.

$$\mathcal{L}(\theta, \phi; \mathbf{X}) \approx \tilde{\mathcal{L}}^M(\theta, \phi; \mathbf{X}^M) = \frac{N}{M} \sum_{i=1}^{M} \mathcal{L}(\theta, \phi; \mathbf{x}_i) \tag{22}$$

Although we will not dive much further into this concept, it must be noted that VAEs act differently than variational Bayesian methods in that the gradients of the marginal log-likelihood $\tilde{\mathcal{L}}^M(\theta, \phi; \mathbf{X}^M)$ are approximated using a technique called the reparameterization trick combined with stochastic gradient optimization. The result from this approximation can be seen in the right part of Eq. 22 and can be derived according to the parameters θ and ϕ which later allows us to plug it back into an optimizer framework. The full methodology behind a VAE can be viewed in Alg. 2.

Algorithm 2 Training of a VAE

$(\theta, \phi) \leftarrow$ Initialize Parameter
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M data points
  $\epsilon \leftarrow L$ random samples of $p(\epsilon)$
  $\mathbf{g} \leftarrow \nabla_{(\theta,\phi)}(\tilde{\mathcal{L}}^M)(\theta, \phi; \mathbf{X})$ (Gradients of Eq. 22)
  $(\theta, \phi) \leftarrow$ Update parameters based on $\mathbf{g}$ (SGD often used)
until Convergence of (θ, ϕ)
return (θ, ϕ)

4.4 Optimal Transport theory

4.4.1 The origins of Optimal Transport: Monge-Kantorovich formulation

Optimal Transport is another central topic to this paper, used as a transfer learning methodology, to understand the links between disjoint datasets' latent spaces as well as a stabilisation method for the generative network. Cuturi [47] even defines it as a "mathematical gem at the interface between probability, analysis and optimization". At its core, this problem introduced by Monge [48] aims at displacing a body of mass with a variable volume with the least amount of energy (see Fig. 6).

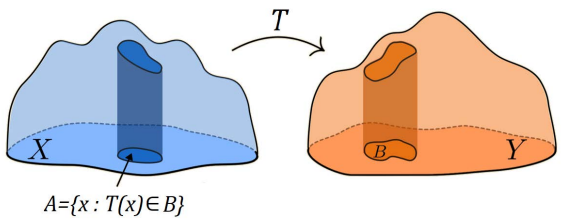


Fig. 6: Monge's Optimal Transport map visualization

Theoretically speaking, [49] explains that Optimal Transport is a way to compare measures μ and ν in a Lagrangian framework. Both of these measures are assumed to be probabilities on spaces X and Y respectively. By setting $c(x, y)$ where $c : X \times Y \in \mathbb{R}^+$, a cost function which quantifies the displacement of a unit mass from $x \in X$ to $y \in Y$. Hence, the Optimal Transport problem is the one by which we transport μ to ν while minimising the cost c of doing so. In a more rigorous notation, $T : X \to Y$ transports $\mu \in \mathcal{P}(X)$ to $\nu \in \mathcal{P}(Y)$ where T is our transport map (only true if $\nu(B) = \mu(T^{-1}(B))$ for all ν-measurable sets B). These notations allow us to compute Monge's Optimal Transport problem ($\min \mathbb{M}(T)$) in which the transfer function $\mathcal{T}_c(\mu, \nu)$ as follows:

$$\min \mathbb{M}(T) = \int_X c(x, T(x)) d\mu(x) \tag{23a}$$

$$\mathcal{T}_c(\mu, \nu) = \inf_{T \# \mu = \nu} \mathbb{M}(T) \tag{23b}$$

It should be noted that the notation below uses inf instead of min as the minimum may not exist. The symbol # represents the connected sum of two manifolds (on either side of the sign).

Later, Kantorovich developed a relaxed formulation of Monge's theorem, putting forward the fact that in the initial formula, the *mass* is mapped entirely with the transport map ($x \mapsto T(x)$) which may cause issues in the discrete case. For this reason, he made it possible for the *mass* to be split, allowing x_1 to go to y_1 and x_2 to go to y_2 for instance. By following the same methodology as before, we consider $\pi \in \mathcal{P}(X \times Y)$ and consider $d\pi(x, y)$ as the amount of mass transferred from x to y which makes it possible to deliver x to several locations. This entails that any mass removed from a set $A \subset X$ has to equal $\mu(A)$ which can be reciprocally stated for a set B as $B \subset Y = \nu(B)$. This allows us to compute Kantorovich's Optimal Transport problem as shown below:

$$\min \mathbb{K}(\pi) = \int_{X \times Y} c(x, y) d\pi(x, y) \tag{24}$$

where $\pi \in \prod(\mu, \nu)$ with $\prod(\mu, \nu)$ defined as the set of *transport plans* between μ and ν.

4.4.2 An overview of the Wasserstein distance

If both these Optimal Transport formulations are relevant, another development at the heart of our method is the Wasserstein distance. The Wasserstein distance differs in that it takes into account the geometry of the underlying space which can also be coined here as a *geodesic space*. By following the proof in [49], given that $\mu, \nu \in \mathcal{P}_p(X)$, the Wasserstein distance can be defined as follows: 25.

$$d_{W^p}(\mu, \nu) = \min_{\pi \in \prod(\mu, \nu)} \left(\int_{X \times X} |x - y|^p d\pi(x, y) \right)^{\frac{1}{p}} \tag{25}$$

Although we will not develop the proof here, the Wasserstein space $(\mathcal{P}_p(X), d_{W^p})$ must be considered as being a *geodesic space*. In simple terms, the term *geodesic* refers to the shortest path between two points in a surface which may be generalized as a Riemannian manifold. An important feature of the Wasserstein distance is that it is differentiable which makes it a usable metric when building Generative Adversarial Networks.

4.4.3 Applications of (Computational) Optimal Transport

Optimal Transport has become a powerful way to compare probability distributions with each other as well as compute mappings to minimize cost functions [50]. We often refer to such methods as *computational* optimal transport as the goal is to minimize the computational cost of a transport function. Its link to deep generative models is mostly due to its implication in specific elements such

as Convolutional Wasserstein Distance [51] (Eq. 26 (a)) which is currently used as an error measure for the comparison of different probability distributions. It should be noted that such distance can be used alongside barycentres in the Wasserstein space for tasks such as image interpolation which may be of interest later in this project. Other metrics such Minibatch Energy Distance [52] (Eq. 26 (b)) bring a greater statistical consistency by improving on the idea of mini-batches with the implication of energy functions and OT.

$$W_{2,\gamma}^2(\mu,\nu) = \gamma[1 + \min_{\pi \in \prod} KL(\pi|K_\gamma)] \tag{26a}$$

$$D_{MED}^2(\mu,\nu) = 2\mathbb{E}[W_c(X,Y)] - \mathbb{E}[W_c(X,X')] - \mathbb{E}[W_c(Y,Y')] \tag{26b}$$

where γ represent the mass that must be transported from x to y in order to transform the distributions from μ to ν, $K_\gamma(x,y) = e^{-\frac{d(x,y)^2}{\gamma}}$ is a kernel function, X, X' and Y, Y' are mini-batches sampled from μ and ν respectively and $c = c^*$ is the optimal transport function learned in an adversary manner.

5 Methods

5.1 Dataset

Throughout this project, the publicly available `Wikiart` dataset is used to train the different models. The initial dataset contains 87312 images divided into 22 classes (representing different styles). A CSV file subdivides each style category further into 10 subject-matter classes. However, the whole dataset is not of interest because the goal of this paper is to better understand the links between artistic movements. To do so, there must be a *link* which is why only Western art is kept in the dataset. Moreover, some movements are very similar historically speaking and can easily be merged in order to reduce potential confusion. For instance movements such as *Analytical Cubism* and *Synthetic Cubism* are encompassed into the broader *Cubism* movement. By cutting down the dataset to a sufficient 14 classes and removing all non-painting images, the dataset is reduced to 56846 images. Moreover, when running tasks on subject-matter, three classes are often preferred as they usually have more data: portraits, landscapes and still life.

Table 1: Modified `Wikiart` dataset

Style	N°	Period	Added movements
High Renaissance	1333	1500-1550	-
Baroque	4239	1600	-
Realism	10702	1848	-
Impressionism	13044	1860s	-
Post-Impressionism	6425	1886-1905	-
Naive Art Primitivism	2392	1885	-
Pointillism	507	1886-1905	-
Fauvism	934	1905-1908	-
Expressionism	6696	1905-1920	-
Cubism	2541	1907-1914	Analytical & Synthetic Cubism
Abstract Expressionism	2848	1940-1950	Action Painting
Minimalism	2858	1955-1960	Color Field Paintings
Pop Art	1466	1950-1960	-
Contemporary Realism	785	1960s	New Realism

5.2 State-of-the-art CNN

5.2.1 A CNN built from scratch

To better understand the intrinsic features that makes each movement so specific and to also have a tangible point of validation for the generative model, a goal of this project is to develop a fine-art style classifier. Initially, a very simple CNN was built and trained from scratch, containing three blocks composed of five layers, namely: a convolution, a ReLU, a convolution, a ReLU and finally a MaxPool. After those three blocks, a final one first flattens the output and then connects all of the nodes by using three linear layers with ReLU layers in between each of them. However, this proved to result in very low accuracy of about 15%.

5.2.2 Using pretrained architectures

Pretrained architectures, especially residual neural networks and in particular ResNet50 are implemented to boost accuracy. The reason for this is that ResNets can train very deep networks and still achieve compelling results as they are able to deal efficiently with the vanishing gradient problem. This is possible thanks to the *identity shortcut connection* which skips one or more layers as shown in Fig. 7. In ResNet50, each residual block consists of three convolutions with 1x1 filters on conv1 and conv3 and 3x3 or 5x5 filter on conv2. The network has input dimensions $224 \times 224 \times 3$ and batch normalisation is used throughout the architecture. As most pretrained models, the last layer needs to be modified to cater for the number of classes.

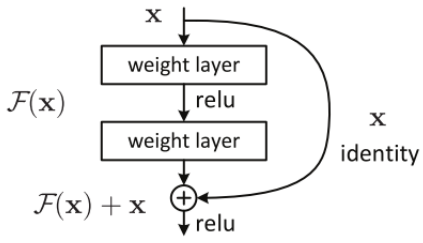


Fig. 7: Diagram of a residual block with an identity shortcut connection.

5.2.3 Modifying the architecture

The sequential nature of data means that movements closer in time are often more similar to one another. This difference is most notable for artworks before and after the 1910s, especially with the dawn of cubism that brought immense changes to the art world. Hence, we develop an architecture which separates the data into two batches: pre-1910s (8 classes) and post-1910s (6 classes). The architecture of the CNN is now composed of two steps: an initial binary classifier and then a classifier for individual movements contained in both batches as shown in Fig. 8. All networks are using ResNet50 architectures.

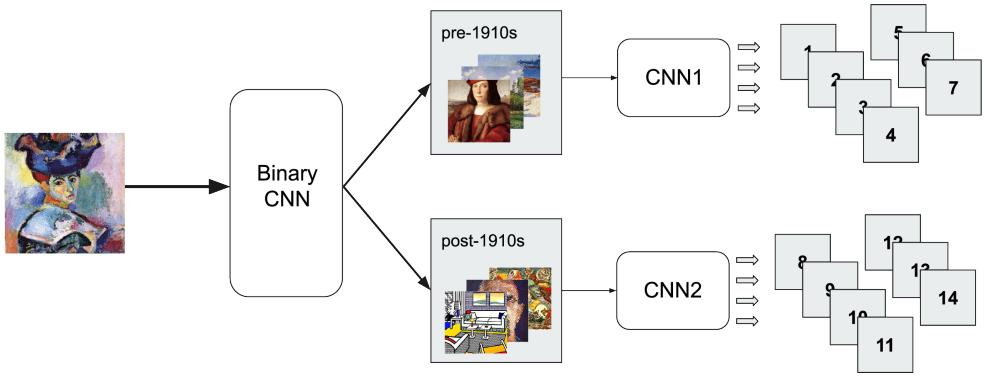


Fig. 8: Diagram of the final CNN architecture for fine-art style classification.

5.3 An ACWGAN-GP model that generates art

5.3.1 Network architecture

The developed Auxiliary Classified Wasserstein GAN with Gradient Penalty is inspired from the Improved WGAN model [5] and adds an auxiliary classifier to the architecture. As for any other GAN models, our architecture is composed of a Generator and a Discriminator. The reason why an ACGAN architecture was favoured over the classical cGAN model when conditioning the network is discussed in the Section 6.2.2. The architecture of the network can be seen in Fig. 9.

On the one hand, the Generator takes as an input random Gaussian noise of size 128 (denoted $\mathbf{z} \in \mathbb{R}^{128}$) sampled between 0 and 1 as well as the labels of the dataset. It is first up-sampled to a convolutional representation containing 2048 feature maps which results in a tensor of size

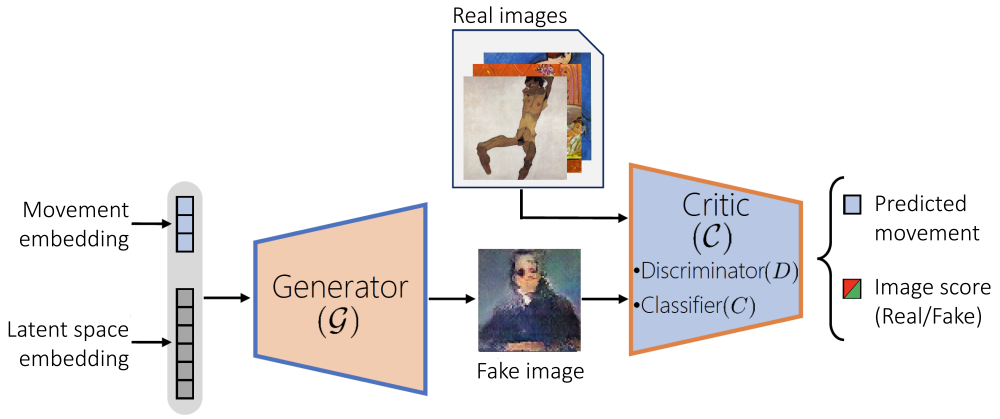


Fig. 9: Diagram of the final ACWGAN-GP model.

$4 \times 4 \times 2048$. After that, a series of four deconvolutions more commonly called fractionally-strided convolutions are implemented before converting this representation to a 64×64 image. Starting from the noise, we basically get the following pipeline: $\mathbf{z} \in R^{128} \rightarrow 4 \times 4 \times 1024 \rightarrow 8 \times 8 \times 512 \rightarrow 16 \times 16 \times 256 \rightarrow 32 \times 32 \times 128 \rightarrow 64 \times 64 \times 3$. When increasing generated image size to 128×128, another fractionally-strided convolution layer is added to the network. Nonlinear layers are implemented in between each fractionally-strided convolution. In our case, the nonlinear layer is derived from the conditional PixelCNN network, the reasons behind this choice are developed in Section 6.2.3. Moreover, the label is also propagated throughout the network to successfully condition the GAN on the class it is tasked to generate.

On the other hand, the Discriminator is composed of 4 convolutional layers taking the following structure $128 \times 64 \times 64 \rightarrow 256 \times 32 \times 32 \rightarrow 512 \times 16 \times 16 \rightarrow 1024 \times 8 \times 8 \rightarrow 2048 \times 4 \times 4$. Just like for the Generator, when increasing the generated output image to 128×128, another convolution layer must be added. After those convolution layers, the Discriminator network is composed of two heads: a source output (determining whether the image is art or not) and the multi-label output (determining in which class the image is). Each convolution layer of the network is followed by a nonlinear layer (LeakyReLU here). After an image passes through the Discriminator, it produces a $4 \times 4 \times 512$ feature map. The real/fake part of the Discriminator collapses the feature map into

a fully connected layer to produce $D_r(c|x)$ whilst the multi-label part of the Discriminator will collapse into a fully connected layer of size K to produce $D_c(c_k|x)$, where K is the number of labels (style or subject matter depending on the generative task) and c the condition.

5.3.2 Training procedure

The presented network is characteristic of an ACGAN architecture. However, as demonstrated in [14], using Computational Optimal Transport in the training procedure of the network by implementing the Wasserstein distance in the loss function is of interest in order to stabilise the model and help it converge more accurately. Moreover, as per [5], using Gradient Penalty over Weight Clipping in the Discriminator training tends to increase network performance. Therefore, a hybrid ACWGAN-GP framework is implemented and aims to minimize two loss functions, namely Eq.27 (a) for the Generator and Eq.27 (b) for the Critic (a combination of Discriminator and Classifier).

$$L_G = \underbrace{-\mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}[D(\tilde{\mathbf{x}})]}_{\text{Original Generator Loss}} + \underbrace{\lambda_2 \mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}[-\log(P[C(\tilde{\mathbf{x}}) = c_{\tilde{\mathbf{x}}}])]}_{\text{Cross-entropy Loss}} \tag{27a}$$

$$L_C = \underbrace{\mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}[D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x}\sim p_{data}}[D(\mathbf{x})]}_{\text{Original Discriminator Loss}} + \underbrace{\lambda_1 \mathbb{E}_{\hat{\mathbf{x}}\sim p_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2]}_{\text{Gradient Penalty}} + \underbrace{\lambda_2 \mathbb{E}[-\log(P[C(\mathbf{x}) = c_x])]}_{\text{Cross-entropy Loss}} \tag{27b}$$

where λ_1 is the gradient penalty term, λ_2 is the true probability distribution which must be set to 1 as it is a one-hot encoded vector and c_x is the true label associated with an image $\mathbf{x}$ (whether real or generated).

In our work, similar to [10], the weights are initialized from a zero-centred normal distribution with a standard deviation of 0.02. Batch size is set between 64 and 128 when generating 64×64 images but must be reduced to 8 for 128×128 images due to increased computational costs. The number of critic iterations is initially set to 5 but increased to 10 for better results. The Adam optimizer is used for both Generator and Discriminator as it proves to accelerate the training, learning rate (α) is set to 0.0001, the exponential decay rate for the first moment is 0.5 (β_1) and that for the second moment is 0.9 (β_2). Batch normalization is used for both Generator and Discriminator as a means to stabilize the model by setting the input to each input unit to have a mean of 0 and a variance of 1. The training algorithm is detailed in Alg. 3.

Algorithm 3 Training algorithm of the ACWGAN-GP architecture.

Require: $\lambda_1 = 10, \lambda_2 = 1, n_c = 10, \alpha = 0.0001, \beta_1 = 0.5, \beta_2 = 0.9, \text{batch} = m$
Require: initial critic parameters w_0, initial Generator parameters θ_0
  while θ has not converged **do**
    for $t \in \{1, \dots, n_c\}$ **do**
      for $i \in \{1, \dots, m\}$ **do**
        Sample real data $\mathbf{x} \sim p_{data}$, latent variable $\mathbf{z} \sim p(\mathbf{z})$, a random number $\epsilon \sim U[0, 1]$
        $\tilde{\mathbf{x}} \leftarrow G_\theta(\mathbf{z})$
        $\hat{\mathbf{x}} \leftarrow \epsilon\mathbf{x} + (1 - \epsilon)\tilde{\mathbf{x}}$
        $\mathcal{L}^{(i)} \leftarrow D_w(\tilde{\mathbf{x}}) - D_w(\mathbf{x}) + \lambda_1(\|\nabla_{\hat{\mathbf{x}}} D_w(\hat{\mathbf{x}})\|_2 - 1)^2 - \lambda_2 \log(P[C(\mathbf{x}) = c_x])$
      end for
      $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{i=1}, w, \alpha, \beta_1, \beta_2)$
    end for
    Sample a batch of latent variables$\{\mathbf{z}^{(i)}\}_{i=1}^{m} \sim p(\mathbf{z})$
    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\mathbf{z})) - \lambda_2 \log(P[C(\tilde{\mathbf{x}}) = c_{\tilde{x}}]), w, \alpha, \beta_1, \beta_2)$
  end while

5.3.3 Validation measurements

Validating a GAN model quantitatively is tricky due to the subjective nature of image generation. However, Inception Score (IS) [53] is a common technique which is based on the Inception v3 Network; a DCNN architecture that classifies images from 1000 classes such as the network outputs $p(c|\mathbf{x}) \in [0, 1]^{1000}$ where $\mathbf{x}$ is the image and c the class label.

IS is a common technique which tends to correlate relatively well with human evaluation of image quality as it measures both the variety of generated images ($p(v)$) with high entropy) as well as the quality of each distinct image ($p(v|\mathbf{x})$ low entropy). The IS calculates a statistical measure of the network's output applied to the Generator's synthesised images as per: 28.

$$\text{IS}(G) = \exp(E_{\mathbf{x} \sim p_g} D_{KL}(p(c|\mathbf{x}) \parallel p(c)) \tag{28}$$

where $\mathbf{x} \sim p_g$ indicates that $\mathbf{x}$ is sampled from p_g, D_{KL} is the KL divergence between the two probability distributions p and q, $p(c|\mathbf{x})$ is the conditional class distribution and $p(c) = \int_{\mathbf{x}} p(c|\mathbf{x}) p_g(\mathbf{x})$ is the marginal class distribution.

5.4 Faithfully increasing image resolution with a SRGAN

5.4.1 Network architecture

Due to the difficulty of increasing the resolution of generated images in the GAN procedure, less computationally expensive solutions such as Super-Resolution GANs are used instead. As this is not one of the key elements of this project, the SRGAN model from [3] is fully implemented in order to increase the resolution of the outputs by a factor of 4. The proposed model is based on a ResNet model with skip-connection and diverges from the commonly used MSE metric as the optimization target by using a novel perceptual loss.

The goal of this single-image super-resolution model is to upscale a low-resolution input I^{LR} to a high-dimension one I^{SR}. To do this, a Generator network with parameters θ is trained as a feed-forward CNN. By following the initially formulated GAN procedure as per Eq. 12, a Discriminator is also defined such as the min-max problem is as follows:

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{I^{HR} \sim p_{train}(I^{HR})}[\log D(I^{HR})] + \mathbb{E}_{I^{LR} \sim p_G(I^{LR})}[\log(1 - D(G(I^{HR}))] \tag{29}$$

On the one hand, the Generator inspired by [54] is composed of two convolutional layers with 64 feature maps and square kernels of length 3, followed by batch-normalisation and ParametricReLU as the activation function. The resolution is increased by using two sub-pixel convolutional layers which was initially proposed by [55]. On the other hand, the Discriminator follows the architecture of [42] and uses LeakyReLU as its activation layer ($\alpha = 0.2$) but avoids max-pooling. Similar to the VGG network, the Discriminator is composed of eight convolutions with an increasing number of squared kernels of length 3 as follows $64 \rightarrow 64 \rightarrow 128 \rightarrow 128 \rightarrow 256 \rightarrow 256 \rightarrow 512 \rightarrow 512$. The full architecture can be seen in Fig. 10.

5.4.2 Training procedure

The main difference between the chosen SRGAN model and other architectures is that it formulates a custom perceptual loss l^{SR}, calculated as a weighted sum of a content loss based on the VGG network l^{SR}_{VGG} and an adversarial loss l^{SR}_{Gen} as follows:

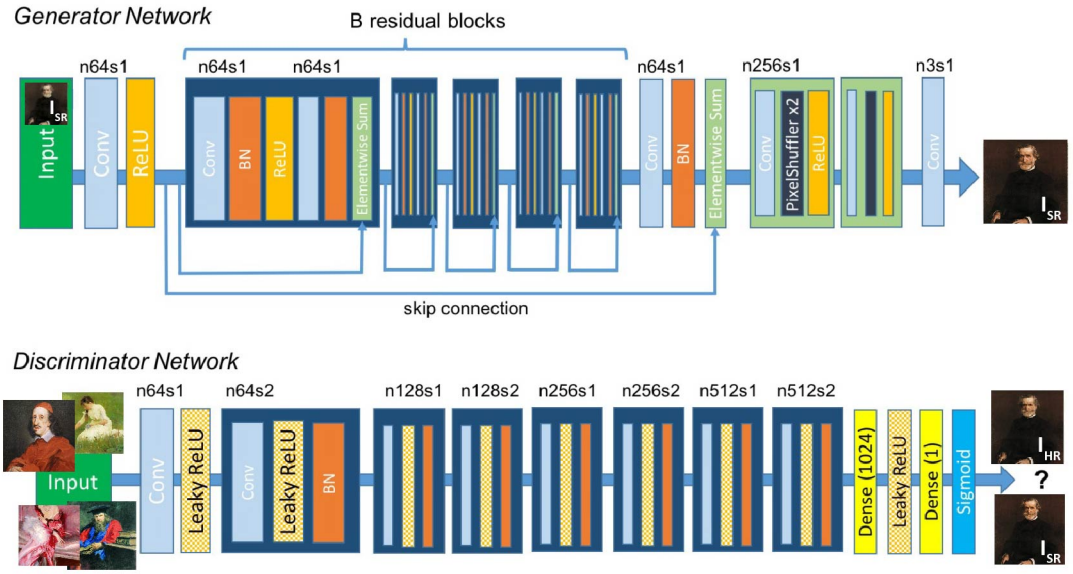


Fig. 10: Generator (top) and Discriminator (bottom) architectures of the SRGAN model with kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer [3].

$$l^{SR} = l^{SR}_{VGG} + l^{SR}_{Gen} \tag{30a}$$

$$l^{SR}_{VGG} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G(I^{LR}))_{x,y})^2 \tag{30b}$$

$$l^{SR}_{Gen} = 10^{-3} \sum_{n=1}^{N} -\log D(G(I^{LR})) \tag{30c}$$

where $W_{i,j}$ and $H_{i,j}$ are the dimensions of the feature maps in the VGG network, $D(G(I^{LR}))$ is the probability that the generated image is a super-resolution version of the input.

To upscale an image by a factor of two, the network must be trained on a dataset with image dimensions 4 times bigger. Hence, as we need images of size 224×224 for the classifier, when generating 64×64 outputs, only images larger than 1024×1024 are kept from the dataset for the up-sampling. As this reduces the amount of available data, the model is initially pre-trained on a set of 1000 hand-picked very high resolution images of textures from Unsplash [4]. The aim of that is to increase image resolution while improving stylistic quality of the work.

[4] `https://unsplash.com`

5.5 Leveraging the latent space

5.5.1 Data embedding

The data embedding procedure uses an AutoEncoder with a methodology similar to [1] as it shows very promising results for low-dimensional embedding of high-dimensional data. Following their methodology, the real data distribution is modeled as a probability distribution denoted ν_r on a h dimensional data manifold $\mathcal{X}$ embedded in the D dimensional Euclidean space $\mathbb{R}^D$ where $h << D$. The goal of the AutoEncoder is to successfully embed the real data manifold $\mathcal{X}$ into the latent manifold Ω. In doing so, we are inherently computing the encoding map f_θ and decoding map g_ξ as follows:

$$(\nu_r, \mathcal{X}) \xrightarrow{f_\theta} (\mu_r, \Omega) \xrightarrow{g_\xi} (\nu_r, \mathcal{X}) \tag{31}$$

where f_θ and g_ξ are parameterized by CNNs with θ and ξ corresponding to the parameters of the networks. These parameters are computed by minimizing a loss function $\mathcal{L}(\theta, \xi)$:

$$\mathcal{L}(\theta, \xi) \rightarrow \sum_{i=1}^{n} \|x_i - g_\xi \circ f_\theta(x_i)\|^2 \tag{32}$$

By considering a densely sampled image manifold as well as an ideal optimisation ($\mathcal{L}(\theta, \xi) \rightarrow 0$), we denote $f_\theta \circ g_\xi$ as the identity map. Once training is accomplished, f_θ is known as a homeomorphism which means that it is a continuous and invertible map whereas g_ξ is its inverse. From this, we get $f_\theta : \mathcal{X} \rightarrow \Omega$ as a data embedding which then pushes ν_r forward to the latent distribution $\mu_r \rightarrow f_{\theta\#}\nu_r$. As explained in [1], we only possess the empirical data distribution which is defined by $\hat{\nu}_r = \frac{1}{n}\sum_{i=1}^{n} \delta(x - x_i)$ which is then pushed to an embedded latent distribution defined by $\hat{\mu}_r = \frac{1}{n}\sum_{i=1}^{n} \delta(z - z_i)$ with n as the number of samples.

5.5.2 Interpolating the next movement

In order to predict the next movement in time, we base our methodology on that of video frame prediction in [4] which uses a three-layer MLP with ReLU activation. The network works in the latent space instead of the actual data space in order to better leverage the disentangled representations. Data from art movements is embedded by using the previously defined AutoEncoder as

a sequence of points in the latent space. This is possible only by considering that the chosen art movements occur one after the other in a sequential manner.

By considering an output z_0 and a target z_T we leverage the geodesic distance between $g(z_0)$ and $g(z_T)$ as the loss function instead of using a simple linear interpolation or a squared-distance $\|z_0 - z_T\|^2$. Here, we define g as the differentiable map from the latent space to the data manifold $\mathcal{X}$ which represents the action of the Decoder such as $g : \mathbb{R}^n \to \mathcal{X} \subset R^N$. The algorithm used is that from [4] and is shown in Alg. 4.

Algorithm 4 Geodesic Interpolation

Require: $z_0, z_T \in Z$, α (the learning rate)
  Initialize z_i as the linear interpolation between z_0 and z_T
  while $\Delta E_{zt} > \epsilon$ **do**
    for $i \in \{1, \cdots, T-1\}$ **do**
      $\nabla_{z_t} E_{z_t} \leftarrow -(\nabla g(z_i))^T [g(z_{i+1}) - 2g(z_i) + g(z_{i-1})]$
      $z_i \leftarrow z_i - \alpha \nabla_{z_t} E_{z_t}$
    end for
  end while

The algorithm defined in Alg. 4 aims at minimizing the energy of the path E_{z_t} in order to find the geodesic by computing its gradient $\nabla_{z_t} E_{z_t}$. Both are defined as follows:

$$E_{z_t} = \frac{1}{2} \sum_{i=0}^{T} \frac{1}{\delta t} \|g(z_{i+1} - g(z_i)\|^2 \tag{33a}$$

$$\nabla_{z_t} E_{z_t} = -(\nabla g(z_i))^T [g(z_{i+1}) - 2g(z_i) + g(z_{i-1})] \tag{33b}$$

Initially, $\{z_i\}$ is defined as being uniformly-spaced points along the line between the initial point z_0 and the target z_T. As the algorithm performs the minimization of the energy function, the distance between the points is gradually modified until the energy falls under a predefined threshold. This will also give an interesting representation of the similarity between movements as the distance between each point will represent the *distance* between each movement of the timeline. A representation of the geodesic interpolation is presented in Fig. 11.

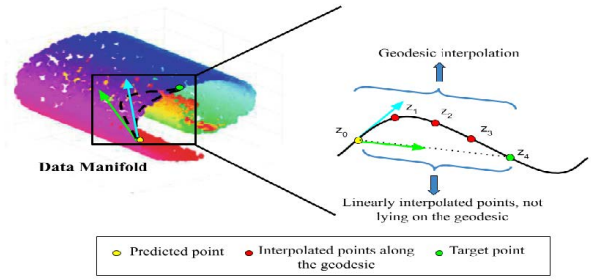


Fig. 11: Geodesic loss function used to predict z_4 as per [4].

5.5.3 Predictive modelling

By embedding each movement into the latent space, we get a sequence of probability distributions each composed of vectors with latent codes $\mathbf{c}_{k,n}$ such that $k \in K$ as the k^{th} style in the sequence and $n \in N$ as the n^{th} sample number of the chosen latent code. To generate art from a predicted movement, the goal is to condition the Generator of the GAN on a code sampled from the predicted probability distribution computed by using the geodesic interpolation method defined in 5.5.2. Hence, given random noise $\mathbf{z} \in \mathbb{R}^{d_z}$, the Generator learns the mapping $G : \{\mathbf{z}, \mathbf{c}\} \rightarrow x$ while the Discriminator estimates $p(x|\mathbf{c})$. Similar to [21], by assuming that each distribution of images $\mathcal{F}_X^{(k)}(x)$ is mapped to a probability distribution $\mathcal{F}_C^{(k)}(\mathbf{c})$ and that the sequence of distributions for $k \in \{1, \cdots, K\}$ has non-trivial relationship between each two distributions, we get:

$$\begin{array}{ccc} \mathcal{F}_X^{(1)} & & \mathcal{F}_C^{(1)} \\ \vdots & \xrightarrow{\quad\text{AE}\quad} & \vdots \\ \mathcal{F}_X^{(k)} & & \mathcal{F}_C^{(k)} \\ \vdots & & \Big\downarrow \text{geodesic} \\ \mathcal{F}_X^{(k+1)} & \xleftarrow{\quad\text{cGAN}\quad} & \mathcal{F}_C^{(k+1)} \end{array}$$

The GAN model to generate *future* art is based on a cGAN and not an ACGAN because it can condition the model on a latent code while the ACGAN conditions the model on one-hot encoded

vectors from predefined classes which exist in the data space. Indeed, we cannot condition the ACGAN on a class that does not exist in the data space. The architecture is presented in Fig. 12.

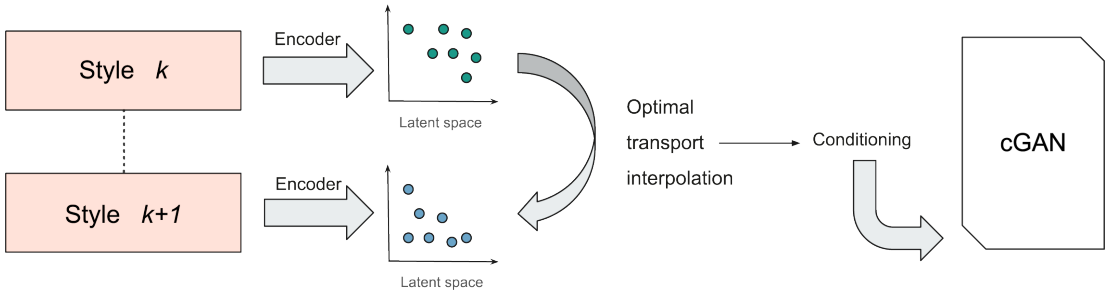


Fig. 12: Diagram of the predictive architecture based on geodesic interpolation in the latent space.

6 Discussion and Results

6.1 Getting the best out of the classifier

6.1.1 Image modification for better results

Operating on a ResNet architecture, our model must be fed images of dimension $224 \times 224 \times 3$. However, resizing the images to this size can be done in different ways which may impact the final accuracy of the algorithm. Here, three options are considered: cropped, stretched, added black space (as shown in Fig. 13). Initial results demonstrate that the cropped version enables better results. This might be due to the fact that distorting the image will intrinsically modify key stylistic features and that adding black space will reduce the available amount of information the network can use. By keeping the aspect ratio constant and cropping the image, all details such as brushstrokes or textures are kept intact.

(a) Added black space (b) Compressed (c) Cropped

Fig. 13: Modigliani's *Cafe Singer* (1917) modified in three different manners.

As a whole, the dataset has relatively small amount of data and obvious imbalances in the size of each classes. According to [56], state-of-the-art classifiers usually require about 5000 images per class. However, although our modified dataset averages 4055 images per class, 5 classes have less than 1500 images (*High Renaissance, Pointillism, Fauvism, Pop Art, Contemporary Realism*) and the median class size is of 2695. This is one of the reasons why training a network on the modified `Wikiart` dataset is such a difficult task.

6.1.2 Network training experimentation

Using pretrained models is a way to go around the aforementioned issue as the network has already been trained on a much bigger dataset and is able to recognise *coarse* features, retraining deeper layers allows us to fine-tune the algorithm on specific features of the dataset used. As per [11], an AlexNet model is used in which only the last layer of the network is retrained. In doing so, a maximum validation accuracy of 44.24% is achieved by training it during 50 epochs, with $\alpha = 0.001$, a batch size of 16, SGD as the optimizer and cross entropy as the loss function.

The increase in accuracy of the model when using a pretrained architecture is significant. After fine-tuning the model with other architectures such as VGG11, ResNet50, DenseNet or SqueezeNet, an optimal accuracy of 48.35% is obtained by using ResNet50. Using derivative networks of ResNet50 such as ResNet151, wide ResNet50 or ResNext does not show significant improvement and often results in a longer training time which is why the regular ResNet50 architecture is favoured. Furthermore, by using the two-phase ResNet50 architecture detailed in 5.2.3, we can see another accuracy boost with an 85.96% binary accuracy, 63.33% accuracy for artworks pre-1910s and 62.35% accuracy post-1910s resulting in a total accuracy of 53.96%.

As shown in the literature, retraining more than the last layer may be beneficial to increase accuracy. This might be due to the fact that the first layers of the network are used to recognise large features while deeper layers are used to recognise finer features. Hence, with the optimal architecture and parameters determined, we can start retraining the network. To get the best possible results, a strict protocol is followed to understand how many layers need to be frozen. The network is first retrained from scratch and then five layers are gradually frozen at each iteration from the first layer to the last. Once the interval in which the best accuracy is located is determined, the network is retrained by repeating the process in steps of 1 instead of steps of 5. The results show that the optimal accuracy of 61.80% is obtained by retraining 32 layers with a binary classifier accuracy of 88.83%, pre-1910s accuracy of 70.18% and post-1910s accuracy of 69.12%. Fig. 14 shows the accuracy of the network with respect to the number of retrained layers.

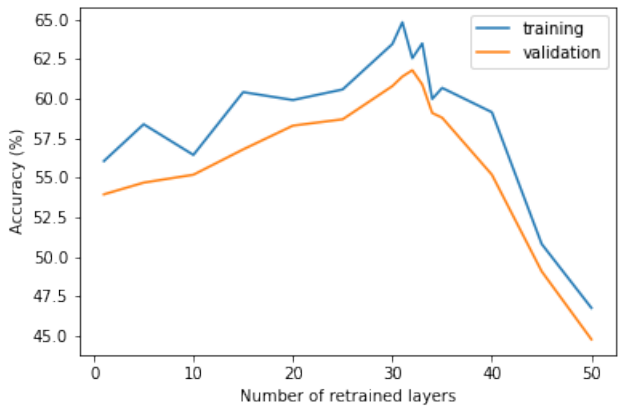


Fig. 14: Network accuracy with respect to the number of retrained layers.

As explained in [8], we tried adding two fully connected layers at the end of the network in order to reduce the number of nodes and better generalise the model. However, the network's performance does not improve so this methodology is not chosen as it increases training time unnecessarily.

6.1.3 Bagging and data augmentation

To further increase accuracy, two techniques are implemented: bagging and data augmentation. As mentioned in [57], the bagging method tends to increase the accuracy of a ResNet model as it reduces over-fitting. Also known as *bootstrap aggregation*, bagging basically averages out the results from different predictions outputted by the model by using several smaller datasets. By applying it, the accuracy increase is of about 1%.

Data augmentation is also used in order to obtain bigger diversity in the images to allow for a more robust model that tends to generalize better. Contrary to bagging which takes place after the training procedure, data augmentation takes place during. The augmentation used here consists of horizontal image flips, horizontal and vertical transitions as well as rotations. Distortions are not implemented as they prove to decrease accuracy as explained in 6.1.1. The results in Tab. 2 show that Data Augmentation is directly implemented as it cannot have a negative impact on the accuracy.

Table 2: Optimising network accuracy and performance with fine-tuning and other experiments.

Image	Model	Optimizer	Retrained layers	Bagging	Data Augmentation	Accuracy
Blank	ResNet50	SGD	1	No	Yes	46.34%
Compressed	ResNet50	SGD	1	No	Yes	47.55%
Cropped	ResNet50	SGD	1	No	Yes	48.35%
–	Alexnet	SGD	1	No	Yes	44.24%
–	Squeezenet	SGD	1	No	Yes	46.64%
–	Densenet	SGD	1	No	Yes	47.70%
–	VGG11	SGD	1	No	Yes	48.17%
–	ResNet50	SGD	1	No	Yes	48.35%
–	**Two-phase ResNet50**	SGD	1	No	Yes	53.96%
–	–	Adam	1	No	Yes	52.27%
–	–	RMSprop	1	No	Yes	52.88%
–	–	**SGD**	1	No	Yes	53.96%
–	–	–	**32**	No	Yes	61.80%
–	–	–	–	**Yes**	**Yes**	**62.56**%

6.2 Obtaining GAN stability and optimal convergence

6.2.1 The use of WGANs with Gradient Penalty

As demonstrated by [14], WGANs bring significant improvements to model stability and convergence. However, the weight clipping method used in this initial formulation enforces a Lipschitz constraint on the critic which may cause undesired behaviours of the network. Hence, the developed network in this project is based on the initial WGAN formulation but with Gradient Penalty instead of Weight Clipping as demonstrated by [5]. The advantages of gradient penalty over other methods can be seen in Fig. 15 which highlights the Inception Score of the model over Generator iterations and wall-clock time when using `CIFAR-10`.

Not only does Gradient Penalty demonstrate an increase in Inception Score of the network, it also reduces the probability of mode collapse as it mitigates for the common gradient vanishing or exploding that other methods are subject to. By not suffering form those, the generated outputs are often times more realistic. Both methods were implemented and although hardly comparable as such due to the nature of GANs, the outputs seem to have better structure and more defined features when using gradient penalty over weight clipping. It is worth noting that the gradient penalty term is arbitrarily set to 10. However, with more time and computational resources, testing out different values of that penalty term would have been interesting.

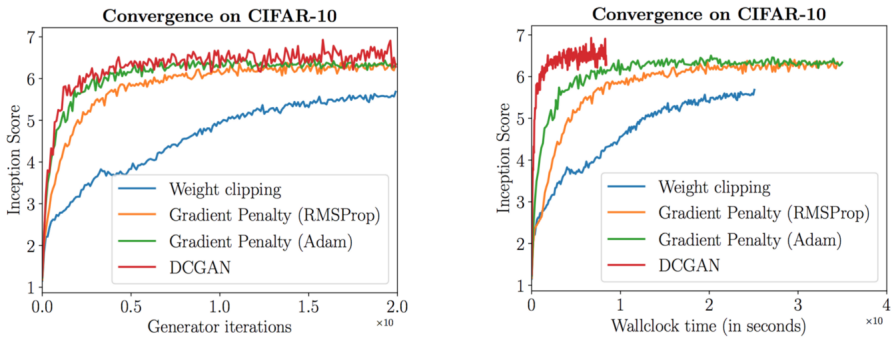


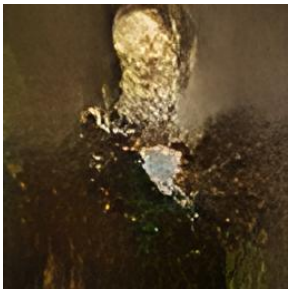
Fig. 15: IS of the model over Generator iterations and wall-clock time on `CIFAR-10` [5].

6.2.2 Auxiliary Classifiers over cGANs

Previously, we mention the use of cGANs as a way to condition our model on different labels in order to generate a specific category of images. However, our architecture uses Auxiliary Classified GANs instead, an extension of the common cGAN architecture developed by [58]. This architecture not only uses a conditioning vector in the Generator but also leverages the use of a classifier in the Discriminator that we now call Critic. The model aims to minimise the multi-label classification error as well as the regular binary (real/fake) objective of a GAN. The proposed work proves that using this conditional architecture increases synthesised image discernibility and diversity as well as reduces over-fitting.

As shown by Eq. 27, the objective of [5] is modified twice. On the one hand, a penalising term is added to the Discriminator's cost function which aims to minimise the cross-entropy of the prediction of synthesised class vs. real class. On the other hand, a penalising term is also added to the Generator to minimise the cross-entropy of the Discriminator's prediction against the class it has been conditioned to generate. Fig. 16 shows images generated by a regular cGAN (Fig. 16a) and by an ACGAN (Fig. 16b).

Those two outputs are chosen as being the most representative of each set and show that the implementation of the auxiliary classifier renders better face structure and improved model stability compared to the original cGAN model.



(a) cGAN

(b) ACGAN

Fig. 16: Portraits in the style of Realism generated by different conditional models.

6.2.3 Other methods to increase performance

Firstly, global conditioning is introduced to the model to allow for more complex and robust conditioning. Here, we use Conditional PixelCNN [6] as the nonlinear layer in the Generator. This methodology uses the conditioning vector of the ACGAN as both an addition to the initial noise vector of the Generator but also as a condition for each activation function in the network. The Conditional PixelCNN architecture uses gated multiplicative activation functions which seems to improve the convergence of the model as shown by initial results. Theoretically speaking, following the proof in [6], by considering $\mathbf{c}$ as a latent vector of the Generator's condition, we can model the conditional distribution of images by using the conditional PixelCNN formulation as follows:

$$p(\mathbf{x}|\mathbf{c}) = \prod_{i=1}^{n^2} p(x_i|x_1, \cdots, x_{i-1}, \mathbf{c}) \tag{34}$$

Given Eq. 34 and by using the initial formulation for a gated activation unit, we can replace the ReLU nonlinearities between the masked convolutions in the Generator by the following:

$$\mathbf{y} = \tanh(W_{k,f} * \mathbf{x} + V_{k,f}^T \mathbf{c}) \odot \sigma(W_{k,g} * \mathbf{x} + V_{k,g}^T \mathbf{c}) \tag{35}$$

where k is the layer number, $\mathbf{c}$ is one-hot conditioning vector, σ is the sigmoid non-linearity. In simple terms, using global conditioning in the network enables us to leverage the conditioning vector as a means to influence the layer's activation. Indeed, its gated multiplicative activation means that it may act differently depending on the layer of the network. This is more clearly illustrated by the diagram in Fig. 17.

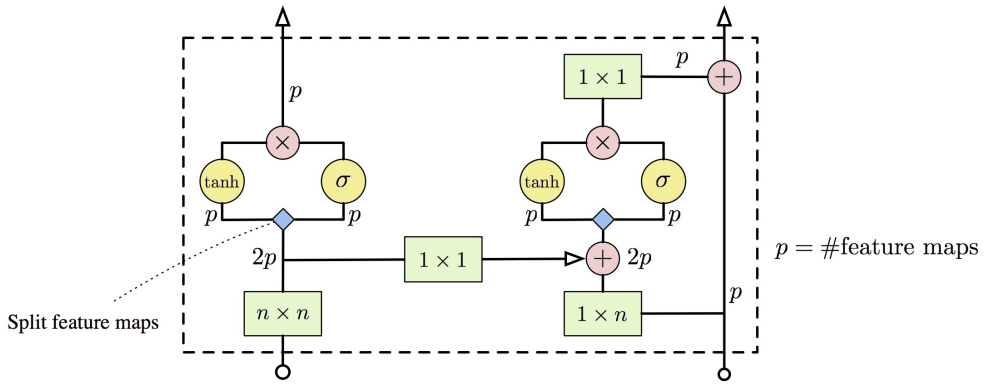


Fig. 17: Single layer of the Gated PixelCNN architecture developed in [6].

Secondly, the lack of initial label differentiation in the Generator's outputs leads us to hypothesise that the network needs training prior to any generation to create more diversity in its outputs. Similar to the image classifier, using pre-trained architectures drastically increases accuracy [59] which is why this methodology is implemented here.

Similar to pertaining, the ACGAN architecture differs mostly because of its use of Critic instead of the regular Discriminator where the classifier is trained n_c times for each Generator's iteration. Initially set to $n_c = 5$, experiments show that increasing that number improves generated output. However, increasing the number of critic iterations also increases training time. Hence, setting $n_c = 10$ is found to be a good middle ground.

Finally, batch normalisation is performed after each activation layer of the Generator. Doing so reduces over-fitting and has somewhat of a *regularization* effect as it inhibits any activation to take an extreme (high or low) value. However, batch normalisation is not used in the Critic as it would change the Discriminator's problem from a one-to-one problem to a many-to-many problem as explained in [14].

6.3 A fully functional pipeline

6.3.1 The last step in generated art: super resolution.

The Super-Resolution model is developed to obtain more clear outputs as 64×64 was deemed too small for visual appeal and discernment. To successfully double image resolution, the model needs to be trained on data which is four times bigger. Initially, the model is trained only with high-dimensional images from the `Wikiart` dataset. However, in order to increase the resolution and after witnessing the noticeable improvements of the CNN when using pretrained models, the SRGAN is pretrained on very high resolution images of hand-picked textures from Unsplash. Doing this noticeably improves the quality of the images as shown in Fig. 18.

(a) Low resolution original
Dim: (64×64)

(b) SR with no pretraining
Dim: (256×256)

(c) SR with pretraining
Dim: (256×256)

Fig. 18: Original and Super Resolution results of a portrait by Zinaida Serebriakova.

6.3.2 Limitations

One limitation of our model similar to that described in [58] is that increasing the Generator's resolution tend to decrease visual discriminability. This is at the root of the mode collapse which occurs when generating 128×128 outputs or bigger, limiting us to generate 64×64 images and forcing us to use a SRGAN.

Another limitation of the model is the fact that the ACGAN segment of the framework outputs better results for works generated by conditioning on subject matter within a specific style rather than works generated by conditioning on style as a whole. This might be due to the variety of

genres than exist within a single movement, explaining why most works generated by conditioning on style will output a mostly abstract images. By conditioning on style, the model is trained on portraits, landscapes, cityscapes or still lives which creates more feature maps.

6.4 A (not so) predictive model

6.4.1 A simple Encoder for latent space embedding

The main idea behind the predictive model is to compress the data into a latent space and use the low-dimensional representation as a means to infer on the *next* cluster in the sequence. Hence, a simple three-layer Convolutional AutoEncoder is trained on our data set and fine-tuned to reach a 10-epoch accuracy of 90.03% on training data and 89.71% on validation data. The developed model embeds the data so that it has 100 features. Fig. 19 shows the results obtained from applying the AutoEncoder to images from the dataset. The results are visually very strong which is mostly due to the relatively high number of features to input image dimension ratio.

Fig. 19: AutoEncoder input (above) and reconstructed output (below).

Once the Encoder's parameters are fully trained and fine-tuned, it is used to predict the latent codes of the images in the test set. The codes are saved for later use in the predictive model.

6.4.2 *Future* art prediction and generation

With latent codes ($\mathbf{c}$) saved from the trained Encoder, we now have $k = [0, \cdots, K]^T$ where K is the number of style labels (14) and $C = [\mathbf{c}_1, \cdots, \mathbf{c}_N] \in \mathbb{R}^{N \times d_c}$ where N is the number of images the Encoder is trained on (about 8000). A simple linear regression model is used, such that $\mathbf{y} = C\beta + \epsilon$, where $\epsilon \sim N(0, \sigma^2 I)$ with β and σ^2 being the linear parameters and I the identity matrix.

To generate art, a basic cGAN model conditioned on style is trained on the dataset. By leveraging the methodology in [21], the defined linear predictive model is coupled with a sparse Variational AutoRegressor applied to the means of the latent vectors $(\hat{\mu}^{(1)}, \cdots, \hat{\mu}^{(K)})$ of the different movements. Doing this results in the predicted mean $\tilde{\mu}^{(K+1)}$ and enables us to compute the predicted covariance $\tilde{\sum}^{(K+1)}$ as the mean of the different movements' covariance which in turn makes it possible to generate a *future* latent vector sampled from $N(\tilde{\mu}^{(K+1)}, \tilde{\sum}^{(K+1)})$. The sampled latent codes are then used to condition the cGAN's Generator which enables us to produce works from an unobserved probability and data distribution.

As explained, although we can technically generate *future* art as being the next movement in our sequence of 14 styles, there is little quantitative validation that can be inferred from such an output. Hence, to validate the model, we rather *predict* an existing style and compare it to the existing one. Fig. 20 shows three images, namely Realism, Predicted Realism and Future Art.

(a) Landscape in the style of Realism generated by the ACWGAN-GP

(b) Landscape in the style of Realism predicted by the cGAN

(c) Future art predicted by the cGAN

Fig. 20: Image generation comparison between ACWGAN-GP and cGAN.

Although not exactly our proposed architecture, the linear regression model does provide a good baseline to prove that predictive image generation is feasible. Compared to actual generation of

Realism with the original model, the predictive network is very blurry and lacks structure but does contain similar colours. Moreover, the Future Art images demonstrate highly abstract art with very little features and only a few specks of colour which seems to be in line with the current stylistic evolution of art. The model is still being developed as this paper is being written and very little fine-tuning has been conducted which may explain the low quality of outputs.

6.5 Quantitative validation of the outputs

6.5.1 Applying the classifier

An essential quantitative validation of the GAN comes from our state-of-the-art classifier. After generating some outputs, the SRGAN model is used to upscale the outputs to 224×224 so that they may then be fed to the classifier. Our model performs best when conditioned on subject matter within a specific movement. Considering the high computational requirements of generating outputs and the fact that conditioning on style alone renders results of lesser quality, it is decided to generate art from four movements, two pre-1910s and two post-1910s, namely: Realism, Impressionism, Abstract Expressionism and Minimalism. Each movement contains a single batch of 84 generated images. The results of this experiment can be viewed in Tab. 3.

Table 3: Results of applying the CNN to generated art.

Accuracy	Realism	Impressionism	Abstract Expressionism	Minimalism
Binary	80.95%	82.14%	79.76%	83.33%
Pre-1910s	19.05%	16.67%	-	-
Post-1910s	-	-	22.62%	23.81%
Total	15.42%	13.69%	18.04%	19.84%

The relatively underwhelming results when compared to the CNN's performance on the `Wikiart` dataset are somewhat expected for several reasons. Firstly, the binary classifiers' accuracy being relatively high may be due to the fact that pre-1910 works have much less colour and usually more details than post-1910 ones, making it relatively easy for the algorithm to make the distinction. However, when trying to determine the exact movement from which the art is generated, the lack of details makes it much more complicated to distinguish. Indeed, the level of detail of the works is

determined during the generation of the 64×64 images; the SRGAN can only increase resolution, not detail. Nonetheless, given that random accuracy is of 1/14 or 7.14%, our model still performs over twice as well as random choice.

6.5.2 Feature extraction and visualization

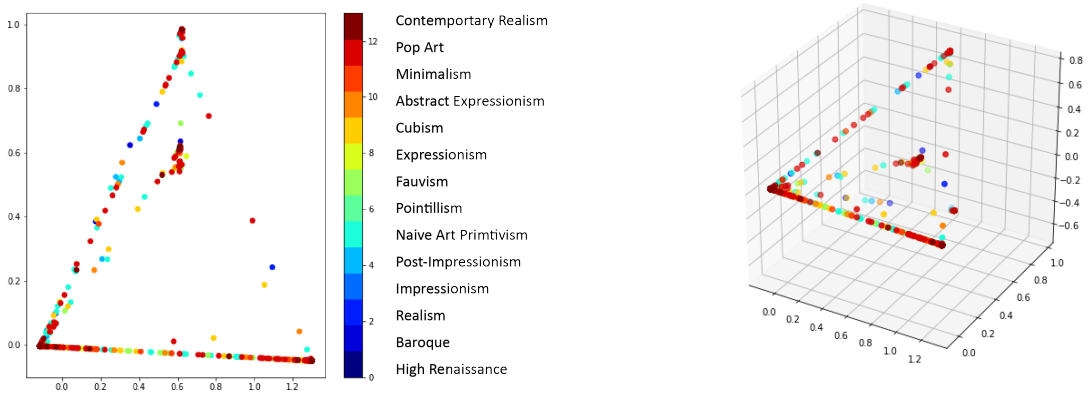
To get around the obstacle created by the limited generation size of our model which causes the CNN to under-perform, the model is further validated through data visualization. The idea here is to train a classifier on our dataset (with 64×64 images) in order to obtain its features. A VGG-16 architecture pre-trained on Tiny ImageNet [5] is fine-tuned on the 14 classes developed for our framework. Then, dimensionality reduction techniques are used to visualize the features of the images determined by the CNN (and display clusters in the data). In doing so, two methodologies are adopted: PCA and t-SNE. On the one hand, PCA or Principle Component Analysis is a technique which projects data points onto only the first few principal components (which are often eigenvectors of the data's covariance matrix) with the aim of obtaining lower-dimensional data that preserves variance as much as possible. On the other hand, t-SNE or t-distributed Stochastic Neighbor Embedding is a non-linear statistical method that attributes to each datapoint a coordinate in a two or three-dimensional map in order to visualize it.

Both dimensionality-reduction techniques are applied to a reduced dataset with 75 samples of each class (resulting in 1050 images) to visualize the classifier's features in both 2D and 3D-space. It should be noted that movements are labeled 0 to 13 in ascending order of date as per Tab. 1.

Firstly, Fig. 21 displays results from the PCA analysis. The two visualizations do not demonstrate any organisation within the features of the data. Indeed, the features of the classifier seem to be placed along the edges of a tetrahedron. Although movements towards the end of the sequence appear to cluster at the vertices, no tangible interpretation can be made with the given results. The PCA decomposition results in the variance being distributed as follows: 91.73% for PC-1, 4.09% for PC-2 and 1.76% for PC-3.

Secondly, Fig. 22 displays results from t-SNE analysis. Both visualizations are much more promising that that of the PCA. On the one hand, Fig. 22a shows that the VGG16-features are effectively clustered based on style. The superimposed generated works from Expressionism (index 6) and Baroque movement (index 1) are presented respectively as circles and triangles. The embedding

[5] `https://paperswithcode.com/dataset/tiny-imagenet`



(a) 2D-space visualization

(b) 3D-space visualization

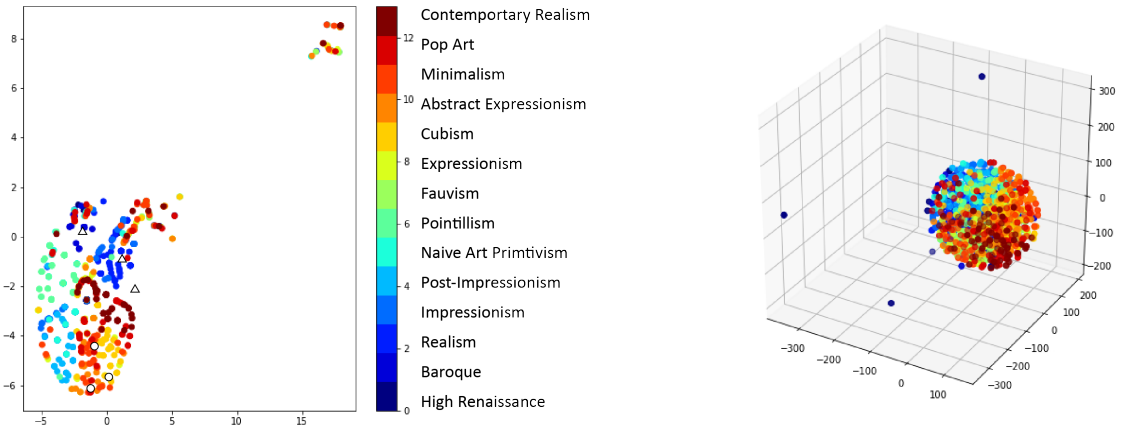
Fig. 21: PCA on VGG16-features.

demonstrates that the style-specific generated works are mostly part of their attributed cluster which is tangible quantitative validation of the model. On the other hand, Fig. 22b unveils another insight in the 3D-space. Although the features mostly overlap, this visualization demonstrates that earlier movements pre-1910s (0-7) are closer to one another while later movements post 1910s (8-13) are closer to one another. This validates the use of an initial binary classifier in the CNN architecture and also serves as an interesting discussion point, showing that the early XXth century represents a true turning point in stylistic representation.

The inferior results obtained with PCA compared to t-SNE may be attributed to the fact that our dataset is very high-dimensional and cannot be linearly reduced by PCA. We argue that no-linear methods like t-SNE (that minimises KL divergence) are needed to achieve better results.

6.6 Qualitative validation through a holistic method

Due to the inherent subjective nature of art, it is deemed necessary to adopt both a quantitative but also a qualitative human-centric validation. A survey of 5 experiments inspired by [10] was taken by 82 people who are not art connoisseurs as this would give them an unfair advantage and inherent bias. A question about the artistic knowledge of the trialed subjects should have been added before starting the experiments but was unfortunately omitted.



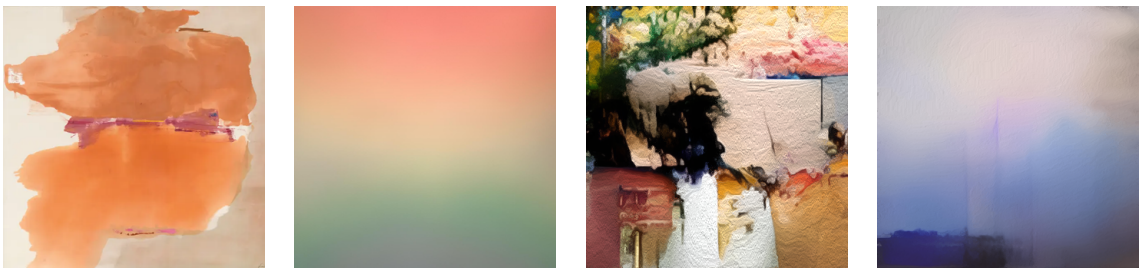
(a) 2D-space visualization

(b) 3D-space visualization

Fig. 22: t-SNE on VGG16-features.

6.6.1 Experiment n°1

Experiment n°1 aims at understanding whether subjects are capable of recognizing machine generated art from art created by artists on a case to case basis. Subjects are shown four images independently and asked whether they are AI-generated or created by an artist. Following that, viewers are asked to score the painting on a scale of 1 to 5. Out of the four presented works, two are generated by the MichelGANgelo model and two are painted by actual artists. However, this information was not given to the subjects in order to give them full freedom of choice. The chosen works are abstract and shown in Fig 23.



(a) *Sphynx 1976*, Helen Frankenthaler

(b) *Series 12*, Leon Berkowitz

(c) *Composition n°XI*, MichelGANgelo

(d) *Composition n°VIII*, MichelGANgelo

Fig. 23: Works presented in experiment n°1.

The results show that the generative model is successful in fooling people. As shown in Tab. 4, although most people correctly identify Frankenthaler's painting as a human creation, the painting that most subjects thought to be AI-generated is actually made by a human. Concerning the works generated by MichelGANgelo, the results show that subjects' opinions are very divided as both works are correctly classified 50.0% and 65.9% of the time for Fig. 23d and Fig. 23c respectively. It is also interesting to see that the subject's favourite work is AI-generated while their least favourite one is created by a human. Moreover, only 15 participants (or 18.5% of people) managed a perfect score which underlines the difficulty of the task at hand.

Table 4: Results from experiment n°1.

Painting	Artist	Correct	Avg Score	Stdev	Med Score
Sphynx 1976	Frankenthaler	81.7%	3.23	1.07	3
Series 12	Berkowitz	32.9%	2.60	0.99	2
Composition n°XI	MichelGANgelo	65.9%	3.38	1.14	3
Composition n°VIII	MichelGANgelo	50.0%	3.52	0.92	4

6.6.2 Experiment n°2

Experiment n°2 is similar to the first in that subjects are asked to identify the machine generated artwork from four sets of three images in which two are painted by an artist and one is generated by the model. This task should be less challenging for the subjects as they are able to compare the works between each other to make an informed decision.

As expected, the results demonstrate that subjects are mostly able to differentiate machine generated works from actual paintings when presented as such. However, as highlighted in the previous experiment, the abstract works in Fig. 24a are the hardest to differentiate as only 54.9% of viewers correctly identify the generated one. This differs from the other sets of paintings which contain an actual subject matter (portrait Fig. 24b, landscape Fig. 24c or Fig. still life 24d) as results show correct differentiation of 78.0% for portraits, 73.2% for landscapes and 73.2% for flowers. This is still remarkable as the actual paintings have been created by some of the most notable modern artists. Results are displayed in Tab. 5. Interestingly, only 33 subjects (or 40.7% of participants) managed a perfect score. Furthermore, subjects are also asked to attribute a differentiation difficulty score

(a) C. Still (left & middle), MichelGANgelo (right).

(b) F. Bacon (left & right), MichelGANgelo (middle).

(c) E. Degas (middle), C. Pissarro (right), MichelGANgelo (left).

(d) K. Kovalenko (middle & right), MichelGANgelo (left).

Fig. 24: Works presented in experiment n°2.

based on their choice. Results show that abstract art and flowers are hard to differentiate while portraits and landscapes are subjectively easier. This might be due to the lack of structure and detail of the generated work compared to the very defined paintings from actual artists. Finally, users are asked if they consider all three works belong to the same movement. Once again, as expected, the more abstract representations have better scores than the more figurative ones.

Table 5: Results from experiment n°2, each set of images is a row of Fig. 24 from top to bottom.

Set	Genre	Artist	Generated	Correct	Differentiation Difficulty	Same Mvt
1	Abstract	C. Still	Right	54.9%	3.62 (0.98)	70.7%
2	Portrait	F. Bacon	Middle	78.0%	2.87 (1.12)	53.7%
3	Landscape	E. Degas & C. Pissarro	Left	73.2%	2.85 (1.28)	57.3%
4	Flowers	K. Kovalenko	Left	73.2%	3.07 (1.20)	67.1%

6.6.3 Experiment n°3

The third experiment aims at understanding whether subjects are able to distinguish subject matter in a series of generated images. In this task, the viewers are being presented with three sets of three images which are all portraits, landscapes and flowers respectively. The results show that subjects are constantly able to identify subject matter in a generated work. Indeed, 100% of viewers correctly identified flowers and landscape paintings while 98.8% of subjects correctly identify the portraits; only one person labelled the set of portraits as *abstract*. The set of images presented to the subjects can be seen in Fig. 25.

6.6.4 Experiment n°4

The fourth experiment aims at better understanding the quality of the generated batches by the MichelGANgelo framework. Two batches of 84 paintings of dimensions 64×64 are presented to the subjects, one composed of still life paintings (mostly flowers) shown in Fig. 26 and the other composed of generated portraits.

In both cases, all subjects except one clearly identified the genre of the generated batch. The quality of a GAN can often times be assessed by its generated outputs over a batch. Certain criterion such as diversity, structure, blurriness or artefacts enable to determine whether the model is subjected to mode collapse. The relatively high scores given by the subjects in Tab. 6 show

(a) Portraits generated by MichelGANgelo.

(b) Landscapes generated by MichelGANgelo.

(c) Flowers generated by MichelGANgelo.

Fig. 25: Works presented in experiment n°3.

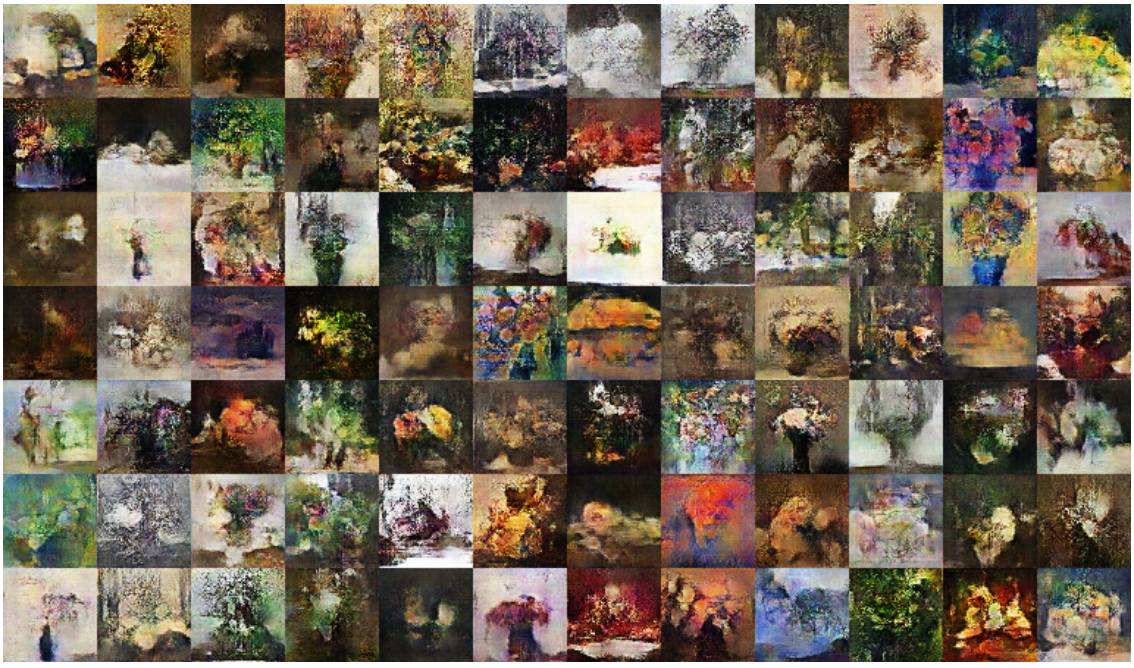


Fig. 26: Batch of generated flower paintings from the Realism movement.

that the MichelGANgelo model is effective and has good performance. An issue here would be the blurriness/artefact score which demonstrates that the outputs have not fully converged and would require additional details to improve.

Table 6: Results from experiment n°4 (σ in parenthesis).

Batch	Genre	Correct	Diversity	Structure	Artefacts	Overall Score
1	Flowers	98.8%	3.56 (1.13)	3.33 (1.08)	3.37 (1.17)	3.78 (0.99)
2	Portrait	98.8%	3.78 (1.13)	3.30 (0.99)	3.78 (1.09)	3.56 (0.97)

6.6.5 Experiment n°5

The fifth and final experiment of this survey consists in a subjective emotional test to understand how one of our generated works (*Composition n°XI*, 2022) shown in Fig. 23c compares to the work of renowned contemporary artist Eddie Martinez (*Christmas in July*, 2016) shown in Fig. 27.

Fig. 27: *Christmas in July*, Eddie Martinez, 2016

The subjects are tasked to rate how much they agree or disagree to a set of adjectives that describe the canvas. The 10 words presented are: interesting, creative, intellectually stimulating, aesthetically appealing, novel, ambiguous, skillfully-made, compelling, inspiring and complex.
The results in Tab. 7 show that although quite similar, the generated artwork seems to be more popular than the one created by a renowned artist. Already, by computing the average score, *Composition n°XI* has an average score of 3.42 with a standard deviation of 0.33 and a median score of 4 while *Christmas in July* obtains an average score of 3.37 with a standard deviation of 0.40 and a median score of 3. By diving a bit deeper into the data, most adjectives tend to have a similar score for both works except for two which have a variation of more than 5%. Those two adjectives are *aesthetically pleasing* and *novel* which score respectively 13.28% and 8.30% more for the AI-generated work than for the artist created one. The only adjective that has a noticeably lower score for our generated work would be *creativity* which scores -6.63%. This is not too surprising as we still expect human inventiveness to be more creative than that of a computer. Nonetheless, those results are still remarkable and show how close MichelGANgelo's generated works are to that of an acknowledged contemporary artist.

Table 7: Results from experiment n°5.

center

Artist	Interesting	Creative	Stimulating	Aesthetic	Novel	Ambiguous	Skillful	Compelling	Inspiring	Complex
Human	3.84	4.05	3.22	2.94	3.23	3.72	3.17	3.15	2.85	3.50
AI	3.83	3.78	3.21	3.33	3.50	3.87	3.22	3.21	2.83	3.41
Var.	-0.32%	-6.63%	-0.38%	13.28%	8.30%	3.93%	1.54%	1.94%	-0.85%	-2.44%

7 Conclusion

7.1 Summary

This paper effectively develops a novel deep generative framework coined MichelGANgelo with the aim of creating art from existing styles (and genres) as well as *future* predicted ones. Moreover, a state-of-the-art classifier is developed as being part of the quantitative validation of the framework and offers novel insights into the links which exist between movements and the way style has evolved through time. Apart from the aforementioned quantitative validation, a series of qualitative and highly subjective experiments have been performed on several participants. This holistic survey, shows that the generated art is often times very complicated to differentiate from actual art and is considered by surveyed users as highly aesthetically appealing and creative.

The task of generating novel art from *scratch* has proven to be more complex than that of style transfer for instance, due to the high number of factors that come into it. All in all, the framework implements advanced deep learning concepts for image generation such as Convolutional Neural Networks, Generative Adversarial Networks or AutoEncoders as well as elaborate mathematical concepts such as Optimal Transport or Super-Resolution loss functions.

7.2 Future work

Despite the overall successful output of this project, a number of limitations inherently require future work to be conducted. Firstly, the developed ACWGAN-GP is able to generate outputs of dimension 64×64 but suffers mode collapse when increasing image size to 128×128 or bigger. Hence, this results in having to use a SRGAN to increase image resolution by a factor of four (to effectively pass them through our ResNet-based classifier architecture) which reduces the quality of feature extraction and causes the state-of-the-art CNN to under-perform. Being able to obtain superior model convergence and stability when increasing generation size is definitely required for both better quality results and CNN performance. Secondly, the compression method used to embed data in the latent space is a simple AutoEncoder. However, it is assumed that leveraging a Variational AutoEncoder will increase encoding accuracy as it should enhance feature extraction and make disentangled representations more exploitable. Thirdly, replacing the linear prediction model to determine *future* art probability distributions with a geodesic interpolation model based

on Optimal Transport should refine the quality of the forecasted latent vectors. We argue that the high-dimensionality of the data cannot be correctly interpreted linearly which is why results are sub-optimal. Moreover, fine-tuning the cGAN architecture and implementing computational Optimal Transport techniques such as Wasserstein-based loss functions in the Generator may also prove beneficial for optimal model convergence.

References

[1] Dongsheng An, Yang Guo, Min Zhang, Xin Qi, Na Lei, Shing-Tung Yau, and Xianfeng Gu. AE-OT-GAN: Training GANs from data specific latent distribution. 1 2020.

[2] Tariq Daouda, Reda Chhaibi, Prudencio Tossou, and Alexandra-Chloe Villani. Auto-encoders with fibered latent spaces: A geometric approach to batch correction. Technical report, 2020.

[3] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. 9 2016.

[4] Sarthak Bhagat, Shagun Uppal, Zhuyun Yin, and Nengli Lim. Disentangling Multiple Features in Video Sequences using Gaussian Processes in Variational Autoencoders. 1 2020.

[5] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. 3 2017.

[6] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. 6 2016.

[7] Iria Santos, Luz Castro, Nereida Rodriguez-Fernandez, Álvaro Torrente-Patiño, and Adrián Carballal. Artificial Neural Networks and Deep Learning in the Visual Arts: a review, 1 2021.

[8] Ahmed Elgammal, Marian Mazzone, Bingchen Liu, Diana Kim, and Mohamed Elhoseiny. The Shape of Art History in the Eyes of the Machine. 1 2018.

[9] Wei Ren Tan, Chee Seng Chan, Hernan Aguirre, and Kiyoshi Tanaka. Improved ArtGAN for Conditional Synthesis of Natural Image and Artwork. 8 2017.

[10] Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. CAN: Creative Adversarial Networks, Generating "Art" by Learning About Styles and Deviating from Style Norms. 6 2017.

[11] Eva Cetinic, Tomislav Lipic, and Sonja Grgic. Fine-tuning Convolutional Neural Networks for fine art classification. *Expert Systems with Applications*, 114:107–118, 12 2018.

[12] Wei Ren Tan, Chee Seng Chan, Hernán E Aguirre, and Kiyoshi Tanaka. Ceci n'est pas une pipe: A Deep Convolutional Network for Fine-art Paintings Classification. Technical report.

[13] Adrian Lecoutre, Benjamin Negrevergne, Florian Yger, Yung-Kyun Noh, and Min-Ling Zhang. Recognizing Art Style Automatically in painting with deep learning. Technical report, 2017.

[14] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. 1 2017.

[15] Vaios Laschos, Jan Tinapp, and Klaus Obermayer. Training Generative Networks with general Optimal Transport distances. 10 2019.

[16] Na Lei, Kehua Su, Li Cui, Shing-Tung Yau, and David Xianfeng Gu. A Geometric View of Optimal Transportation and Generative Model. 10 2017.

[17] Gil Avraham, Yan Zuo, and Tom Drummond. Parallel Optimal Transport GAN *. Technical report.

[18] Wenju Xu, Chengjiang Long, Ruisheng Wang, and Guanghui Wang. DRB-GAN: A Dynamic ResBlock Generative Adversarial Network for Artistic Style Transfer. Technical report.

[19] Bingchen Liu, Kunpeng Song, Yizhe Zhu, and Ahmed Elgammal. Sketch-to-Art: Synthesizing Stylized Art Images From Sketches. Technical report.

[20] Reiichiro Nakano. Neural Painters: A learned differentiable constraint for generating brushstroke paintings. 4 2019.

[21] Edoardo Lisi, Mohammad Malekzadeh, Hamed Haddadi, F. Din-Houn Lau, and Seth Flaxman. Modeling and Forecasting Art Movements with CGANs. 6 2019.

[22] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. 12 2015.

[23] Y. Bengio et al. Learning deep architectures for AI. Foundations and trends R in Machine Learning. 2(1):1–127, 2009.

[24] I. Lajoie Y. Bengio P. Vincent, H. Larochelle and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, pages 3371–3408, 12 2010.

[25] K. Sohn X. Yan, J. Yang and H. Lee. Attribute2Image: Conditional image generation from visual attributes. *ArXiv e-prints*, 12 2015.

[26] Gaëtan Hadjeres, Frank Nielsen, and François Pachet. GLSR-VAE: Geodesic Latent Space Regularization for Variational AutoEncoder Architectures. 7 2017.

[27] Ilya Tolstikhin, Olivier Bousquet Google Brain Zürich, Sylvain Gelly Google Brain Zürich, and Bernhard Schölkopf. WASSERSTEIN AUTO-ENCODERS. Technical report.

[28] Paul K. Rubenstein, Bernhard Schoelkopf, and Ilya Tolstikhin. On the Latent Space of Wasserstein Auto-Encoders. 2 2018.

[29] Kart-Leong Lim, Xudong Jiang, and Chenyu Yi. Deep Clustering with Variational Autoencoder. Technical report.

[30] Paulino Cristovao, Hidemoto Nakada, Yusuke Tanimura, and Hideki Asoh. Generating In-Between Images through Learned Latent Space Representation Using Variational Autoencoders. *IEEE Access*, 8:149456–149467, 2020.

[31] Huidong Liu, Yang Guo, Na Lei, Zhixin Shu, Shing-Tung Yau, Dimitris Samaras, and Xianfeng Gu. Latent Space Optimal Transport for Generative Models. 9 2018.

[32] Yann Brenier. Polar Factorization and Monotone Rearrangement of Vector-Valued Functions. Technical report.

[33] Oliver Zhang, Ruei-Sung Lin, and Yuchuan Gou. Optimal Transport Based Generative Autoencoders. 10 2019.

[34] David Alvarez-Melis and Nicolò Fusi. Geometric Dataset Distances via Optimal Transport. 2 2020.

[35] Villani Cédric. Optimal transport, old and new. Technical report, 2008.

[36] Tianlin Xu, Li K Wenliang, Michael Munn Google, and Beatrice Acciaio. COT-GAN: Generating Sequential Data via Causal Optimal Transport. Technical report.

[37] Bowen Liu, Yu Chen, Shiyu Liu, and Hun-Seok Kim. Deep Learning in Latent Space for Video Prediction and Compression. Technical report.

[38] Ricardo Ñanculef, Petia Radeva, and Simone Balocco. Training Convolutional Nets to Detect Calcified Plaque in IVUS Sequences. In *Intravascular Ultrasound: From Acquisition to Advanced Quantitative Analysis*, pages 141–158. Elsevier, 1 2020.

[39] Gabrel Turinici. The convergence of the Stochastic Gradient Descent (SGD) : a self-contained proof. 3 2021.

[40] Sebastian Bock and Martin Weiß. A Proof of Local Convergence for the Adam Optimizer. Technical report.

[41] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. 6 2014.

[42] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. 11 2015.

[43] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. 11 2014.

[44] Pierre Baldi. Autoencoders, Unsupervised Learning, and Deep Architectures. Technical report, 2011.

[45] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. 3 2020.

[46] E. Fong and C. C. Holmes. On the marginal likelihood and cross-validation. *Biometrika*, 107(2):489–496, 6 2020.

[47] Gabriel Peyré and Marco Cuturi. Computational Optimal Transport. Technical report, 2017.

[48] Gaspard Monge. "Memoire sur la theorie des deblais et des remblais". *Histoire de l'Academie Royale des Sciences de Paris*, 1781.

[49] Matthew F Thorpe. Introduction to Optimal Transport. Technical report.

[50] Luis Caicedo Torres, Luiz Manella Pereira, and M. Hadi Amini. A Survey on Optimal Transport for Machine Learning: Theory and Applications. 6 2021.

[51] Justin Solomon, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains. Technical report.

[52] Tim Salimans, Han Zhang, Alec Radford OpenAI, and Dimitris Metaxas. Improving GANs using Optimal Transport. Technical report.

[53] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. 6 2016.

[54] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. 3 2016.

[55] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. 9 2016.

[56] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 2016.

[57] Tristan Cazenave. Residual Networks for Computer Go. *IEEE Transactions on Games*, 10(1):107–110, 3 2017.

[58] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. 10 2016.

[59] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. 5 2015.