Generating Coats-of-Arms with Generative Adversarial Networks

Ray MacDonncha (B.Comm)
Student ID: 08025479
School of Computer Science
National University of Ireland Galway

Supervisor
Prof. Michael Madden

In partial fulfilment of the requirements for the degree of
MSc in Computer Science (MAO2 Artificial Intelligence Online)
DEGREE DATE (August 2022)

DECLARATION I, Ray MacDonncha, do hereby declare that this thesis entitled 'Generating Coats of Arms with Generative Adversarial Networks is a bonafide record of research work done by me for the award of MSc in Computer Science (Artificial Intelligence - Online) from National University of Ireland Galway. It has not been previously submitted, in part or whole, to any university or institution for any degree, diploma, or other qualification.

Signature: Ray MacDonncha

Abstract

One of the most interesting concepts in data generation in recent years has been the advent of Generative Adversarial Networks, which have been able to produce highly-realistic recreations of data in a range of domains. This paper investigates the possibility of using one of these models to generate convincing samples of coats-of-arms, which would be a challenge due to their multimodal nature, and the fact that they typically contain both a mixture of abstract imagery and representations of real world objects.

We review a number of architectures which seem best suited to handle this task, and try to apply a selection of these to the problem at hand. We conclude that using either a clustered-conditional Wasserstein GAN or a StyleGan2-ADA model should result in convincing representations of coat-of-arms, given sufficient training data. We come to this conclusion by seeing convincing representations of the typical abstract imagery present in these coats-of-arms, and the emergence of some of the representations of the real-world imagery, despite having a small amount of training data.

Keywords: generative adversarial networks, coats-of-arms, StyleGAN

Contents

Chapter 1 Introduction

1.1 Background

The purpose of this project is to investigate whether it is possible to create computer-generated images of coats-of-arms using a class of deep learning algorithms known as Generative Adversarial Networks (GANs).

Before getting into GANs, it would be useful to briefly cover deep learning, of which GANs are a subset of. Deep learning is a category of machine learning which uses the human brain as inspiration for the design of its architecture, known as neural networks. These neural networks can be described as layers of nodes, through which data is passed through and which loosely represent how a human brain cell processes information [1]. As input data is passed through one of these nodes, it's transformed by that node's coefficients, or weights, and passed through to an activation function, which determines whether this node should be 'activated' ie) should the data be passed through this node to the following layer of nodes. This eventual output of the data as its passed through a sequence of nodes is evaluated using a cost function, which measures the differences between the predicted value and actual value and then the weights of the nodes are updated accordingly. Eventually, through many iterations (known as epochs), the combination of weights to be used and nodes to be activated which results in the lowest cost function is considered the trained model.

Neural networks are typically associated with classification tasks, though by no means is that their exclusive purpose. One of the use cases of neural networks and deep learning is computer vision, where the goal is to help computers 'see' in the sense of being able to process and understand the content of images via pattern recognition (Chen, 2015). Another use case for deep learning is to use neural networks for deep generative modelling, where the goal is to 'understand' and approximate complex, high-dimensional data distributions. GANs can be considered an example of both aforementioned branches of deep learning.

A GAN describes a class of deep learning algorithms used for generating data. Typically, they are composed of two neural networks: a generator which generates new samples of data, and a discriminator which classifies whether the data it takes as an input is a real sample or a generated sample from the generator. Depending on whether the discriminator has made a successful prediction, the weights of either the generator or discriminator are updated and this cycle continues with an end goal of the generator being able to create samples of data that are indistinguishable from samples from a real data distribution [2].

This competition between generator and discriminator is how the Generative Adversarial Network (GAN) is trained and over the course of many iterations can lead to generated data that is remarkably close to the distribution of real data, such as images of faces that are photorealistic, as can be seen on the website thispersondoesnotexist.com [3]. Here, a new photorealistic rendering of a

human face is created by a well trained generator from a GAN architecture every time the webpage is refreshed.

A blog post by Al Gharakhanian [4] provides an excellent illustration of the basic architecture of a GAN which can be seen below, and which shows the core components that have been mentioned above.

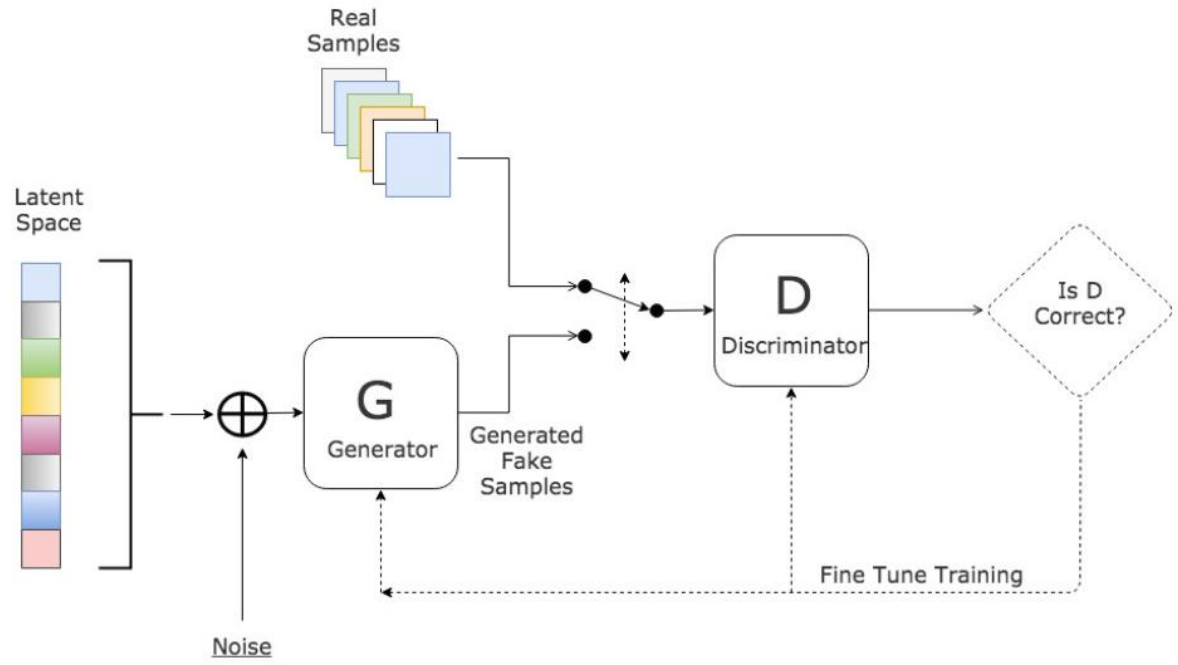


Figure 1: Overview of a GAN architecture [4]

GANs have also demonstrated an impressive ability to learn an underlying artistic style from input images, and either apply this artistic layer to the content of another image [5], or use this to generate completely new images mimicking this artistic style [6].

1.2 Motivation

While GANs have proven successful in generating new samples of data that have the same general structure, such as faces with a nose, mouth etc.., there are less examples of GANs generating data from a multi-modal distribution, where the features of the dataset are unlikely to be present in the majority of individual samples, and where the generated content must not only include abstract elements in the 'style' of the input dataset - it must also generate something that resembles or represents objects seen in the real world.

A coat-of-arms, or family crest, would then provide a challenge for a GAN to generate plausible new samples of. A coat-of-arms is typically a shield-shaped outline, but the content within this shape will be unique for each sample, and will include a finite but random assortment of images representing real world items such as swords, lions, helms, trees etc.. in a variety of number, size, shape and placement on coloured background that may or may not have a pattern of its own ie) chequered, striped etc.. [7]

The motivation for this proposed project is to see if a GAN can be made to generate plausible samples for such a challenging dataset, both as an exercise in computational creativity in and of itself, but also to potentially uncover any fundamental underlying challenges in GAN architecture that prevents it from doing so, which can then be used as a starting point for future work.

Chapter 2 Related Work

2.1 Search Method

My initial search was made on the James Hardiman Library online resource, followed by a second search on Google Scholar. In both cases, the terms I used in my search were the following, both individually and in combination with each other: ***gan, multi-modal data, crests, coat-of-arms, logos, generative adversarial network, stylegan, layers, discrete data generation, conditional gan, mode collapse.***

There was no past work to be found on generating family crest/coats-of-arms, with GANS or any other Machine Learning algorithm, but there was an abundance of papers on generating logos with GANS, so I made sure to include all of those as the problem will be similar in nature and the work done on those papers will definitely be useful.

In addition, I found some papers on generating samples from multi-modal data distributions using various GAN networks, such as generating handwritten characters or multiclass image generation with Conditional GANS. As I envision mode collapse being an issue for training a GAN of this nature, I have found some papers on how to address this issue.

2.2 Notes

GAN's are a relatively new and popular topic, which has had its benefits and drawbacks for a literature survey. On one hand, there is no shortage of papers on the topic, all of which are relatively recent (within the last 2-3 years). On the other hand, a sizeable portion of these aren't peer reviewed or published in journals. As such, I had no trouble in sourcing a large number of papers, but quite a lot of these articles had to be assessed for quality if they weren't published. That being said, I have still kept some of the more interesting and relevant sounding papers.

2.3 Similar work in different domains

2.3.1 Deep-Convolutional Generative Adversarial Network

A Deep-Convolutional Generative Adversarial Network (DCGAN) is one of the most popular and widely-used GAN architectures, and was first introduced in the paper "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" [8]. A DCGAN uses a deep convolutional network architecture which seeks areas of correlation within an image using convolutional stride as opposed to max pooling [8]. This integration of convolutional neural networks into the GAN architecture tend to make the DCGAN particularly fitting for generating natural image or video data samples [8], and as such seemed like a good starting point for examining the problem statement of this paper.

2.3.2 Least-Square GAN

Moving on from DCGAN - exploring the utility of Least-Square GANS (LSGAN) in this problem domain seems like a logical next step, as LGGAN's are very similar in architecture to a DCGAN, except that they adopt the least squares loss function for the discriminator. The reasoning behind this change of loss function is outlined in the paper in which LSGANs were introduced [9], where the authors assert that that using a least-squares method helps to solve for the vanishing gradient problem which can occur in regular GANs that use binary cross-entropy loss function (including DCGANs).

In addition, the authors state that LSGANs generate higher quality images that regular or DCGANs, which can be especially beneficial when it comes to the generation of the smaller, recognisable shapes and fine details contained within the crests. LSGANs are noted are more stable during the learning process too which will be helpful with regards to avoiding problems like mode collapse.

2.3.3 Wasserstein GAN

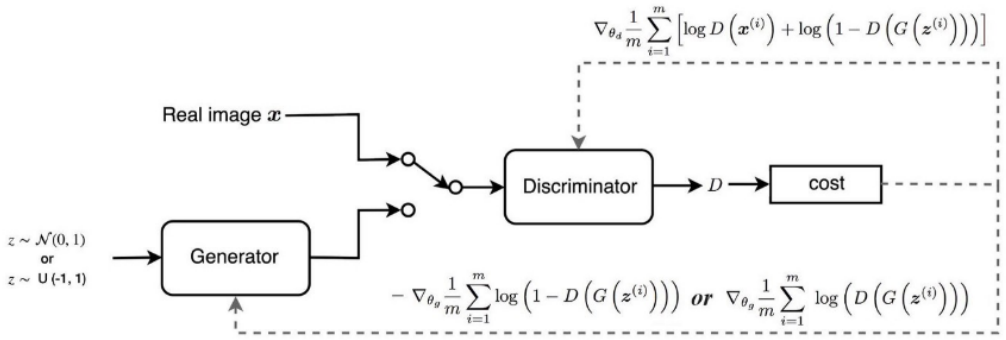
Mode collapse is one of the more prevalent issues when training to train a stable GAN model *(Chapter 2.5.1)*

There are many proposed solutions to address this problem, one of the most popular is to include noise to the GAN architecture to increase sampling diversity in the target distribution and thus hopefully reduce the likelihood of repeated sampling from the same area of the distribution. However, there have been some criticisms of this approach, include those by the authors of the WGAN paper [10] who argue that adding noise the GAN architecture results reduced image fidelity.

As an alternative solution, they propose using Wasserstein distance (or Earth-Mover distance) as a cost-function instead of binary-cross entropy or mean-sq error. The way that is works is that the discriminator in a classical GAN architecture is replaced by what is called a 'critic', whose role is to score the images produced by the generator based on their 'realness' as opposed to the aforementioned discriminator which would have provided a simple boolean response to whether or not it figured the image was real or generated.

As we can see from the below image the overall architecture of the WGAN is not too different from a regular GAN, barring these changes to the cost functions:

GAN:



WGAN

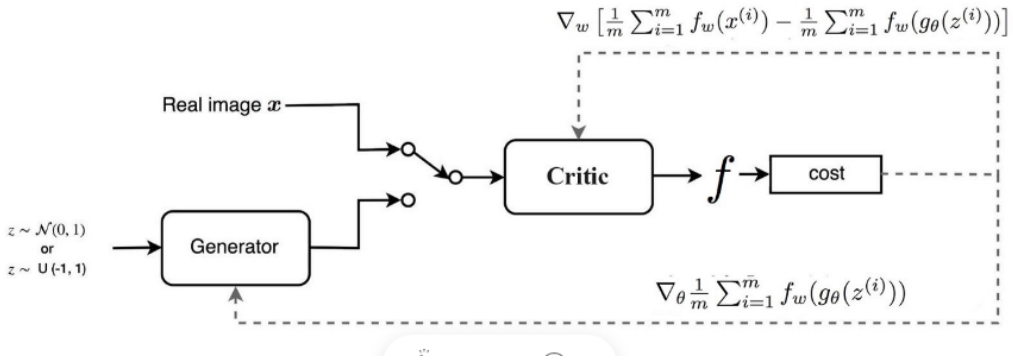


Figure 2: Comparison of GAN & WGAN architectures [11]

The authors argue that Wasserstein distance as a cost function is the most optimal measure to use to compare two probability distributions in a low-dimensional manifold space, and thus could be used to train the critic optimally. This results in smoother gradients and reduced mode collapse when combined with a Lipschitz constant to stop exploding gradients. In the case of a WGAN this is applied by using clipping to put a ceiling on the maximum weights of the critic.

The below image taken from this paper shows the smoother gradients associated with the WGAN in contrast with a regular GAN suffering from vanishing gradients:

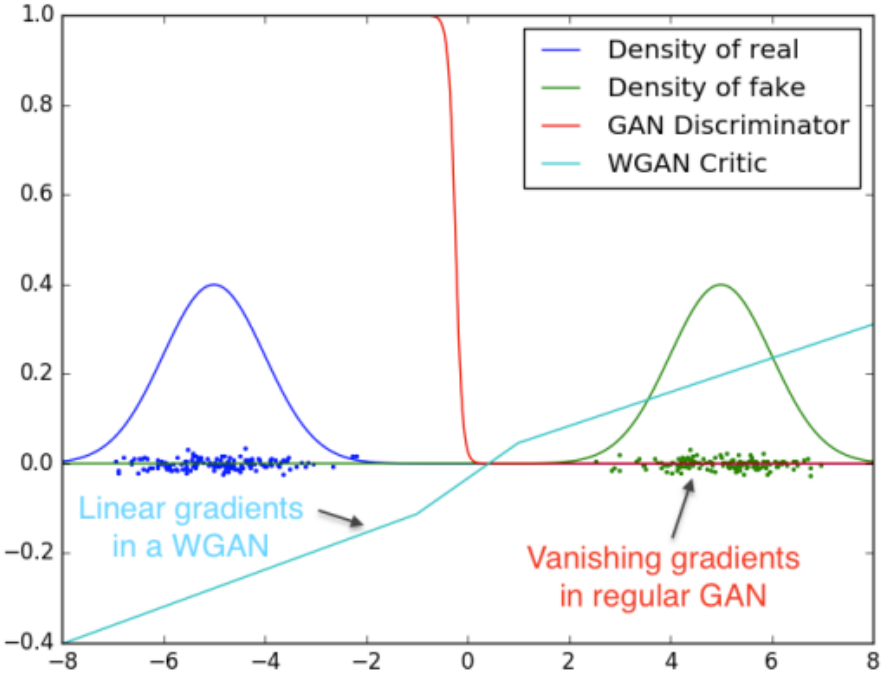


Figure 3: Comparing gradients regular GAN vs WGAN [10]

2.3.4 Conditional GAN

Conditional GANs (CGAN) were first mentioned in the paper 'Conditional generative adversarial nets' [12], where the authors highlight the lack of control a more traditional GAN architecture has on the number of modes its generator produces, and how it can be solved by conditioning the data on additional data, such as class labels, in order to exert more control on the data generation [12].

The proposed conditional CGAN model would involve both the generator and discriminator being conditioned on some sort of additional information representing one of the many modes within the data, which in turn would help map the generated data to one of the aforementioned modes [12].

The authors then demonstrate this concept on a multi-modal dataset by taking images from a website that have multiple user defined tags, then train the model on the image data and the associated tags, using the user defined tags as the as additional conditional data. The CGAN then generated its own labels describing the image data, with very large dictionary size of almost 250k unique words used to describe the image [12].

To evaluate the model, the authors generated a hundred tags for each image and then employed cosine similarity [12] to get the top twenty words for each sample. Cosine similarity is where the cosine angle between two vectors is used to as a measure of similarity [29]. In this instance, the vectors would represent the encoded images and text. In qualitative terms the results were very impressive, with the twenty generated tags doing a relatively good job of describing the image in most cases [12].

For our purposes, CGANs could be a good way of handling the multimodal nature of our dataset and reduce the chances of mode-collapse. Many of the other GAN

architectures that we will be attempting in this paper will have been inspired by, or use some aspects of the CGAN model, and so it is important to understand how it can be utilised for our purposes.

2.3.5 LoGAN

In the paper 'Logo Synthesis and Manipulation with Clustered Generative Adversarial Networks' [13], the authors agree that GAN's are more suitable for generating detailed samples from a multi-modal distribution than other state of the art frameworks, such a Variational Autoencoders, due to the latter's blurry and low resolution results which are caused primarily by the "nature of the pixel-wise L2 loss used during training" [13]. In contrast, the adversarial/competitive nature of GANs lead to results of much higher resolution [13], which is what would be required for our problem of generating coats-of-arms. This higher resolution comes at the cost of the GAN being a notoriously difficult model to train due to its instability and its propensity to experience mode collapse, amongst other issues [14].

In Sage et al's paper, they proposed training a clustered GAN (in other words a conditional GAN with labels obtained from clustering) to help generate samples from a high modality logo dataset [13]. This clustering can be done in either using an autoencoder, or within a ResNet classifier, DCGAN or WGAN that uses an auxiliary classifier or layer conditional model [13].

The resulting generated samples were impressive, receiving both high inception and diversity scores, and CORNIA scores equivalent to those of the original images from the dataset [13]. In addition, the latent space of the trained networks allowed smooth transitions between generated samples, proved by performing interpolations between points in the latent space and showing the transition of the generated samples from one point to the other. This allows for further image manipulation, best exemplified by the further sharpening the generated images by using a "sharpening vector" by "averaging over the z-vector of a number of blurry samples and subtracting from this the average of a number of sharp samples" [13].

In Mino & Spanakis's paper 'LoGAN: Generating Logos with a Generative Adversarial Neural Network Conditioned on colour' [15], the authors propose "LoGAN: an improved auxiliary classifier Wasserstein generative adversarial neural network (with gradient penalty) that is able to generate logos conditioned on twelve different colours" [15]. The authors criticise Sage's approach of using clustered GAN's as inflexible from a designer's standpoint due to the synthetic cluster labels, stating that there is a need for "labels that could better convey what is in the logos, and present designers with more detailed selection criteria" [15]. In LoGAN, these labels are the most prominent colour to be used, with the end goal being to generate logos conditioned on one of these colour labels [15].

Regarding the architecture of the LoGAN model itself, it is an Auxiliary Classifier Wasserstein Generative Adversarial Neural Network with gradient penalty, with three components – the generator, the discriminator and then the additional classification network based on the colour labels [15].

Ultimately the results were not as impressive as Sage et al's work, with the generated images being very blurry in comparison. This paper did not use the same evaluation criteria as Sage's work, so they could not be directly compared, but a qualitative comparison between the sample of results generated by each paper

clearly shows that Sage's model generated images of much higher resolution and diversity, which Mino & Spanakis themselves acknowledge [15].

In a follow-up paper 'LoGANv2: Conditional Style-Based Logo Generation with Generative Adversarial Networks' [16], the authors try a new approach for LoGAN V2, inspired by the work done with StyleGAN [5] for "increased training stability for higher dimensional problems and better feature separation within the embedded latent space" [16]. This paper attempts to extend the StyleGAN architecture to allow for class conditions, in the hopes to gain an element of control when specifying the style for generated content, but also assisting the model in tacking the complex multi-modal nature of the dataset [16].

In addition, LoGANv2 uses progressive training (training on lower resolution images and progressing to higher resolutions) to achieve training stability and in combination with the StyleGAN elements, results in higher resolutions of the output [16]. Unlike LoGAN V1, the class conditioning of the model abandons the approach of using colour-based labels in favour of using labels based on the extraction of human interpretable characteristics via clustering [16].

The results of these clustering based labels led to visually distinct classes, but the associated language labels were not descriptive of the visual characteristics of the labels themselves [16]. With that said, these conditional labels were able to "help the model capture modes that might not be captured by an unconditional model" [16], and StyleGAN architecture led to stable training and output with much higher resolution to previous work [16].

2.3.6 EANN

In the paper 'EANN: Event Adversarial Neural Networks for Multi-Modal Fake News Detection' [17], the authors propose an adversarial network similar to a GAN for detecting fake news. Though there is no real generative component to this model, it does use a discriminator as part of its architecture and in the context of handling multi-modal datasets. The paper highlights the challenge of detecting fake news due its multi-modal nature and the fact that many fake news items are 'unseen events' [17], thus a need arises for a framework that can "learn transferable features for unseen events" [17].

This proposed architecture consists of three components, a multi-modal feature extractor, the fake news detector and an event discriminator [17]. The role of the discriminator is to measure the "dissimilarities among different events, and further learns the event invariant features which can generalize well for the newly emerged events" [17], and separate and remove events features specific to a single event from the event features that are shared amongst multiple events [17].

The multi-modal feature extractor component of the architecture is composed of two main elements, a text-based feature extractor and a visual feature extractor [17]. The text-based feature extractor uses a modified convolutional neural network known as of Text-CNN [17]. More relevant to us is the visual feature extractor which takes images of the news articles as an input and where the visual featured are extracted using a pretrained VGG19 network [17].

2.3.7 Multi-Modal Generative Adversarial Network

Another paper that uses GANs in the context of multi-modal datasets is Zhang et al's paper 'Multi-Modal Generative Adversarial Network for Short Product Title Generation in Mobile E-Commerce' [18], where the paper proposes a Multi-Modal Generative Adversarial Network (MM-GAN) for generating succinct product titles to be used in e-commerce storefronts, based on the original (longer) product titles and other information such as product descriptions and tags [18]. The part of this paper that is relevant to our problem is the multi-modal generator that that is used.

In essence the multi-modal generator is composed of 3 main components:

1) A Long Short-Term Memory (LSTM) encoder for the original long product titles. An LSTM [19] is a variation of a Recurrent Neural Network (RNN) architecture. The purpose of an RNN is to retain some information from a previous state of the neural network and pass it on to the current state in order to help form its calculations – this is extremely useful when your data is sequential in nature ie) the previous state is connected to the present state (such as time-series), but one of the major limitations is the vanishing/exploding gradients, where the gradients that are backpropagated though the network either go to zero or increase exponentially due the computations involved in keeping track of the sequential dependences in the data. An LSTM uses its architecture to overcome this problem and thus can outperform regular RNN's where the sequential nature is an import aspect of training the network.
2) Another encoder for the product tags
3) For the product image, a pretrained VGG16 is used to extract the features from the product images. A VGG16 [20] is a particularly powerful implementation of a Convolutional Neural Network (CNN), which is a neural network design for computer vision classification problems. Essentially the architecture of a VGG16, as well as CNN's in general is to transform the pixels of the input image into numerical values which are then passed through a series of filter 'layers' sequentially within the network architecture. The purpose of these filters are to identify the features of the image data and then use this to ultimately classify that image.

All there above components are then fed through a LSTM decoder to generate the short product titles [18]. Within this generator architecture, component dealing with the image data will be what is of interest to us.

2.3.8 GAN-Tree

Another GAN architecture worth examining is the GAN-Tree, first proposed in the paper 'GAN-Tree: An Incrementally Learned Hierarchical Generative Framework for Multi-Modal Data Distributions by Kundu et al' [21].

The authors of this paper highlight the challenges of generating samples from multi-modal datasets and acknowledge that though other frameworks exists for tacking this problem, they tend to be less successful depending on the mode components initially selected [21], such as using colours in LogoGAN V1.

In contrast to these approaches, the authors introduce the GAN-Tree architecture, where no assumptions are made about the number of modes within the dataset

and which instead uses a top down hierarchal approach to split the data into the modes which are most effective [21], along with allowing for the addition of additional child modes within parent-modes through updating the branches of the GAN-Tree [21]. The authors claim that this allows for more flexibility in choosing the right modes and superior results to prior approaches [21].

Each node of this GAN-Tree is its own trained GAN, and starting with the trained node, from which the generate results are clustered into two classes, and each class is passed down to a new node whereby the a new GAN is trained on the data passed down to it, and whereby the generated results are again clustered bi-modally and where both clusters are passed onto the next child node, and the process continues [21]. Various implementations of GAN-TREE on different datasets and in different configurations show great results when evaluated for on FID and Inception scores [21].

2.3.9 StyleGAN

StyleGAN was first introduced in the paper A Style-Based Generator Architecture for Generative Adversarial Networks [5], where the motivation was to create an architecture that would lead to the separation of the feature attributes within the images that are generated, along with providing the user with additional controls on separate feature attributes.

StyleGAN is an extension of the Progressive Growing GAN (ProGAN) architecture [22], whose architecture is centered on training with incremental increases in the size of the Generators output, so that the model starts by learning the low-level baseline details of the images and increasing learns to generate finer and finer details as the model output grows in size. This results in very high-quality synthesized images when the model is fully trained.

This is all handled by a new Generator architecture, with no novel changes made to the Discriminator, loss function, parameters etc.. This new Generator consists of three major contributions. Firstly, it introduces mapping network which generated a style vector from the input latent vector, and where this style vector allows control over distinct features of the generated content. By creating this additional style vector, the intra-feature correlations found in the original data distribution are divorced from one another, and thus reduces the aforementioned entanglement issue.

Secondly, noise is introduced at each layer of the generated images, which increases stochastic variation in the generated images. This is especially useful for our case, given the stochastic nature of coats of arms, and the glyphs and patterns contained within. Finally, this style vector that was created by the mapping network is applied to the each layer of the generated images using Adaptive Instance Normalisation [23] after the noise from the previous step was added.

The below image displays the architecture in more detail, where z is the latent vector, w is the style vector, A is the Adaptive Instance Normalization being applied and B is the noise being injected into the network:

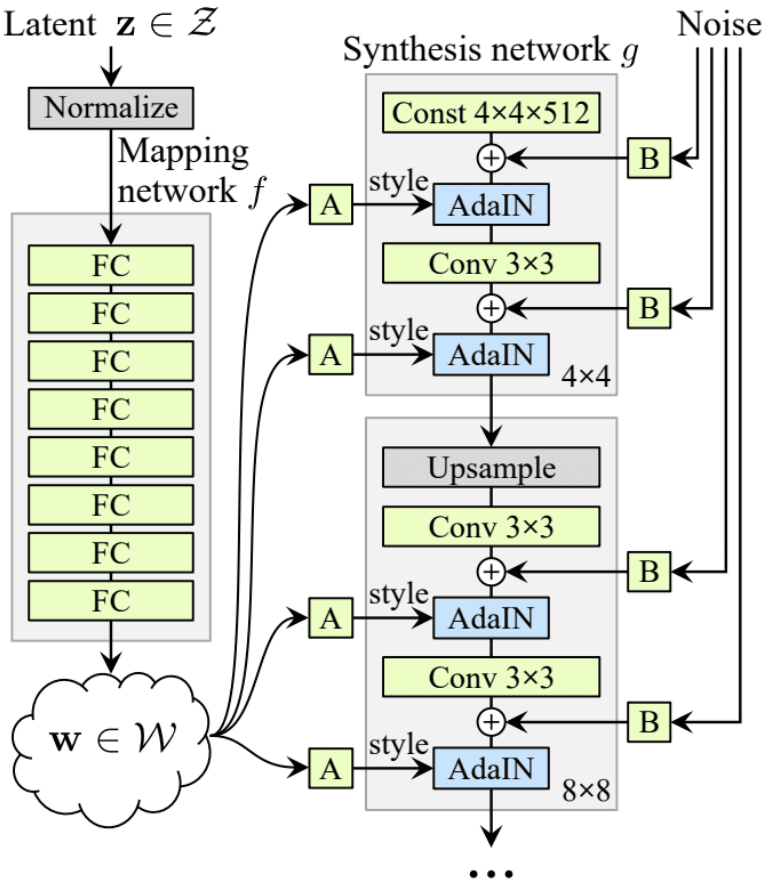


Figure 4: StyleGAN architecture [5]

2.3.10 StyleGan2

StyleGAN2 was first introduced in the paper, Analyzing and Improving the Image Quality of StyleGAN [24], where it's primary motivation was to improve upon the original StyleGAN by handing some of the image artifacts that the original model had a tendency to create and creating an overall improvement in the quality of generated images.

The two biggest changes in StyleGAN2 include weight demodulation, which help solve for the 'droplet' artifacts what were commonly seen in images synthesized by the original model, and a complete overhaul of the progressive nature of increased resolution sizes. In StyleGAN2, the latter has been replaced by a skip connection design, inspired by ResNet [24]. Skip connections, as the name implies, involves skipping layers of the neural network, but without any loss to the quality of the generated images.

2.3.11 StyleGan2-ADA

StyeGAN2-ADA is a further extension on the StyleGAN2 architecture, but designed specifically to work in uses cases where there is limited data. This makes it perfect for our scenario, where we have a relatively low number of training images available. First introduced in the paper 'Training Generative Adversarial Networks with Limited Data' [25], the ADA in StyleGAN2-ADA stands for Adaptive Discriminator Augmentations (not to be confused with Adaptive Instance

Normalisation), and is the key contribution to this new architecture that allows for quality results with low training data availability. It works by applying data augmentations to both the generated and real images that are presented to the Discriminator, and using this to train the generator.

The key aspect here is that these augmentations are not made on the training data itself before it is passed through the model, instead it is only applied to the real images and generated images when presented to the discriminator. Due to the same augmentations being applied to both real and generated, the discriminator makes its decision and the generator is trained accordingly without having to deal with a 'circular' dataset where generated images are used to bolster the real dataset [27].

In essence, this allows for high-quality images to be generated with low number of real samples - so it is the StyleGAN variant most suited for our problem domain.

2.3.12 DALL-E

While DALL-E isn't a GAN model, it is generally considered to be the state of the art when it comes to image generation at the time of this writing, so it is worth examining.

DALL-E was revealed by OpenAI in January of 2021, and uses text prompts to generate high-resolution images representing this prompt in a variety of styles. In April of the following year, DALL-E 2 was revealed, which produces results with up to four times better resolution and with a much greater ability to capture the desired results from the text prompt provided. As per OpenAI's website, users were asked to evaluate the results from both DALL-E 1 & 2 and 71.7% showed a preference for DALL-E 2 in terms of caption matching, and 88.8% agreed that it performed better at generating photorealism [26].

As the source code for either model has not been released, it is difficult to say what changes to DALL-E 2 have contributed to the improved results, so we will focus on DALL-E 2 exclusively for now.

DALL-E 2 was revealed in the paper "Hierarchical Text-Conditional Image Generation with CLIP Latents " [27]. In essence, the model works by mapping text inputs to visual representations, using Contrastive Language Image Pre-Training (CLIP). CLIP is another OpenAI model, first introduced in the paper "Learning Transferable Visual Models From Natural Language Supervision" [28], that is trained on 400 million captioned images, and with the ability to measure how close an image corresponds to a text prompt using cosine similarity.

With a trained CLIP model in place, the next step would be to take the text input provided by a user, and generating an image from using the trained CLIP representation space. For this task, OpenAI use their own one of their own models: Guided Language-to-Image Diffusion for Generation and Editing (GLIDE) [30].

GLIDE is an a kind of a diffusion model, which generally work by adding gaussian noise to an image, then work backwards to remove it, in the process learning how to generate images from pure noise. The difference with GLIDE is that is has been specifically trained to work conditionally given text inputs. DALL-E's version of GLIDE is further modified to generate images based on the encoded text-image mappings

from CLIP.

Image below taken from DALL-E 2's source paper is a simplified representation of the model:

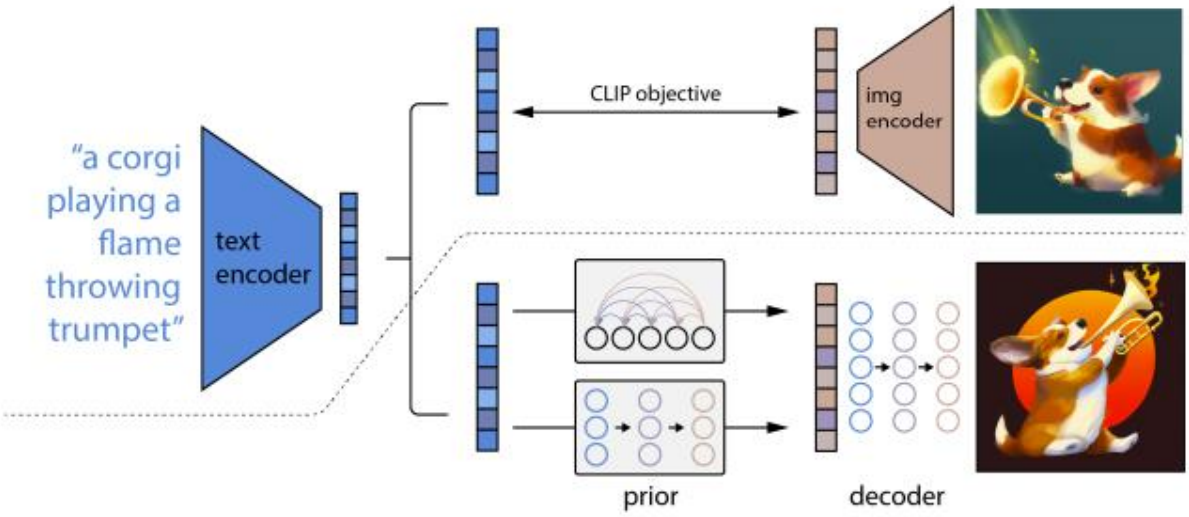


Figure 5: Representation of DALL-E 2 architecture [27]

2.5 Addressing mode collapse and feature entanglement

2.5.1 Mode Collapse

Mode collapse has been one of the most prevalent issues when trying to successfully train a GAN network since their original conception.

Mode collapse is when the generator or a GAN architecture fails to capture the diversity of data from the true distribution and instead collapses to a single mode, ie) produces near identical examples that continue fool the discriminator, as described in the paper 'Improved Techniques for Training GANs' [31]: "Because the discriminator processes each example independently, there is no coordination between its gradients, and thus no mechanism to tell the outputs of the generator to become more dissimilar to each other. Instead, all outputs race toward a single point that the discriminator currently believes is highly realistic" [31].

According to the authors of 'Diversity-sensitive conditional generative adversarial networks' [32], Conditional GANs are particularly susceptible to mode collapse [32], where "most cGAN approaches tend to learn an overly simplified distribution where an input is always mapped to a single output regardless of variations in latent code" [32] . The authors propose a method of regularisation on the generator ensure diversity of generated samples by adding an additional maximisation objective to the generator which penalises mode-collapse behaviour [32].

There are many other papers that address the issue of mode collapse and offer their own solutions, such using Wasserstein distance as a loss function [10], or using

minibatch discrimination where the discriminator compares batch samples to determine whether a given batch is real [31]. Unrolled GANS where the generator not only tries to fool the current iteration of the discriminator but also the discriminator's counter to the generator's response per iteration [33], or AdaGAN which addressed mode collapse by training multiple GANS across different modes and using the combination of each of these GANS as the final architecture [34].

2.5.2 Feature Engtanglement

Another issue that tends to arise with generated data from a GAN is that of feature entanglement. Feature entanglement, as the name suggests, is the nature for a GAN during training to 'tangle' distinct features together in its output if the training data has strong correlations between separate features. An intuitive example would be hair length and gender. In the case of a GAN trained to generate human faces, the entanglement between the hair and gender features may prevent the GAN from generating data where male faces have long hair or females short hair, which is due to the correlation between hair and gender in the training data [5].

This may be especially problematic when generating a GAN to generate coats-of-arms, as the presence of one feature in an image might lead to a lack of diversity in other features, for example, if a horse symbol appears with relative frequency on red background, and/or with a two other horse symbols – then GAN may continue to produce crests of this description if any one of those features are present. Considering these coats-of-arms are constructed of any number or combination of symbols, in any combination of colour or background and in any position – this problem would need to be addressed.

The most pertinent solution to feature entanglement in this instance would be the separation between feature layers in a StyleGAN architecture [5], as the StyleGAN architecture is a core component to some of the architectures that have already been called out as similar work, such as LoGAN V2.

2.6 Evaluation criteria

GANs lack a universal objective function, meaning that there is no standardized way of evaluating a GANs' performance [35]. Though there is no universally acknowledged standard measure, there are many different measures that have been used, the most commonly referenced being *Inception Score (IS)*, *Frechet Inception Distance (FID)* & *MM-SIM*

2.6.1 Inception Score

Inception Score (IS) was initially proposed in the paper "Improved Techniques for Training GANs" [35], where the authors use a classifier based on the Inception image classifier [36] to simultaneously measure the diversity of the generated images, and how similar it is to real examples of the data the generator is attempting to replicate. IS is named after the Inception v3 classifier on which it is based [36], where it takes images and returns probability distributions of for a set of possible labels for an image.

When a single subject is present in the image, as that will result in a much more

narrow probability distribution skewed towards a particular label, whereas multiple subjects present in an image will result in the a more uniform distribution. IS extends on this by summing the label probability distributions across all images, resulting in what is called the marginal distribution an which represents the overall diversity within the generated dataset.

Thus, the IS looks for two things, a clear probability skewness towards a single object within an image, and then a diversity of objects throughout the dataset. To consolidate these two measures, which are opposites in a way, into a single score, the Kullback-Leibler (KL) [37] is used, which is a measure of similarity between probability distributions.

Using these two distributions , we would hope for a high KL divergence (as we want these intra-image probability to be skewed and marginal probability to be uniform). This is essentially the Inception Score. There is no limit to high an Inception Score can be, but the lowest a model can score is 0.

One of the biggest drawbacks with using an Inception Score for this project is that it's limited by the class labels contained within the Inception classifier, and having had a look at https://image-net.org - one largest labelled image databases available, there were no samples of crests or coats of arms.

The other issue is that the nature of dataset is in directly conflict with how IS is calculated. Coats-of-arms by nature have several discrete symbolic images embedded within them, so we would never be able to get a highly skewed probability for a single class level in most instances. This lack of suitable reference images in the database and the multimodal nature of within each image means that our model would get a poor Inception Score regardless of the quality of the generated content

2.6.2 Frechet Inception Distance

A Frechet Inception Distance (FID) was proposed in the paper "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium" [38] as an improvement over the Inception Score as an evaluation criteria for GANs.

The core difference between Inception Score (IS) and FID is that the IS score is based purely on the generated images, with no comparison between the generated images and real-life equivalent, whereas the FID does evaluate the generated images by comparing them directly with real images. As FID is an extension on IS, it is also based on Google's Inception v3 classifier, but in FID - the last pooling layer of the images is used to capture the features of both the real and generated images, across which both the mean and covariance is calculated separately for each.

Similar to IS, the distance of these two distributions is then measured, but instead of using the KL divergence as a measure of distance, FID uses the Frechet distance. The Frechet, or Wasserstein-2, distance factors in the order and location of points along the curves of the data to calculate the similarity. In contrast to an Inception score, a FID score that is lower indicates generated content that is closer to the real images.

While using FID might not raise the issue of needing a highly skewed probability distribution for most images, it still has a reliance on an image Inception v3 which has not been trained on any images of coats-of-arms. Further to this, the authors of

the FID paper recommend an absolute minimum of 10k samples in our training data for out FID to be effective, which writes this off for us as an evaluation metric [38].

2.6.3 MM-SIM

Though generated image diversity is encapsulated within the Inception and FID scores, it can be also measured using multiscale structural similarity (MS-SSIM) [39], which is scored between 0 and 1 where the lower scores indicate greater diversity between the samples and more similar images the closer to 1 an image gets.

As the background colour and overall shape of the shield itself is identical for all samples, and the true diversity of the content only takes place within the boarders of the shield of the crest, then there could be a risk that this diversity measure might be not be suitable in measuring diversity as the MM-SIM might be artificially inflated by the consistency of the background and crest shape.

2.6.4 Visual Assessment

Despite these formal measures that have been introduced to assess performance of GANs, it is still acknowledged that human visual assessment is still one of the more common and reliable means of evaluation [40]. This of course, depends on the context of what the GAN is attempting to generate – human assessment of the quality of a generated image of a face would be far more reliable than an assessment of generated images of x-rays, or tabular data.

However, given that the generated content in this instance will be a coat-of-arms, I would be very comfortable in assessing the quality of the images myself. Diversity would also be easily assessed manually, as one would be able to quickly identify a lack of diversity between the symbol in the crest with the combination of position/background/colour etc..

Chapter 3 Data

The initial dataset used for this project will be approx. 1000 images sourced from irishsurnames.com [41]. This was supplemented by a further 373 images scraped from various sources on Google images.

The images were all resized to the same size. Below is a sample from this dataset:

Figure 6: Sample from our coat-of-arms dataset.

The biggest risk I can envision is that the training data used will be insufficient. As compared to the typical datasets which are used for training GANS and consists of tens of thousands of images (Chapter 6).

Data augmentation is a possibility, though some of the models, specifically StyleGAN2-ADA are include data augmentation as per of their architecture (Chapter 2.3.11), so will be particularly well suited to scenarios where there is a low amount of training data, such as this one.

Chapter 4 Methodology

4.1 Introduction

Here we will discuss the models that were actually used to attempt to generate convincing samples of coats of arms, including where the models were sourced from, their hyperparameter settings and any modifications that were made for this exercise.

We first started with DCGAN and LSGAN as a starting point to review performance on relatively simple GAN architectures. We then moved on to a Wasserstein GAN to review the impact it would have on mode collapse. Following from this, we examined a Conditional GAN to see if we could get better results by training conditionally on crests based on their imagery. As the qualitative grouping of crests based on their content proved difficult, we also trained a Conditional GAN based on groupings done by a clustering algorithm which is broadly the same approach LoGAN takes. A further modification was then applied to this model by applying the Wasserstein loss function from WGAN to this clustered Conditional GAN.

Finally, we applied a StyleGAN2-ADA model to see how a state-of-art GAN architecture would perform. This was followed by examining how the the state-of-the-art in image generation would fare by seeing if DALL-E 2 could produce better results than the GAN models.

All models were built on PyTorch (bar DALL-E 2).

All final models along with the data can be found my GitHub repo for this project:

https://github.com/rmacdonncha/Capstone_Masters_Thesis

4.2 DCGAN

The source code for this model was taken from the the GitHub repository

GANs by user mahmoudmelsayed [42]

For this DCGAN, there were 64 feature maps set in both the generator and discriminator, with a batch size of 128 and set to iterate over 500 epochs.

An ADAM optimiser was used, of which the Beta1 hyperparameter was set to 0.5 and a learning rate was set to 0.002.

4.3 LSGAN

The source code for this model was taken from the same repository as the DCGAN model [42].

Given the source of the LSGAN was the same as the DCGAN, no additional modifications were made and the hyperparameter setting were kept identical as the DCGAN.

The model as a whole was identical other than the change in loss function.

4.4 Wasserstein GAN

The model used was sourced from the Kaggle.com page “WGAN to regenerate Celeb Faces using Pytorch” [43], which as the name suggests was built for the CelebA dataset and modified to handle the coat-of-arms dataset.

Like the DCGAN and LSGAN, feature maps were set to 64 and training was set to 500 epochs. As per the recommendation of the author of this article, learning rate was set to 0.00005 and the weight clipping for the critic was set to 0.02.

4.5 Conditional GAN

The source code for this model was taken from the same repository as the DCGAN and LSGAN models [42].

All the hyperparameter settings were kept the same as those aforementioned models, exect for the image size which was reduced from 64 to 32, and a increase of batch size from 12 to 50, as these settings seemed to offer better performance in this case.

One of the challenges we face is that conditional GANs require image data that have class labels, such as the MNIST dataset, and our coat-of-arms dataset has no such labelling. This being the case, we needed to create and assign class labels to our dataset as a first step. From manually reviewing the data, 22 class labels were created based on the most predominant glyph or pattern present in a given crest. This resulted in a somewhat skewed dataset in which lion glyphs were considerably more frequent than most others. It was also decided to have a requirement of no less than 15 instances of a glyph being present before making it its own class to prevent further skewedness of the dataset and having too many classes. Given the wide diversity in our data set that resulted in a large ‘other category’, which may hurt the models’ capacity to generalize on this category.

Predominant Glyph	Count of Crest	% Of Total	Class Label
Lion	239	17.4%	1
Other	194	14.1%	2
Stripe	141	10.3%	3
Bird	114	8.3%	4
Star	100	7.3%	5
Dragon	68	5.0%	6
Moon	57	4.2%	7
Cross	49	3.6%	8
Cross_Diag	48	3.5%	9
Fleur	47	3.4%	10
Horse	37	2.7%	11
Stag	36	2.6%	12
Flower	34	2.5%	13
Tree	35	2.5%	14
Hand	24	1.7%	15
Wheel	30	2.2%	16
Shell	23	1.7%	17
Castle	24	1.7%	18
Sword	19	1.4%	19
Bull	18	1.3%	20
Diamond	21	1.5%	21
Ship	15	1.1%	22

Figure 7: Table of most prominent glyphs present in our datasets

In what is probably the most challenging aspect of this dataset, is that there are many cases where there are more than a single kind of glyph present, and sometimes it is not clear which is more predominant. My qualitative assessment of which class label to assign to a crest could no doubt be questioned, but in cases like this I defaulted to assigning the class label based on the category which had the lower count, so as to try and reduce the skewness of the dataset. For example, if I was presented with a case where a lion and a Sword where given equal prominence within the crest, I would have given it the class label of the sword so as to not further exacerbate the dominance of the lion class label.

Here are some examples of the dataset and the respective labels I had assigned for reference:

Bird:

Lion:

Ship:

4.6 Clustered Conditional GAN

As noted when performing the CGAN, there are challenges with manually annotating our crest images and categorizing them qualitatively. Due to the nature of glyphs often containing several different kinds of discrete representations of real-world objects, making an a decision as to which class they should belong to was often arbitrary. Obviously this does not lead to the best results when our CGAN attempts to recreate examples from each class.

In an effort to help resolve this, I have used unsupervised learning to cluster our crest images based on visual similarity and create the classes this way. These crests in their new classes would be then passed through the same CGAN as was used before to see if the results produced showed any improvement. This was broadly the same approach taken in the LoGan model (Chapter 2.3.5).

This approach was recommended in Sage et al's paper, they proposed training a clustered GAN (in other words a conditional GAN with labels obtained from clustering) to help generate samples from a high modality logo dataset [13].

The CGAN model itself, including hyperparameter settings, was identical to CGAN model previously used. The changes made here were on the data before being fed into the CGAN model. The steps were feature extraction, principal component analysis and finally k-means clustering, the code of which was take from the following article [44].

4.6.1 Feature extraction with VGG16

The first step involved in clustering our crest images is to get the feature vector of the images that we can then subsequently cluster. For this task, I used a pre-trained VGG16 convolutional neural network. The VGG16 model was first proposed in the paper Very Deep Convolutional Networks for Large-Scale Image

Recognition, by K. Simonyan and A. Zisserman from the University of Oxford [20].

The key differentiator for VGG16 was that instead of having a high number of hyperparameters, the model instead uses three Covnets sequentially, with each using an increasing number of very small (3x3) convolutional layers, from 10 to 13 to 16 and up to 19 weight layers. This model was very successful, and was the 1st runner-up in the image classification and localisation in the 2014 ImageNet Large Scale Visual Recognition Competition.

For our purposes, we will not be using it to classify images, but instead as a feature extractor, by removing the final prediction layer leaving us with the feature layer as an output.

4.6.2 Dimensionality Reduction Using Principal Component Analysis (PCA)

The resulting feature vector has 4096 dimensions, so in order to reduce the computing power required for the next step we will use Principal Component Analysis (PCA) to reduce the number of features down to 100.

PCA works by combining the our dimensions and leaving behind the best set of combined dimensions which best explain our data, and independent of one another. This results in a reduction of the overall number of variables to consider as well [45].

In this case I chose to reduce the dimensions down to 100 - an arbitrary choice but one I felt left enough dimensions to not have the final feature set be oversimplified, while still reducing the number of features down by 98%.

4.6.3 Clustering our feature vector with K-Means Clustering

Once we have the final feature we can to the clustering step, where we will be using K-Means Clustering. The standard model for K-means clustering as it's known today was first described by Stuart P.Lloyd in the paper Least Squares Quantization in PCM [46].

K-Means Clustering works by segmenting our observations in to k clusters, and which cluster each observation belongs to is determined by which cluster has the minimum squared Euclidean distance from its mean to the observation. This is known as the Assignment Step, and it is followed by the Update Step, where after an observation is assigned to a cluster, the cluster means are recalculated before the next observation is assigned.

The k in k-means clustering is the number of clusters we want to be left with, and it is an important step to decide how many clusters we want to use. If the value of k is too low, then the observations within the cluster will not be meaningfully distinct from one another. If the value of k is too high, then we will have too much separation and too few observations of data within each cluster. This could be especially problematic for us, given that our data is relatively scarce anyways, and would definitely have a negative impact on our CGAN models ability to generate meaningful recreations of the data within each class.

To decide on the right value of k, I plotted then reduction in the sum of squared Euclidean distance over the value of k using an elbow chart:

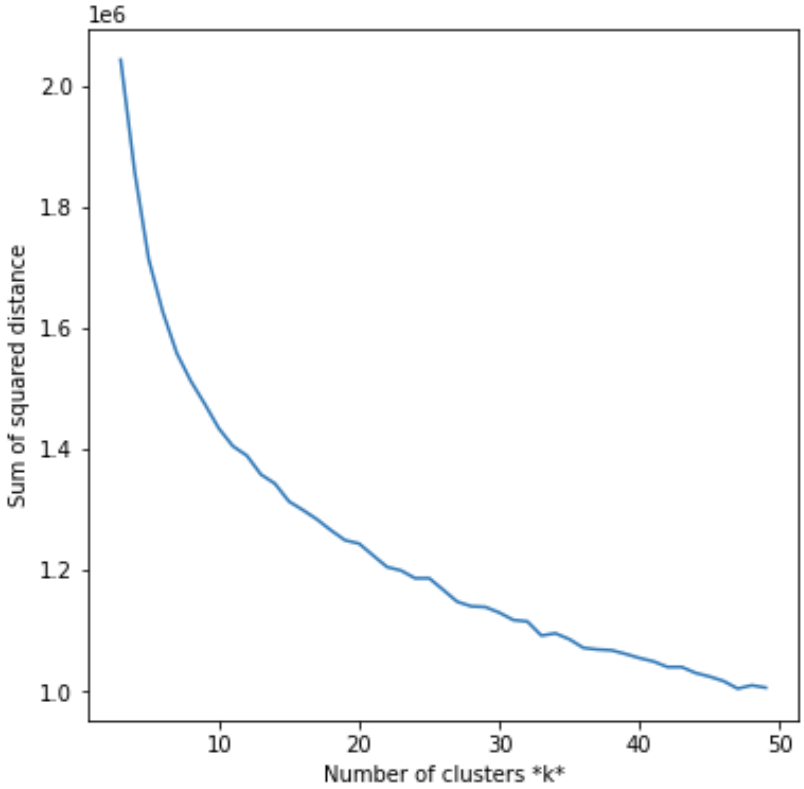


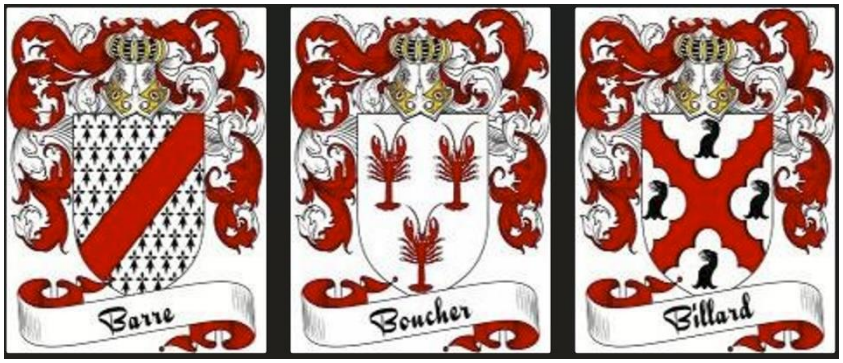
Figure 8: Plot of number of clusters over Sum of sq-distance in our dataset

As we can see from the above, it would seem that setting k=10 would be a good choice as it is the point where we stand to see diminishing returns in the sum of squared distance as the value of k increases.

After setting the k-value, we then run the k-means clustering algorithm on our feature vector which leaves us with 10 clusters of out crest images, which are then saved within their own folders on my local machine.

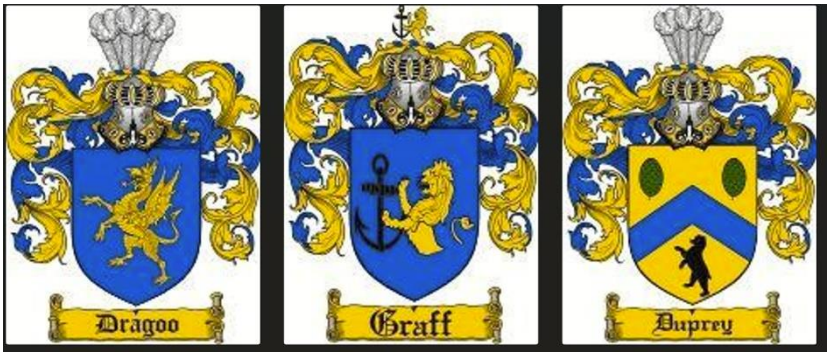
The resulting images within each class seem to be primarily clustered based on the dominant colour, which may be problematic as the real problem we are trying to solve is overcoming the multimodal nature of the glyphs within each crest, though there does seem to be some degree of qualitative similarity in this respect too, as we can see from some examples below:

Cluster 1:



Cluster 2:

Cluster 3:



Finally, using the exact same CGAN model as before we pass through our new classes of crests and run it.

4.7 Clustered Conditional WGAN

To further increase the likelihood of image diversity and reduce mode-collapse, the next logical step was to continue using a CGAN, but to use a Wasserstein loss function.

The model was sourced from the GitHub repository Muhammad-Ibrahim-Khan/Conditional-WGAN-GP [47], and was based on the paper "Conditional Wasserstein Generative Adversarial Networks" by Cameron Fabbri at the University of Minnesota [48].

This model was trained on the clustered class labels from the previous model, and as it was quite slow to run, the number of epoch had to be capped at 100. All other hyperparameters were kept as they were set on the source, given it was so slow to run it was not practical to compare and contrast different runs with different hyperparameter settings.

4.8 StyleGAN2-ADA

The source code for this model was taken directly from the GitHub repository [49].

No data preprocessing was done on the dataset, as the ADA aspect of this mode takes care of data augmentation. With regards to this ADA augmentation, images were mirrored left to right, but not upside-down. All other settings were kept as they were on the pre-trained model that was imported. There was no

limit set to the epochs, and the model was trained until Google Colab usage limits were reached (Chapter 5.7)

Chapter 5 Results

5.1 DCGAN

As was expected, the DCGAN struggled with generating the detailed content of the coats-of-arms, where we can see that it was unable to generate recognizable glyphs symbolizing real-world objects inside of the crest shape, and instead produced shapeless abstractions and colours.

Figure 9: DCGAN-generated images after 500 epochs

5.2 LSGAN

Upon being qualitatively assessed, the samples generated by the trained LSGAN did seem to be sharper, and have greater diversity than those generated by the DCGAN. However, they still ultimately suffered from the main issues that DCGAN experienced, namely the lack of recognizable real-world symbols within the crests.

In addition, despite the authors assertion that the LSGAN was more stable during training, I found that it suffered from mode collapse far more frequently than DCGAN, with the output oftentimes resembling the image below

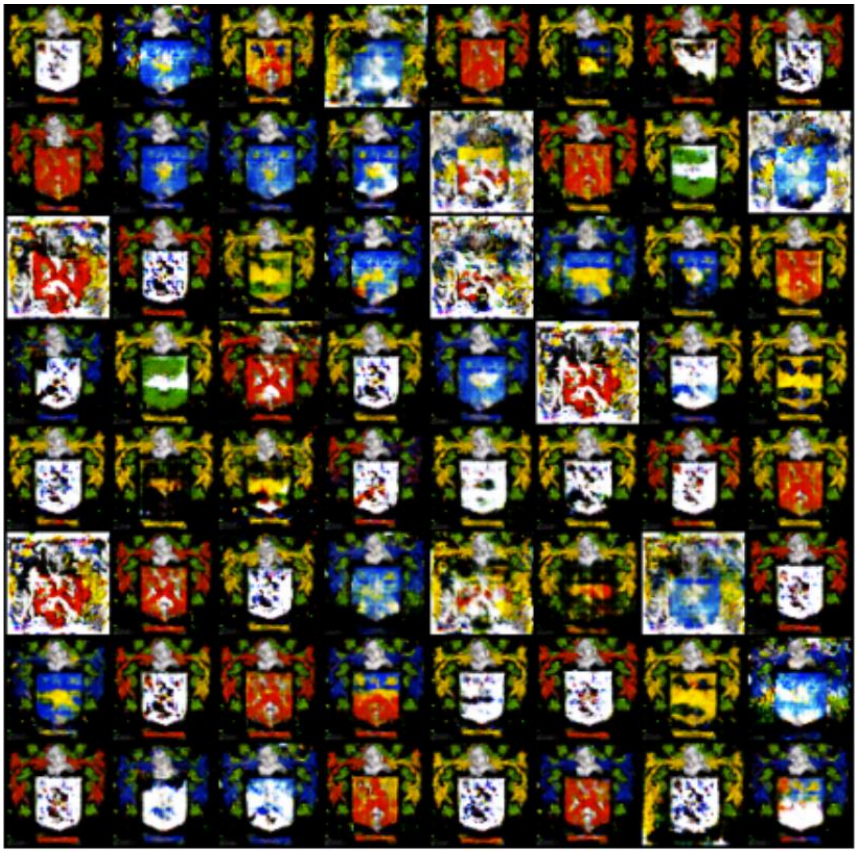


Figure 10: LSGAN-generated images after 500 epochs

5.3 Wasserstein GAN

Upon qualitative assessment, there does appear to be a greater diversity in generated images compared to the DCGAN and LSGAN, which is to be expected.

However, there are still no recognizable glyphs generated within the crests themselves - which remains the core problem we are trying to solve:

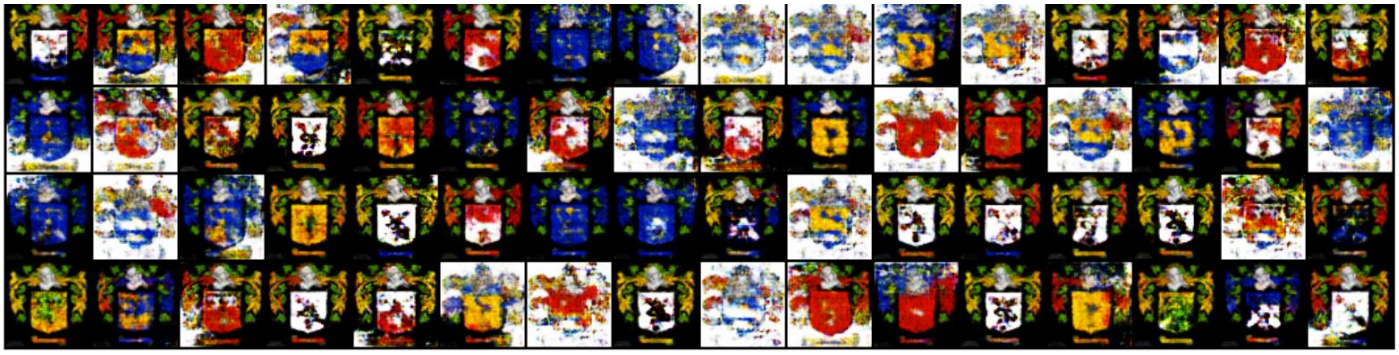


Figure 11: WGAN-generated images after 500 epochs

5.4 Conditional GAN

Unfortunately our CGAN could not product results that were any more impressive the previous attempts.

Again, it seems that while the model was able to produce good representations of the overall shape and structure of a coat of arms, and to some the background pattern, it was not able to reproduce any recognizable glyphs contained within.

In the above images, where each column represents a class, we can see that it also suffered from mode collapse towards the end of its training within certain classes if we compare the generated results after 250 epochs to those after 500 epochs.

Figure 12: CGAN-generated images after 250 epochs (left), and 500 epochs (right)

5.5 Clustered Conditional GAN

Ultimately, there results after the model had finished running were no more impressive than before.

However, towards the tail end, around 400 epochs there did seem to be the beginnings of recognizable glyphs beginning to take shape, that of a lion and tree in the style they are typically presented in these crests, however at this point mode collapse had occurred, and had continued by the time training had finished at 500 epochs.

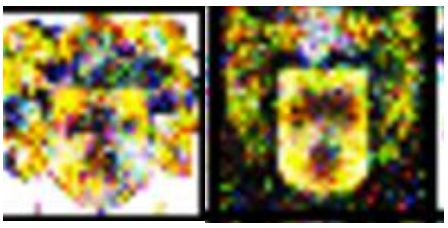


Figure 13: Apparent forming of a lion glyph (left) and tree (right) emerging after 400 epochs

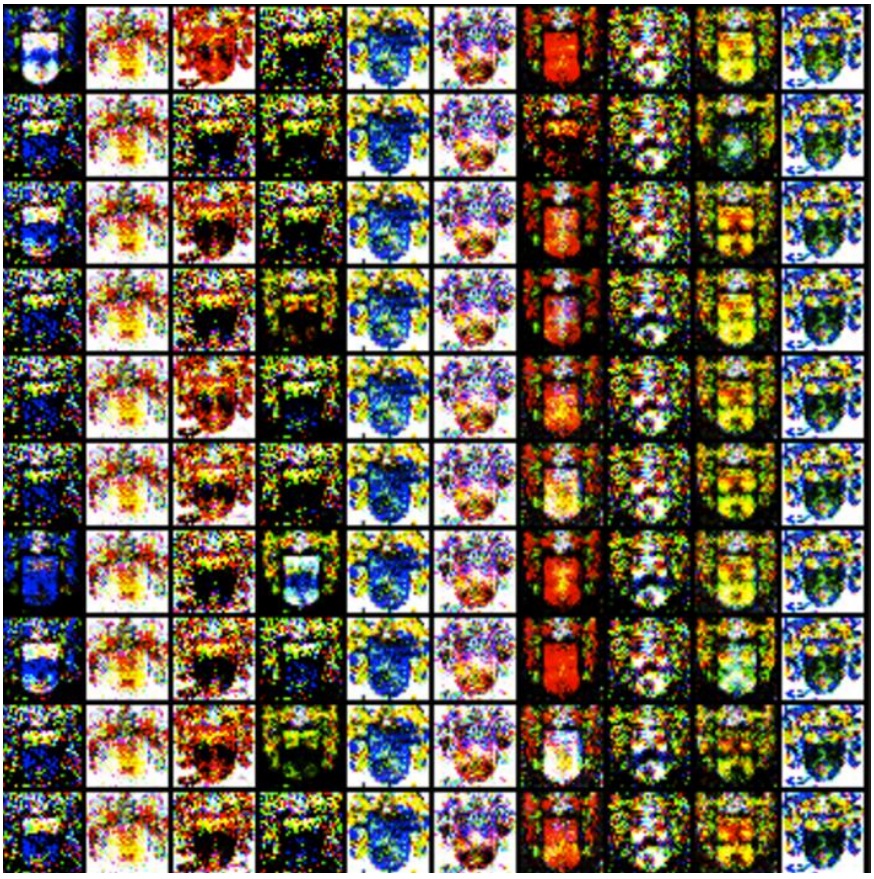


Figure 14: Clustered CGAN-generated images after 500 epochs

5.6 Clustered Conditional WGAN

Here we see a noticeable step-up in results, with greater diversity, no mode collapse, and some of the background patterns we typically see in these crests forming. However, the results are overall still quite blurry, and no recognizable glyphs representing real-world objects can be discerned:

Figure 15: Clustered Conditional WGAN-generated images after 500 epochs

5.7 StyleGAN2-ADA

Within 100 epochs, the results from this model were outperforming any of the previous models – there background patterns had emerged that were clear, there was good diversity and some glyphs with recognizable shapes were beginning to form:

Figure 16: StyleGAN2-ADA sample images after 100 epoch

As such, I decided to leave this running for as long as possible, until the Google Colab usage limits were reached. Because of needing to restart the model several times per day due to running out of space on my drive, I unfortunately lost the final number of epochs that this ran for, but I would estimate it would have been in the range of 2000-3000 epochs in total.

Here is a sample of the results after training had completed:

Figure 17: StyleGAN2-ADA sample images at end of training

While at initial glance the results don't seem that much different from the results at 100 epochs, there were some improvements with regards the emergence of additional recognizable glyphs.

At 100 epochs I could only detect a lion glyph (which was by far the most common in our training data set), but here are some samples of what emerged at the end of full training:

Lion:	Bird:	Dragon Head:
		
Tree:	**Sun & Moon:**	**Anchor:**
		

Figure 18: Table of some of the most convincing generated glyphs

5.8 Comparison with state-of-the-art (DALL-E 2)

As a point of comparison between our models and the true state-of-the-art in image generation, I entered the text prompt 'Coat-of-arms' into OpenAI's DALL-E 2 model, available on their website [26].

The results below show a perfect recreation of various styles of coat's of arms, with impressive detail. The images are sharp, with no blurriness or distortion. Even when we look at the lion glyphs that are present in two of the rests, we can see that though they both have that distinct style that is associated with lion glyphs (standing on two hindlegs, facing left with outstretched forepaws and tongue), there are subtle differences within the details, such as the claws and tongue being coloured vs not, a more jagged representation vs smoother lines and curves and the totally different crown design that rests on top.

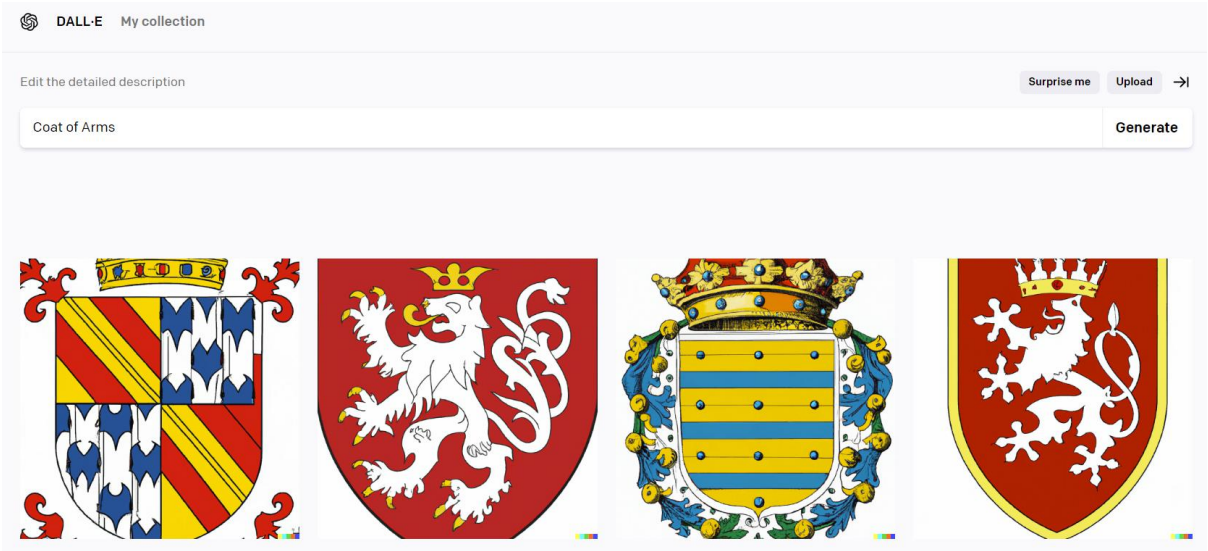


Figure 19: Sample images generated by DALL-E 2

Chapter 6 Conclusion

Based on the emergence of recognizable glyphs and background patterns in the StyleGAN-2 ADA model (and to a lesser extent the clustered conditional WGAN model), we can conclude that it's possible for GANs to create convincing samples of coats-of-arms.

I would assert that this is contingent on a few factors:

1. That the right choice of GAN model is used. We would need a model that is robust against mode-collapse, especially if we are training the GAN conditionally. We would also need a GAN that is robust against entanglement due to the nature of the dataset (Chapters 2.5.1 – 2.5.2).

2. That we have we enough training data. As per Nvidia's website, they recommend between 50,000 to 100,000 images for training high-quality GAN [50]. Our training data was limited to 1,373 samples, which I think severely limited the results of our models. This is evident by reviewing the sorts of glyphs that our most competent model was able to generate (Chapter 5.7) and comparing this to the glyphs that most commonly appeared in out dataset (Chapter 4.5). The high correlation between the two suggests that our GAN would have been able to create additional recognizable glyphs had there been higher numbers of these same glyphs appearing in out dataset.

We can also conclude that there is nothing inherent to coats-of-arms that prevent them from being replicated by generative models in general, given the highly plausible examples produced by DALL-E 2 (Chapter 5.8).

References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proceedings of the national academy of sciences 79*, 1982.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative Adversarial Nets," arXiv preprint arXiv:1406.2661, 2014.

[3] P. Wang, "thispersondoesnotexist.com," 06 2021. [Online]. Available: thispersondoesnotexist.com.

[4] A. Gharakhanian, "GANs: One of the Hottest Topics in Machine Learning," 19 December 2016. [Online]. Available: https://www.linkedin.com/pulse/gans-one-hottest-topics-machine-learning-al-gharakhanian/?trk=pulse_spock-articles.

[5] T. Karras, S. Laine and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[6] K. Jones and D. Bonafilia, "GANGogh: Creating Art with GANs," 18 June 2017. [Online]. Available: https://towardsdatascience.com/gangogh-creating-art-with-gans-8d087d8f74a1.

[7] WIkimedia Foundation, "Coat_of_arms," [Online]. Available: https://en.wikipedia.org/wiki/Coat_of_arms.

[8] A. Radford, L. Metz and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," arXiv preprint arXiv:1511.06434, 2015.

[9] X. Mao, Q. Qing Li, H. Xie, R. YK Lau, Z. Wang and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, Venice, 2017.

[10] M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, 2017.

[11] J. Hui, "GAN — Wasserstein GAN & WGAN-GP," 14 June 2018. [Online]. Available: https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490.

[12] M. Mirza and S. Osindero, "Conditional generative adversarial nets," arXiv preprint arXiv:1411.1784, 2014.

[13] A. Sage, E. Agustsson, R. Timofte and L. V. Gool, "Logo synthesis and manipulation with clustered generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[14] J. Hui, "GAN — Why it is so hard to train Generative Adversarial Networks!," 21 June 2018. [Online]. Available: https://jonathan-hui.medium.com/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b.

[15] A. Mino and G. Spanakis, "Logan: Generating logos with a generative adversarial neural network conditioned on color," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018.

[16] C. Oeldorf and G. Spanakis, "LoGANv2: Conditional style-based logo generation with generative adversarial networks," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019.

[17] Y. Wang, F. Ma, Z. Jin, Y. Yuan, G. Xun, K. Jha, L. Su and J. Gao, "Eann: Event adversarial neural networks for multi-modal fake news detection.," in *Proceedings of the 24th acm sigkdd international*

conference on knowledge discovery & data mining, 2018.

[18] J.-G. Zhang, P. Zou, Z. Li, Y. Wan, X. Pan, Y. Gong and P. S. Yu, "Multi-modal generative adversarial network for short product title generation in mobile e-commerce," arXiv preprint arXiv:1904.01735, 2019.

[19] S. Hochreiter and J. Schmidhuber, "LONG SHORT-TERM MEMORY," *Neural computation 9.8,* pp. 1735-1780, 1997.

[20] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint arXiv:1409.1556, 2014.

[21] J. N. Kundu, M. Gor, D. Agrawa and l. R. V. Babu, "Gan-tree: An incrementally learned hierarchical generative framework for multi-modal data distributions.," in *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[22] T. Karras, T. Aila, S. Laine and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196,* 2017.

[23] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE international conference on computer vision*, Venice, 2017.

[24] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

[25] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen and T. Aila, "Training generative adversarial networks with limited data," *Advances in Neural Information Processing Systems,* vol. 33, pp. 12104-12114, 2020.

[26] OpenAI, "DALL·E 2," April 2022. [Online]. Available: https://openai.com/dall-e-2/.

[27] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125,* 2022.

[28] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin and J. Clark, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning,*, Vienna, 2021.

[29] A. R. Lahitani, A. Erna Permanasari and N. Akhmad Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," in *4th International Conference on Cyber and IT Service Management*, Bandung, 2016.

[30] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv preprint arXiv:2112.10741,* 2021.

[31] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, "Improved techniques for training gans," arXiv preprint arXiv:1606.03498, 2016.

[32] D. Yang, S. Hong, Y. Jang, T. Zhao and H. Lee, "Diversity-sensitive conditional generative adversarial networks," arXiv preprint arXiv:1901.09024, 2019.

[33] L. Metz, B. Poole, D. Pfau and J. Sohl-Dickstein, "Unrolled generative adversarial networks," arXiv preprint arXiv:1611.02163, 2016.

[34] I. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel and B. Schölkopf, "Adagan: Boosting generative models," arXiv preprint arXiv:1701.02386, 2017.

[35] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, "Improved techniques for training GANs (2016)," arXiv preprint arXiv:1606.03498, 2016.

[36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the inception architecture for computer vision.," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[37] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics,* vol. 22, no. 1, pp. 79-86, 1951.

[38] M. Heusel, H. Ramsauer, T. Unterthiner and B. Nessler, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in neural information processing systems, 30*, 2017.

[39] Z. Wang, E. P. Simoncelli and A. C. Bovik, "MULTI-SCALE STRUCTURAL SIMILARITY FOR IMAGE QUALITY ASSESSMENT," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, 2003.

[40] A. Borji, "Pros and cons of gan evaluation measures," *Computer Vision and Image Understanding 179,* pp. 41-65, 2019.

[41] "irishsunames.com," 6 2021. [Online]. Available: www.irishsunames.com.

[42] mahmoudmelsayed, "mahmoudmelsayed/GANs," 13 April 2020. [Online]. Available: https://github.com/mahmoudmelsayed/GANs. [Accessed 2021].

[43] Y. Rampariya, "WGAN to regenerate Celeb Faces using Pytorch," 21 January 2022. [Online]. Available: https://www.kaggle.com/code/yogeshrampariya/wgan-to-regenerate-celeb-faces-using-pytorch. [Accessed 2022].

[44] G. Flomo, "How to cluster images based on visual similarity," towardsdatascience.com, 29 September 2020. [Online]. Available: https://towardsdatascience.com/how-to-cluster-images-based-on-visual-similarity-cd6e7209fe34. [Accessed 2022].

[45] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences,* vol. 374, 2016.

[46] S. Lloyd, "Least squares quantization in PCM," *IEEE transactions on information theory,* vol. 28.2, pp. 129-137, 1982.

[47] M. Ibrahim Khan, "Muhammad-Ibrahim-Khan/Conditional-WGAN-GP," 23 March 2022. [Online]. Available: https://github.com/Muhammad-Ibrahim-Khan/Conditional-WGAN-GP. [Accessed 2022].

[48] C. Fabbri, "Conditional Wasserstein Generative Adversarial Networks," University of Minnesota Computer Science and Engineering, Minnesota, 2018.

[49] D. Schultz, "dvschultz/ml-art-colabs," 20 October 2020. [Online]. Available: https://github.com/dvschultz/ml-art-colabs/blob/master/Stylegan2_ada_Custom_Training.ipynb. [Accessed 2022].

[50] I. Salian, "NVIDIA Research Achieves AI Training Breakthrough Using Limited Datasets," Nvidia Corporation, 7 12 2020. [Online]. Available: https://blogs.nvidia.com/blog/2020/12/07/neurips-research-limited-data-gan/#:~:text=It%20typically%20takes%2050%2C000%20to,falter%20at%20producing%20realistic%20results.). . [Accessed 2022].

[51] C. H. Chen, Handbook of pattern recognition and computer vision, World Scientific, 2015.

[52] K. Abe, B. K. Iwana, V. G. Holmér and S. Uchida, "Font creation using class discriminative deep convolutional generative adversarial networks," in *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, 2017.

[53] J. Brownlee, "A Gentle Introduction to Generative Adversarial Network Loss Functions," 2 September 2019. [Online]. Available: https://machinelearningmastery.com/generative-adversarial-network-loss-

functions/.

[54] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen and T. Aila, "Training generative adversarial networks with limited data," arXiv preprint arXiv:2006.06676, 2020.

[55] N.-T. Tran, V.-H. Tran, N.-B. Nguyen, T.-K. Nguyen and N.-M. Cheung, "On data augmentation for GAN training," in *IEEE Transactions on Image Processing 30*, 2021.