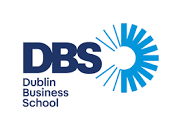
**A comparative study on Generative Adversarial Network and Convolutional Variational Auto-encoder for generating cartoon and logo images**

Mr. Akash Shivaji Kadam



Applied project submitted in partial fulfilment of the requirements for the degree of

[MSc in DATA ANALYTICS]

at Dublin Business School

Supervisor: Prof. Hubert Misztela

MSc in Data Analytics 2019-2020

**Declaration**

I, Akash Shivaji Kadam, declare that this work of research is my original and it has never been presented to any institution for the award of degree or diploma. Also, I have referenced all literature and sources used in the research correctly. Also, this work is fully compliant with the Dublin business school’s academic honesty policy.

Signed: Akash Shivaji Kadam.

Student Number: 10523055

Date: 25/08/2020

Dublin business school 2019-2020

**ACKNOWLEDGEMENT**

I would like to thanks my thesis guide Mr. Hubert Misztela at Dublin business school. Who has been mentoring me in all the aspects of the research and allowed me to think beyond the limitation and gave critical suggestions which proved valuable in the thesis. He had been very enthusiastic and supportive. Extended the helping hand whenever required. Also, controlled me in the privilege the bearing in the points he thought is required. The thesis couldn’t have finished much effectively without his guidance and energetic support long way.

Thank you.

Author – Akash Shivaji Kadam

Table of Contents

[**List of Algorithms and Method** 6](#_Toc49103780)

[**List of Tables and Figure** 7](#_Toc49103781)

[**Abstract** 9](#_Toc49103782)

[**CHAPTER 1: INTRODUCTION** 10](#_Toc49103783)

[**3.2** **Introduction** 10](#_Toc49103784)

[**1.2 Objective** 11](#_Toc49103785)

[**1.3 Dissertation Structure** 12](#_Toc49103786)

[**1.4 Challenges and Limitation** 12](#_Toc49103787)

[**Chapter 2: Related work** 14](#_Toc49103788)

[**2.1 Literature Review** 14](#_Toc49103789)

[**2.2** **Kullback-Leibler Divergence (KL):** 16](#_Toc49103790)

[**2.3: Deep Learning and Generative Models** 18](#_Toc49103791)

[**2.3.1** **Convolutional Neural network (CNN)** 18](#_Toc49103792)

[**2.3.2** **Discriminative Vs Generative Models** 21](#_Toc49103793)

[**2.3.4** **Deep Learning Generative Models** 22](#_Toc49103794)

[**Chapter 3: Methodology** 24](#_Toc49103795)

[**3.2** **Comparision of GANS and VAEs** 24](#_Toc49103796)

[**3.2 Datasets:** 24](#_Toc49103797)

[**3.3 Generative Adversarial Networks (GANs)** 26](#_Toc49103798)

[**3.4 Variational Auto-Encoders (VAEs)** 28](#_Toc49103799)

[**3.5 Data Pre-Processing:** 29](#_Toc49103800)

[**3.6 Model Stucture:** 30](#_Toc49103801)

[**3.6.1 Model A) GAN** 30](#_Toc49103802)

[**3.6.2 GAN Specification:** 31](#_Toc49103803)

[**3.6.3 Model B: VAE** 34](#_Toc49103804)

[**3.6.4 VAE Specification:** 35](#_Toc49103805)

[**3.7 Hardware and Software requirement:** 38](#_Toc49103806)

[**3.7.1 Software:** 38](#_Toc49103807)

[**3.7.2 Libraries Used** 39](#_Toc49103808)

[**Chapter 4 Results and Analysis** 40](#_Toc49103809)

[**4.1 Experimental Set-up** 40](#_Toc49103810)

[**4.1.1 Set-up for Cartoon Dataset:** 40](#_Toc49103811)

[**4.1.2 Set-up for Logos Dataset:** 41](#_Toc49103812)

[**4.1.3 Hyper Parameters** 42](#_Toc49103813)

[**4.2 Accuracy Measurement:** 44](#_Toc49103814)

[**CHAPTER 5 – RESULTS** 45](#_Toc49103815)

[**5.1 Image generation** 45](#_Toc49103816)

[**5.1.1 Cartoon Images** 45](#_Toc49103817)

[**CHAPTER 6: Anaysis and Disscussion** 52](#_Toc49103818)

[**6.1 Generation of Images:** 52](#_Toc49103819)

[**CHAPTER 7 CONCLUSIONS AND FUTURE WORK** 55](#_Toc49103820)

[**7.1 Conclusions** 55](#_Toc49103821)

[**7.3 Knowledge Gained** 56](#_Toc49103822)

[**7.3 Future Work** 56](#_Toc49103823)

[**References** 58](#_Toc49103824)

[**Appendices** 62](#_Toc49103825)

[**LIST OF ABBREVIATIONS** 64](#_Toc49103826)

# **List of Algorithms and Method**

1. Deep Learning
2. Simple Deep learning.
3. TensorFlow with deep learning.
4. Deep Generative models.

# **List of Tables and Figure**

|  |  |
| --- | --- |
| **Figure 1.1: Probability distribution** | **17** |
| **Figure 1.2: Different types of the pooling layer** | **19** |
| **Fig.1.3 Discimivative model classification (Saha, 2016)** | **21** |
| **Figure 1.4: Generative Models** | **22** |
| **Figure 3.1: Cartoon Images from dataset** | **24** |
| **Figure 3.2: Logos Dataset Sample** | **25** |
| **Figure 3.3: Structure of GAN.** | **26** |
| **Figure 3.4: Basic VAEs Architecture.** | **28** |
| **Figure 4.1: Image for Google colab file,logos 60k,epochs 150** | **31** |
| **Figure 5.1: sample of cartoon images generated with 30k sample GANs.** | **45** |
| **Figure 5.2: Images generated using saved model.** | **45** |
| **Figure 5.3: Images generated 60k slice,150 epochs and batch size of 256.** | **46** |
| **Figure 5.4: Generated images using 10k slice test dataset.** | **46** |
| **Figure 5.5: Images generated using slice of 60k,epochs 200 and batch size of 256. These images are predicted based on 10k slice data for training** | **47** |
| **Figure 5.6: logos generated using GANs model with 30k slice,200 epochs and batch size of 128.** | **48** |
| **Figure 5.7: Logos generated by GAN with 60k slice,200 epochs and batch size of 256.** | **48** |
| **Figure 5.8: Generated using VAE model with 60k slice,200 epochs and batch size of 256.** | **49** |
| **Fig. 5.9 Logo Generated using GAN model. 60k images and 150 epochs using model** | **50** |
| **Fig. 5.10 Logo Generated using GAN model. 60k images and 150 epochs** | **51** |
| **Fig.6.1 Over all Loss for GAN logos.** | **54** |
| **Tables List** |  |
| **Table 3.1 GAN Discriminator model for Cartoon images.** | **32** |
| **Table 3.2 GAN Generator model for cartoon dataset.** | **33** |
| **Table 3.3 for Generator model for Logos Dataset.** | **34** |
| **Table 3.4 for Discriminator Model for Logos Dataset** | **35** |
| **Table 3.5 Experimental part A and B on cartoon and logos dataset for image generation.** | **36** |
| **Table 3.6 Encoder for VAE Cartoon images.** | **37** |
| **Table 3.7 Decoder for VAE 30k Cartoon images.** | **38** |
| **Table 4.1 Experimental parameter on cartoon with 30k and 60k images.** | **39** |
| **Table 4.2 Experimental parameter on logos with 30k,60 and 100k images where k denotes thousand.** | **40** |
| **Table 4.3 Hyper parameter settings for VAE model.** | **41** |
| **Table 4.4 Hyper parameter settings for VAE model.** | **42** |
| **Table 4.5 Test set for cartoon and Logos dataset for VAE.** | **43** |

# **Abstract**

Generative Models, a discipline that over a few years has made a tremendous impact on the new technology and methods improvised in deep learning. The continuous progress in the generative model for generating data, image generation has been particularly been challenging. Hence, In the current study, the researcher focuses on achieving image generation of cartoon characters and Logos using the recent Deep learning approach. Exploring image generation technology is the urgency of the future. Keeping in mind that the drastic changes and innovation in deep learning will soon take over the manual artistic and designer work in the coming years which will be replaced by machines to generate unique and unseen designer work and characters. This will help Cartoon industries and multi-sector companies. Moreover, the researcher also focuses on a comparative study between Generative Adversarial Network and Variational Auto-Encoder model which have been proved effective in image generation. The results during the research inferred that image generation using two models is effective and there was some light shed on how few problems can be tackled. Also, the results are presented.

# **CHAPTER 1: INTRODUCTION**

## **Introduction**

In the new era for computers and changing technology it very important to be acquainted with the changes and upgraded technology which is helping the world in showing a completely new face. If we see in the past few decades ago, predicting was only knows as astrology but since past years machine learning and Artificial intelligence have played a big role in prediction and analysis of any domain and field. Which is a significant change. A new face of technology has also been achieved in not only predicting and forecasting future results and demands but now deep learning and Artificial intelligence is been used for generating data based on previous and historical data by studying and learning the data. Generating data that includes raw data, Images, Music generation, etc these all can be achieved by using Generative models. Most of the time for any machine learning algorithm we need a large amount of data for getting better and accurate outputs.

In some of the fields and domains, it is difficult to collect data particular type of data. So, Generative adversarial networks can be used for generating images, Raw tabular data, and music generation. Some of the application of GAN focused on these projects is Image generation models for creating a completely new set of images from existing image dataset. Exploring the field and generating new images from the existing was a more attractive and interesting topic which encouraged me to select the topic of image generation models. Here, Nowadays it has been great competition in the industry for creating and exploring the deep learning module wherein people are using multiple applications for editing and predicting images. Social media, marketing sector, movies, and other industries are making things automatic.

Generating and innovating cartoon character is an art and requires human imagination and energy. Nowadays most of the companies and film industry search for a unique look for any character which can get popular and become and brand. Image generation can be proved to be a boon. Collecting the very famous character and combining them to form a completely new identity will be a great idea. Here, the researcher is looking for generating cartoon characters and Logos for multiple industries and brands. As we all know for business marketing and face value play a crucial role. Whichever new brand has been evolved have a creative innovation and vision which makes them lead and popular among the local people? Considering the logo of AUDI in the automobile sector whose logo has now gained immense popularity. Which ultimately has been acknowledging by people. Thus, every company has a basic logo design requirement. Which can make their brand and outstanding feature in the market.

## **1.2 Objective**

The main motive of developing the model is to generate a completely different and realistic image using two different approaches. Generative models are an interesting branch of unsupervised learning. Many times, it is difficult to understand the data but generative models can do it. The models are trained and learned the data in any domain to generate a similar kind of data. This technique is used in multiple tasks like super-resolution of the image, image inpainting, and denoising, etc. Seeing the future of AI, it will be possible by machines to automatically learn the data and understand better.

The Generative models which achieved major success namely Generative Adversarial Networks (GANs) and Variational Auto Encoders (VAEs). The GANs have a special structure that consists of two Convolutional networks. Also, the VAEs uses an Encoder and Decoder structure. In the research, these two models will be used to generate the images from the training data. This model can solve the problems of researchers and enterprises to find an adequate dataset, create an artistic design and animated cartoon character. Moreover, the comparative study for the two models working and results will help to understand the models which can be used in different domains as per the analysis and requirement.

## **1.3 Dissertation Structure**

The Stucture for dissertation consist of multiple chapters with the actual flow of the thesis. Chapter 1 Covers the introduction for the topic and the objective of the selected topic. While the next chapter consist the literature review and associated work. This covers the reseach required for the completion of the thesis. It also contains the topics which were used through out the process. All the topics related to the generative models and various combination and associated parameters are described. In the third chapter the author has explained the comparison of GANs and VAE models , Methodology followed , dataset , structures for the models and software and libraries used. In the Next chapter, Set-up for the experiment on both the datasets is stated with the parameters. In Chapter 5, Results of the experimental set-up is displayed and explation of the results. In Chapter 6, Discussion and analysis for the performed experiment is well explained for understanding of the outputs and images generated using the model and comparision of the results.

## **1.4 Challenges and Limitation**

Images generation being a wide exploratory area being not fully explored. It was an challenging work as it required a lot of understanding and research of the innovative deep learning technology. When dealing with images and large set of data its always required an strong hardware and specification. As known deep learning models has requirement for large capacity specification. This was been an challenge as for study this specification are difficult. Hence, Google Colab being as free source with GPU and RAM was used for generative models. Even though with this upgraded system which are not fully dedicated system and strong. As Colab has limitation to GPU and Time-outs. As well the issues with the storage and processing timing was an concern. When the size of the data increases the processing time also increased and require better system. Hence, the experiment was not performed on more the 100K images. However, this can be overcomed in future with higer specifications.

# **Chapter 2: Related work**

In this chapter, two parts have been discussed related to the literature review. Part one consists of the study related to the work previously performed and completed by different researchers and how their study helped the author to complete his research. Part two consists of the list of concepts and background research work required and study how to implement them for the research work.

## **2.1 Literature Review**

Many of the researchers have performed an ample amount of work with the continuous process of development and advancement in discriminative models Recurrent Neural Networks and Convolutional Neural networks. (Han,2018) have worked on generating realistic images for Magnetic Resonance Brain images using Generative Adversarial Networks. Since studying of medical images requires a lot of data the author used the Data augmentation technique by reconstructing the original images for better performance. The GAN based approach was used for medical data augmentation using Deep Convolutional GAN also known as DCGAN and Wasserstein GAN (WGAN). The author used BRATS 2016 dataset which contained 15,900 MRI images. In the pre-processing images was resized from 240 X 155 to 32 x 32 ,64 x 64 and 128 x 128 pixels for processing. (Han,2018) used the models DCGAN and WGAN for comparing which can be more feasible for medical image generation. For DCGAN implementation, used the same DCGAN architecture without the tanh in the generator model. ELU discriminator, Filters of 4 x 4, and half channel size for training. Hyperparameter: Batch size of 64 and learning rate of 2.0 x 10^-4 Adam optimizer was used. For WGAN, they used the same architecture of DCGAN with batch size of 64 and optimizer as Root mean square Propagation (RMSprop) with learning rate of 5 x 10^-5. After the visual Turing test by physician it was found that WGAN generated more accurate and realistic images while DGCAN likely less suitable for MR images generation (Han,2018).

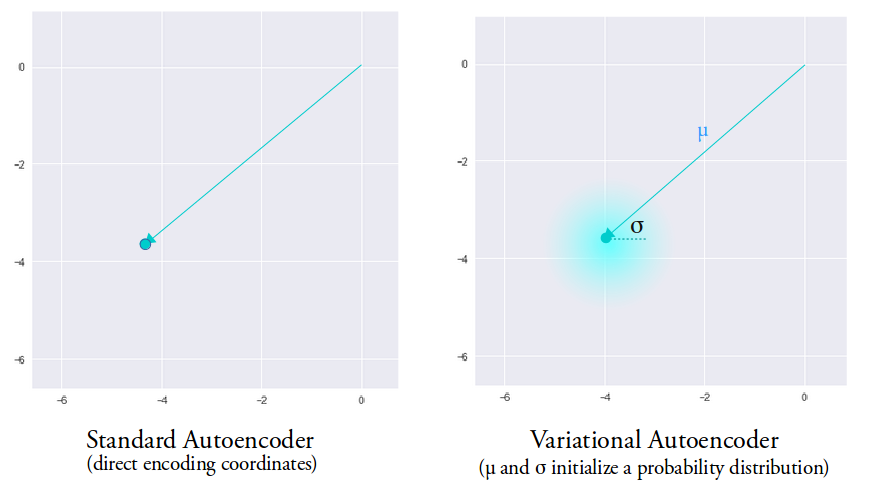
There have been a lot of research work on Generative models, (Liu,Siu and Chan,2019) has implemented his work to Realistic image super resolution using Variational Autoencoder. His work stated that it is possible to generate high resolution images from low-resolution images using generative models. However, Generative networks have limitation with un-realistic sample and mode collapse. Which can be overcome with Image super resolution using Variational Autoencoders (Liu,Siu and Chan,2019). The author has added conditional sampling mechanism for narrowing the latent space for reconstruction. The model generation was evaluated using the KL loss for measuring the divergence between latent vector and standard distribution. Other than using CNN for distortion minimization or GAN the proposal for SR-VAE uses the variational autoencoder for super-resolution. Comparative to other SR algorithms SR-VAE has achieved better performance (Liu,Siu and Chan,2019).

Another research by (ZHANG,TIAN and ZHENG,2018) explains the use of hyper-parameters and the use of a Generative adversarial network. Created a model with a discriminator and generator Network without fully connected layers and implementing batch normalization and deconvolutional networks. In the network architecture, Discriminator the author has used autoencoder used two convolutional layers, and the filter is linearly increased and achieved subsampling by stride 2. Then upsampled by the nearest neighbor method (ZHANG,TIAN and ZHENG,2018). The discriminator is fed with a three-channel image, a feature map is obtained after multiple convolutional operations and then map scalar value and deformed into a vector (ZHANG,TIAN and ZHENG,2018). The author has used Sigmoid as the loss function for the discriminator. The author has used the same architecture for generators with different weights. Random noise is fed to the first layer and multiplied by the weight matrix to transform into three dimensions. The remaining layers are deconvolutional layers and 2x2 stride (ZHANG,TIAN and ZHENG,2018). The activation function used was Relu and the last layer of tanh for generating a three-dimensional image. The loss function based on Wasserstein Distance was used (ZHANG,TIAN and ZHENG,2018). The models DCGAN and WGAN were compared on faces dataset. Setting for the parameter used was 64x64 and 128x128 with a learning rate of 1e-4. Their model has shown better performance and the best recognition rate.

## **2.2** **Kullback-Leibler Divergence (KL):**

The loss function used in VAEs is the KL loss. Understanding this loss is really important as it is a crucial component in reconstructing the new samples for the decoder (Kurt,2017). Firstly we need to understand why we need this loss function. For Variational Auto encoder has some unique properties that separate it from other standard encoders. The VAEs, Encoder has two output vectors of size n which are the Vector of means and vector of standard deviation. This vectors are stochastically generated to pass as an input to the decoder. The mean vector controls the centre where the encoding for the input should be,while the Standard deviation controls the area , how much the encoder vary from the mean. Once the encodings ae generated at random inside the distribution, decoder refers all the near by points as same (Kurt,2017). Ideally we need all the encodings to be as close as possible at the same time it should be distinct which allows smooth interpolation and reconstruct new samples.

KL divergence has its origin from Infromation theory. As the most important parameter in information theory is Entropy.



**Fig.1.1 Probability distribution**

However, Entropy can not define the optimal encoding scheme to achive the compression (Kurt,2017).But the key information obtained is the lower bound of the number of data we need.we have the path to quantify the information in our data. Now we need to know how much information is lost. Thus using KL loss, which have some modifications we can measure the loss. Rather then having just probability distribution p it adds approximating distribution q. which can be represented as,

The KL divergence is the expectation of difference between the probability of original data with respect to approximating data (Kurt,2017). Otherwise, we can write the exquation as the expected information loss,

D(KL)(p||q) = E[log p(x) – log q(x)]

Thus, KL loss allows us to minimize the amount of information we have lost (Kurt,2017). Combining neural networks with KL divergence can be crutial to learn the complex approximating distribution of the data.

## **2.3: Deep Learning and Generative Models**

### **Convolutional Neural network (CNN)**

Computer vision and artificial intelligence have been evolving in the decade very rapidly and have been adopted very quickly due to its outstanding ability to perform the tasks which humans can perform. This has been significantly improving with technology. The major gap between humans and computers is now becoming narrower as there is a great development in Deep learning and machine learning techniques. Using these techniques to improvise machines to make them learn what humans can do. There is rigorous advancement in deep learning with one of the base algorithms well known as the Convolutional Neural network.

A convolutional network is an algorithm for deep learning (Kalchbrenner,Grefenstette and Blunsom,2014)  .Which takes an image as input. For this algorithm, there is no such requirement for pre-processing, unlike other machine learning algorithms. It can classify various images based on image properties. It can easily learn the filters and characteristics of images. It has been evolved based on human brains. Similar to the neurons in the brain. CCN is a multi-layer network which can analyze input images and perform different tasks such as image segmentation, classification, detection, and object detection in a particular image. There are multiple applications and domains where CNN can be used.

CNN comprised of neurons that have weights and biases that can be trained. The neurons receive an array input and perform some dot product. It is a type of neural network designed to work for 1-dimensional, 2-D, and 3-D image datasets. Consider an image of matrix 4 X 4 this can be flattened into a 16 X 1 vector as an input. The role of Convnet is to reduce the data which is much easier to process without losing the features. Any Convnet consists of a Convolutional Layer, Pooling Layer, and fully connected network. The convolution layer is the one where most of the computation on the data is performed. It acts as feature extractor and has a collection of feature map which are discovered from the input image. The conv layer operates along with the ReLu followed by the max or average pooling layer. ReLu is also called as the Activation layer and implemented after each Conv layer as it maintains non-linearity in the scheme. It changes all negative value to zero and to train the model faster it decreases the gradient issues (Kalchbrenner,Grefenstette and Blunsom,2014). The **pooling layer** is used to reduce the spatial resolution of the feature maps since the neurons share the weight (Arya and Singh,2019). The pooling layer is also useful for the extraction of dominant features which are positional and rotational invariant. There are two types of pooling layers namely Average pooling and max pooling.



**Fig. 1.2 Different types of the pooling layer (Saha, 2016)**

1) **Max pooling**: It covers the maximum value of the image which is covered by the kernel. It also used for dimensionality reduction and de-noisy. It can also discard the noisy activation. Max pooling is always better than average pooling.

2) **Average Pooling**: It covers the average value of the image covered by the stated kernel. It can only be used for dimensionality reduction.

Considering fig 1.1 it can be seen how a max layer works as it takes the maximum value from the array of the kernel while the average layer calculates the average of all the values. Depending on the requirement and need for capturing the details in images this layer can be increased if required.

The **flattening** layer is placed in between the two main layers, i.e. convolutional and the fully connected layer (Arya and Singh,2019). Which acts as 2-dimensional data into a single feature vector. In the **Fully Connected Layer,** the neurons are connected with the previous layer neuron. The flattened output is given to the feed-forward network also the backpropagation is applied to each iteration in training. It takes the feature vector as the input and uses the SoftMax layer to classify the image input.

Few of the parameters:

1.**Depth**: Defines the number of filters used in the convolution process for operating output layer neurons.

2.**Stride**: It states the number of pixels used to move across the image of the filter matrix.

3. **Feature Map**: It shows the kernel output applied to the previous layer. The filter is rotated one pixel at moment across the image and results in neuron activation in each place and is stored in the feature map.

4.**Padding**: Padding for a matrix can be achieved by adding zeros in the input matrix. It is required when input volume has to be maintained in the output.

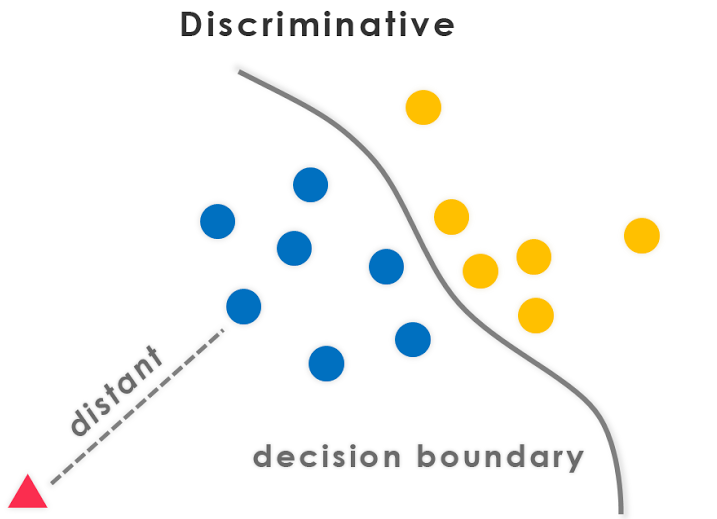
### **Discriminative Vs Generative Models**

In this section we will understand the difference between the Discriminative and generative models. Also the reason why it is nessacry to understand the difference betwwn these models.

While analysing the classifiers which can be suited for operations on your dataset, you will come across multiple classifiers which are limited to Naïve bayes, Logistic regression etc. So its important to understand which classifier will be suited for particular dataset and what can be achived from this analysis.classifiers can be grouped into Discriminator and Generative.

**Discriminator Models:**

Basically, the discriminator learns boundries between the classes (Batra,2016). SVM and Decision trees can learn the explicit boundries hence are discriminative. Discriminative modols do not function as oulier detection, though generative models do. It allows to classify points, without giving a model of how the points are actually generated (Batra,2016).



**Fig.1.3 Discimivative model classification (Batra, 2016)**

1. Probalistic model learns from P(Y|X)
2. Non-probalistic model learns from direct mapping from the points to the classes.

Discriminative models make much fewer assumptions, while generative models make some kind of structure assuptions. This model works better on larger dataset.

**Generative Models:**

This model provides the algorithm of how the data is generated, and discriminative algorithms classify (Batra,2016). This models have discriminative properties, since you may get P(y|x) if you have p(x|y) and p(y) by using Bayes Theorem (Batra,2016). Generative models are classified as probalistic graphical models which can represent the independence dataset relations.

**Fig.1.4 Generative Models**

The Generative model models the actual distribution of each class and learns the joint probability distribution (Datum,2018).

### **Deep Learning Generative Models**

Modernization has changed how humans think, it has brought many changes to our lives, the changes that happened in the field of deep Learning lead to the invention of generative models. The deep generative models are the combination of both generative models and the deep neural network. The variation of deep neural networks with the generative models is, unlike in deep generative models we use the neural networks that have parameters smaller than the amount of data that we train the models. There are different types of generative models, namely, the Gaussian mixture model, the hidden Markov model, the probabilistic context-free grammar, the Bayesian model, the averaged one-dependence estimators, the latent Dirichlet allocation, the Boltzmann machine, the variational autoencoder, the generative adversarial network, the flow-based generative model, and the energy-based model. This research work mainly focuses on the Generative adversarial network and the variational autoencoder as these are the best models that create artificial data.

# **Chapter 3: Methodology**

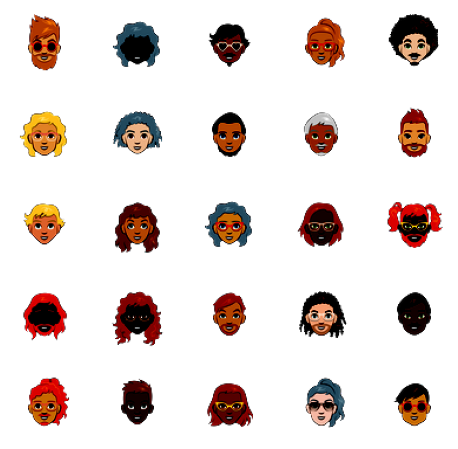
## **Comparision of GANS and VAEs**

In the field of Unsupervised machine learning variational autoencoders (VAE) and generative adversarial network (GAN). Both the networks come under the category of generative models, these models can be described as machines that take input data to return a probabilistic distribution that represents the output. The generative adversarial networks are very good at creating realistic-looking images and generating the artificial data, whereas, the variational autoencoders also generate artificial data, but it renders blurriness that does not look realistic to a human eye. The Variational autoencoders are the models that use variational Bayesian inference function to obtain the approximation of probability density function, while the generative adversarial network works against the adversary. The variational autoencoders work with entities like encoder and decoder, while the generative adversarial neural network works with entities like generator and discriminator. The major difference between the variational autoencoders and the generative adversarial networks is that the generative adversarial neural network works implicitly and the variational autoencoders work explicitly.

## **3.2 Datasets:**

**Cartoon and Logo**

There were two datasets used in the project. One of the datasets includes images of Random cartoon characters which were obtained by GitHub. This dataset consists of two sets one with a 10k dataset and a 100k dataset. Overall, there are 100k images. The sample of the cartoon character figure is below. Cartoon dataset was accessed from the Guithub website. Link for the access is: [**https://google.github.io/cartoonset/download.html**](https://google.github.io/cartoonset/download.html). Another dataset used was the Logo dataset which consists of random Logos of various industries including Food, Medical, transport, sports, etc. The dataset also consists of a total of 100k images in all. Link for Logos Dataset: **https://data.vision.ee.ethz.ch/sagea/lld/** Few of the samples of the logo are shown in the Fig.3.2.



**Fig.3.1 Cartoon Images from dataset.**

****

**Fig.3.2 Logos Dataset Sample**

## **3.3 Generative Adversarial Networks (GANs)**

GANs are the most recent work in deep learning proposed by Ian Goodfellow in the year 2014. It is a unique learning method and a generative model. It can be used for many realistic and challenging tasks like generative real images, Video, and style learning. This is the most efficient class of networks used for unsupervised learning (Goodfellow, Mirza and Xu,2014).

**Basic Theory:**

It is simply based on a game contest wherein the networks or models work in opposition to each other (Goodfellow, Mirza and Xu,2014). These models can determine different variations in the dataset. Developing GAN, it has been observed that most of the neural networks can wrongly predict when a small amount of random noise is added to the original dataset.

Working of GANs:

The working of GANS can be divided into three parts:

1. Generation/Generative: Generative model which generates data in terms of the probabilistic model.
2. Adversarial: Parameters are set to train the model in an adversarial method.
3. Networks: Deep learning neural networks are used to train both the model.

Real Images

dataset

Discriminator

Real or fake

Latent Space

Generated Images

Generator

**Fig.3.3 Structure of GAN.**

In GANs, there are two models named as generator and discriminator. The function of the generator is to generate fake images or data while the job of the discriminator is to identify the fake and real data (Goodfellow, Mirza and Xu,2014). Both models are neural networks and work against each other during the training phase. The discriminator aims to classify whether the data is generated from the generator or real data. It acts as a binary classifier. To become better both the models work to improve their capability to distinguish and generate (Goodfellow, Mirza and Xu,2014). The Generator and discriminator can be represented by any differential function. Hence, these two models can adopt neural networks (Goodfellow, Mirza and Xu,2014). Assume that the discriminator and generator are represented by differential functions D and G respectively, their inputs are real sample S and random variable R respectively. G(R) is the sample generated by G, which follows the distribution of real data. If the input to discriminator is received from real data then it is assigned as one. And if the input sample is G(R) then it is assigned as Zero. To obtain accurate binary classification is the aim for discriminator i.e. Real from real data and Fake from the generated data G(R), While the generator tries to make the generated data G(R) false as the same real data S performing on D(S) (Goodfellow, Mirza and Xu,2014). The performance of both models can be improved by process and optimization. When the discriminator has reached an improved and the data source is not identified correctly, then the generator is considered to learn the actual distribution of the data.

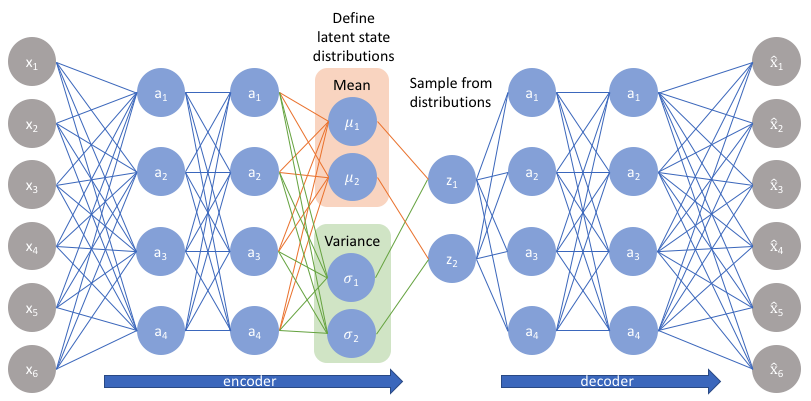
## **3.4 Variational Auto-Encoders (VAEs)**

Variational Autoencoders were introduced by (Kingma and Welling) in the years 2013, which offers and a different approach to generative models by integrating latent variables in the convolutional autoencoders (Doersch,2016). Autoencoders are neural networks whose goal is to copy their inputs to their outputs . The input is compressed into a latent-space representation and the output is reconstructed using the representation. The network consists of two main blocks the Encoder and the decoder.

1. **Encoder**: The Encoder is used to compress the input into the latent-space representation E = f(x).
2. **Decode**r: The function of the decoder is to simplify the decode of the latent space representation to reconstruct the images D =g(h).

Thus, Autoencoder is the sequence of two different functions where the function E converts the input x into a latent space L and the Decoder function uses L to reconstruct data x called r. Mathematically it can be represented as (Doersch,2016),

E(x) = L, D(L) = r



**Fig.3.4 Basic VAEs Architecture (Jordan,2018).**

The basic goal for VAE is to generate realistic images if they are drawn from input data distribution by inputting the noise vector to the decoder (Hou,X,Sun,Shen and Qiu,2019).

## **3.5 Data Pre-Processing:**

Pre-processing of the dataset was done using python on the Jupiter notebook. Python being a powerful language Pre-processing of any type of data is much easier with the extensive libraries. Firstly, the images were loaded into the notebook. For this, the author has used the basic pandas, NumPy libraries. Further, the dataset for cartoon images was been stored in multiple sub-folders with 10 folders each containing 10k images. To read these images **GLOB** has been used to read the images and load. **GLOB** library is one of the useful functionalities as it can read the files from sub-folders containing different types of files. As the cartoon dataset contained different excel files and PNG images. Only images were read through the directory from 10 different folders. This image dataset was then appending and loaded into an array. These images were processed and rescaled to 32x32 and 64x64 size. This file was created into two sets of files with 30k and 60k images. These arrays were then pickled to be used for the GAN and VAE model. Images of cartoon sets were resized into different dimensions.

## **3.6 Model Stucture:**

Images generation for cartoon and logos dataset was performed using two Experimental methods.

1. Generative adversarial Network (GANs)
2. Convolutional Variational Auto-encoders. (VAEs)

### **3.6.1 Model A) GAN**

Generating Cartoon images using GANS. Firstly, the data set was pre-processed in juypter notebook and the training dataset was file was pickled. The Pickled files were then uploaded in Google Drive since the author has used Google Colab for Training the model. There are multiple sources and approaches for performing the GAN models using Pytorch, Fastai Libraries, and TensorFlow and Keras. The researches have selected the TensorFlow approach. Using TensorFlow as backend and Keras on the top. Since the experiment was performed on two datasets slicing of the dataset was required. For the experimental purpose, the dataset was sliced with 30k and 60k respectively where K denoted Thousand. The dataset was pre-processed in python using libraries mentioned in the Libraries section. Once the data was loaded the images were resized to various of 32x32 and 64x64 pixels. The parameter setting is also specified in table 4.3 and 4.4. First for the cartoon data was read from different folders using GLOB and a binary file was generated with re-sizing the images with a slice of 30 and 60k. This pickle file was imported into CoLab for further processing of the images.

### **3.6.2 GAN Specification:**

For the experiment, the models were performed in two different parts for Cartoon and logos dataset. The model created for Generator and discriminator is having two specifications and different parameter settings. First, the models for cartoon datasets are discussed as Part A while for logos dataset is referred to as Part B.

**Part A: GANs on cartoon data and Logos.**

The first step for designing the model was to create a discriminator. As the function of the discriminator is to classify the images as real or fake which are generator by the generator. The input to the Discriminator is the input shape of the images which are defined separately as per the dataset and parameter setting table. While the output is the binary classifier. The normal convolutional layer is used for the discriminator model and uses a stride of 2x2 for the downsampling of the input images. There is no max-pooling layer used in the discriminator model. As there is only a single node in the output layer for predicting whether the output is generated or real sigmoid activation is used. Binary cross-entropy function is used for minimizing the loss of the model. Better practices for using the parameter for the discriminator model are used. The model is initiated with the sequential layer. The first layer of 2-D convolutional is used for the input and added with the Leaky relu layer. Then downsampling is performed using four convolutional Conv2d Layer with different filters and stride of 2x2. padding is set as same for all the four layers. Each of the convolutional layers is added with a LeakyRelu with the value of alpha set to 0.2. After the downsampling layers, the classification layers are added to the model which consists of the flattening layer followed by the Dropout layer with the drop out set to 0.4 and the last layer is the dense layer which is used with the sigmoid function. Which classify the output as generated or real. The optimizer used is Adam with different learning rates and monuments. But for major samples learning rate of 0.0002 and a beta value of 0.5 was selected. The architecture for the cartoon dataset is shown in Table 3.1. The discriminator model needs to be trained on the batch.

Once the Discriminator model is was created, the generator had been defined. The architecture of the generator model is shown in Table 3.2. The generator model aims to create Fake images that need to be learned by the discriminator to classify.

Model Structure for Cartoon with 30K Slice:

Model: "sequential\_5"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_5 (Conv2D) (None, 64, 64, 64) 2368

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_11 (LeakyReLU) (None, 64, 64, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_6 (Conv2D) (None, 32, 32, 128) 73856

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_12 (LeakyReLU) (None, 32, 32, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_7 (Conv2D) (None, 16, 16, 128) 147584

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_13 (LeakyReLU) (None, 16, 16, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_8 (Conv2D) (None, 8, 8, 256) 295168

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_14 (LeakyReLU) (None, 8, 8, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_1 (Flatten) (None, 16384) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 16384) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_4 (Dense) (None, 1) 16385

=================================================================

Total params: 535,361

Trainable params: 535,361

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Table 3.1 GAN Discriminator model for Cartoon images**

Model: "sequential\_7"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_6 (Dense) (None, 4096) 413696

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_19 (LeakyReLU) (None, 4096) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

reshape\_2 (Reshape) (None, 4, 4, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose\_5 (Conv2DTr (None, 8, 8, 128) 524416

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_20 (LeakyReLU) (None, 8, 8, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose\_6 (Conv2DTr (None, 16, 16, 128) 262272

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_21 (LeakyReLU) (None, 16, 16, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose\_7 (Conv2DTr (None, 32, 32, 128) 262272

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_22 (LeakyReLU) (None, 32, 32, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose\_8 (Conv2DTr (None, 64, 64, 128) 262272

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_23 (LeakyReLU) (None, 64, 64, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_13 (Conv2D) (None, 64, 64, 4) 4612

=================================================================

Total params: 1,729,540

Trainable params: 1,729,540

Non-trainable params: 0

**Table 3.2 GAN Generator model for cartoon dataset**

**Below is the structure for Logos dataset:**

Model: "sequential\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_1 (Dense) (None, 4096) 413696

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_4 (LeakyReLU) (None, 4096) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

reshape (Reshape) (None, 4, 4, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose (Conv2DTran (None, 8, 8, 128) 524416

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_5 (LeakyReLU) (None, 8, 8, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose\_1 (Conv2DTr (None, 16, 16, 128) 262272

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_6 (LeakyReLU) (None, 16, 16, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose\_2 (Conv2DTr (None, 32, 32, 128) 262272

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_7 (LeakyReLU) (None, 32, 32, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose\_3 (Conv2DTr (None, 64, 64, 128) 262272

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_8 (LeakyReLU) (None, 64, 64, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_4 (Conv2D) (None, 64, 64, 3) 3459

=================================================================

Total params: 1,728,387

Trainable params: 1,728,387

Non-trainable params: 0

**Table 3.3 for Generator model for Logos Dataset.**

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 64, 64, 64) 1792

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu (LeakyReLU) (None, 64, 64, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_1 (Conv2D) (None, 32, 32, 128) 73856

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_1 (LeakyReLU) (None, 32, 32, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D) (None, 16, 16, 128) 147584

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_2 (LeakyReLU) (None, 16, 16, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_3 (Conv2D) (None, 8, 8, 256) 295168

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

leaky\_re\_lu\_3 (LeakyReLU) (None, 8, 8, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 16384) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout) (None, 16384) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 1) 16385

=================================================================

Total params: 534,785

Trainable params: 0

Non-trainable params: 534,785

**Table 3.4 for Discriminator Model for Logos Dataset**

### **3.6.3 Model B: VAE**

The researcher wanted to test how the different varieties of training set will impact on the VAE model. Therefore, the dataset for cartoon and logos was sliced into different training sets. The total number of tests performed was 30k,60k examples, k denotes thousand. Two separate models were created for the Cartoon and Logos dataset. The slicing and pre-processing of the dataset were done using python using various functions and libraries.

### **3.6.4 VAE Specification:**

Two models were implemented using TensorFlow as backend and Keras on the top. The first model was used for cartoon datasets with for the slices of 30k and 60k dataset respectively. For the first model, **Encoder** was designed in which the first layer was the input layer of the specified shape. The next layer consists of two-dimensional convolutional layers that learn 16 filters with 4x4 kernels. Astride of 2x2 is used, Same padding and activation function as Relu. To obtain mean and variance next batch normalization layer is used to ensure that the stable input is fed to the next conv2d layer. Once more Conv2d layer is added with 32 filters and other parameters remain the same. One more batch normalization layer is used. To flatten the multi-dimensional data into one-dimensional data flattening layer is used. This layer is used since the next dense layer requires the input in that form. Next, the Dense layer is used with 200 neurons the same as the latent space. This is the bottleneck of the encoder as discussed earlier in the related work. Again, one more batch normalization is used. The next two layers are designed to obtain the mean and standard deviation from the first output. These are required to sample the random variables for the points in latent space where the inputs are mapped. **A decoder** has the first input layer known as the decoder input.We need to upsample the point in the opposite order of the encoder. The next layer is the dense layer to convert the latent space into multiple outputs. Then the output of dense is reshaped. This output is then fed to the 2d transposed convolutional layer and batch normalization in the exact opposite order of encoder. Then the last layer is the conv2Dtranspose which will ensure that the number of filters learns equal no of channels. It has activation function as sigmoid and the same padding. With a kernel size of 4. The second model was designed for the Logos dataset with slicing the same as the cartoon dataset. **Encoder** followed the samelayers used for the Cartoon dataset with an additional layer of conv2d and batch normalization. With additional filters and kernel size of 4. The architecture of the model is shown in Table 3.5. **Decoder** followed the same structure with the additional con2d and batch normalization layers. The filter size for the Logos dataset used was different and mentioned in Table 3.5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | VAE | Training samples | Epochs | Batch size |
|  |  |  |  |  |
| Cartoon Dataset | Part A | 30k | 200 | 128 |
|  |  | 60k | 200 | 256 |
| Logos Dataset | Part B | 30k | 100 | 128 |
|  |  | 60k | 100 | 256 |
|  |  |  |  |  |

**Table 3.5 Experimental part A and B on cartoon and logos dataset for image generation**

The decay for the moving average for batch normalization was 0.9 and epsilon was 10^-5. The network used Adam optimizer and the learning rate was set to 0.0002. The loss function used in the decoder was Sigmoid cross-entropy with logits. For the reakyRelu, the slope was set to 0.2 for the leak. Before the training, the model was combined to form the VAE output. The decoder is fed with the encoder as input. Where the input to the encoder is the individual means and deviation values sampled.

**Cartoon 30k Encoder and Decoder Network**

Model: "Encoder"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

=======================================================================

encoder\_input (InputLayer) (None, 64, 64, 4) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_7 (Conv2D) (None, 32, 32, 32) 1184 encoder\_input[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_16 (None, 32, 32, 32) 128 conv2d\_7[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_8 (Conv2D) (None, 16, 16, 64) 18496 batch\_normalization\_16[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_17 (None, 16, 16, 64) 256 conv2d\_8[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_4 (Flatten) (None, 16384) 0 batch\_normalization\_17[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_6 (Dense) (None, 100) 1638500 flatten\_4[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_18 (None, 100) 400 dense\_6[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

latent\_mu (Dense) (None, 120) 12120 batch\_normalization\_18[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

latent\_sigma (Dense) (None, 120) 12120 batch\_normalization\_18[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

z\_sample (Lambda) (None, 120) 0 latent\_mu[0][0]

latent\_sigma[0][0]

=======================================================================

Total params: 1,683,204

Trainable params: 1,682,812

Non-trainable params: 392

**Table 3.6 Encoder for VAE Cartoon images**

Model: "decoder"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

decoder\_input (InputLayer) (None, 120) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_7 (Dense) (None, 16384) 1982464

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_19 (None, 16384) 65536

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

reshape\_3 (Reshape) (None, 16, 16, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose\_5 (Conv2DTr (None, 32, 32, 64) 36928

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_20 (None, 32, 32, 64) 256

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_transpose\_6 (None, 64, 64, 32) 18464

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_21 (None, 64, 64, 32) 128

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

decoder\_output (Conv2DTransp (None, 64, 64, 4) 1156

=================================================================

Total params: 2,104,932

Trainable params: 2,071,972

Non-trainable params: 32,960

**Table 3.7 Decoder for VAE 30k Cartoon images**

## **3.7 Hardware and Software requirement:**

### **3.7.1 Software:**

**Google Colab:**

Google Colab is a notebook that allows you to combine the executable code. The interesting fact is that google Colab is one the platform which provided free GPU and TPU which becomes handy for the researchers and deep learning folks. Since working on deep learning models require a large number of data. Here the researcher has opted for a Colab notebook since there was a huge number of images data to be processed which requires strong hardware to process faster. Also, the generative model and networks need to be trained for a longer time which requires GPU. The number of Epochs required minimum for processing a generative model took around hours for processing. But with Standard GPU and RAM on Colab, it was faster than other systems.

Specification of Google Colab:

GPU: 1xTesla K80, compute 3.7, 2496 CUDA cores, 12GB GDDR5

CPU: Single Xeon Processors @2.3Ghz

RAM: Approximately 12.6 GB Available

Disk: Space of 33 GB

### **3.7.2 Libraries Used**

Since python was used as the base line for coding languge. Multiple python Libraries where used which are listed below.

1. GLOB : This libaray was used for loading and reading the files from the directory. With GLOB it is an easy task to read any type of file from multiple folders. It also contains different commands like wildcard etc to read any particular file.
2. Pandas: Pandas being an common and strong library used for data manipulation and analysis of dataset.
3. TensorFlow: its an open source software library for researcher with high level APIs like keras used for constructing models and architecture.
4. Numpy: Used for image arrays and transformation.
5. Matplotlib: This library was used for visualisation, Ploting images and results.
6. Pickle: It was used to pickle the data which was pre-processed. Pickle is an best option to save the current data. We can pickle any file and use it later when required.
7. PIL (Python imaging Library): PIL is generally used for manupualation of images , to open, save files.
8. Drive: This had been used on the google colab for connecting the google drive for importing dataset files.

# **Chapter 4 Results and Analysis**

## **4.1 Experimental Set-up**

Two types of experiments where performed of using two different techiniques of image generation. This experiments where divided into two part using two different sets of data. Here, the researcher has used Cartoon and Logos dataset.

### **4.1.1 Set-up for Cartoon Dataset:**

For the experiment the cartoon dataset was sliced into two parts with a 30k and 60k data size. This code was executed on google colab using tensorflow. The result of cartoon images where more realistic. The main goal was to generate realistic cartoon images using two different methodology and compare the paramters and generation technique used in both the techniques. Since, both Geneative Networks and Variational Autoencoder functions differently and the paramters need to be carefully design to generate the new patterns. Sarting with the first experiment on Cartoon images dataset with slice of 30k on GAN models. The experimental parameter used are defined in the below table 1.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | Part | Training samples | Epochs | Batch size |
|  |  |  |  |  |
| GAN | Part A | 30k | 200 | 128 |
|  | Part B | 60k | 150 | 256 |
| VAE | Part A | 30k | 200 | 128 |
|  | Part b | 60k | 150 | 256 |
|  |  |  |  |  |

**Table 4.1 Experimental parameter on cartoon with 30k and 60k images.**

As stated in the table the researcher has used slice of 30k images with the dimension of 32x32x4 and 64x64x4. Since the cartoon images were of channel 4 instead of RGB i.e channel 3. The depended variables was the visual quality of the image. The independent variables were the number of training samples and the dimensions of the images.

For VAE, the paramters used are described in the table 4.1 Same set of training was performed on VAE model for cartoon dataset.

### **4.1.2 Set-up for Logos Dataset:**

Logos dataset was also sliced into three types , 30k ,60k . Same experiment was performed on this data similar to Cartoon images. The dimensions for logos images was 32x32x3 , 64x64x3 and 32x32x3 for 30k,60k and 100k respectively. An extra set of data was created for with 100k images for experimental purpose. The below table 4.2 decribes the parameter used for the experiment on Logo dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | Part | Training samples | Epochs | Batch size |
|  |  |  |  |  |
| GAN | Part A | 30k | 200 | 128 |
|  | Part B | 60k | 150 | 256 |
| VAE | Part A | 30k | 200 | 128 |
|  | Part b | 60k | 150 | 256 |
|  |  |  |  |  |

**Table 4.2 Experimental parameter on logos with 30k,60 and 100k images where k denotes thousand**

Experiment for Logos were also performed on all the slice dataset using GANs and VAEs model.

### **4.1.3 Hyper Parameters**

Below are the parameter which were set-up during the training process for the GAN and VAE models.In this section only the summary for the hyper-paramters are stated for reference which was used in the experimental set-up for generating the images.

|  |  |  |  |
| --- | --- | --- | --- |
| Paramters | Configurations |  | Training samples |
|  |  |  |  |
| Generator learning rate  Discriminator Learning Rate  Batch Size | 0.0002  0.0002  128 |  | 30k |
|  | 256 |  | 60k |
| Epochs | 150 |  | 60k |
|  | 200 |  | 30k |
| Optimizer | Adam |  |  |
| Beta value for Optimizer | 0.5 |  |  |
| Loss | Binary Entropy |  |  |
|  |  |  |  |

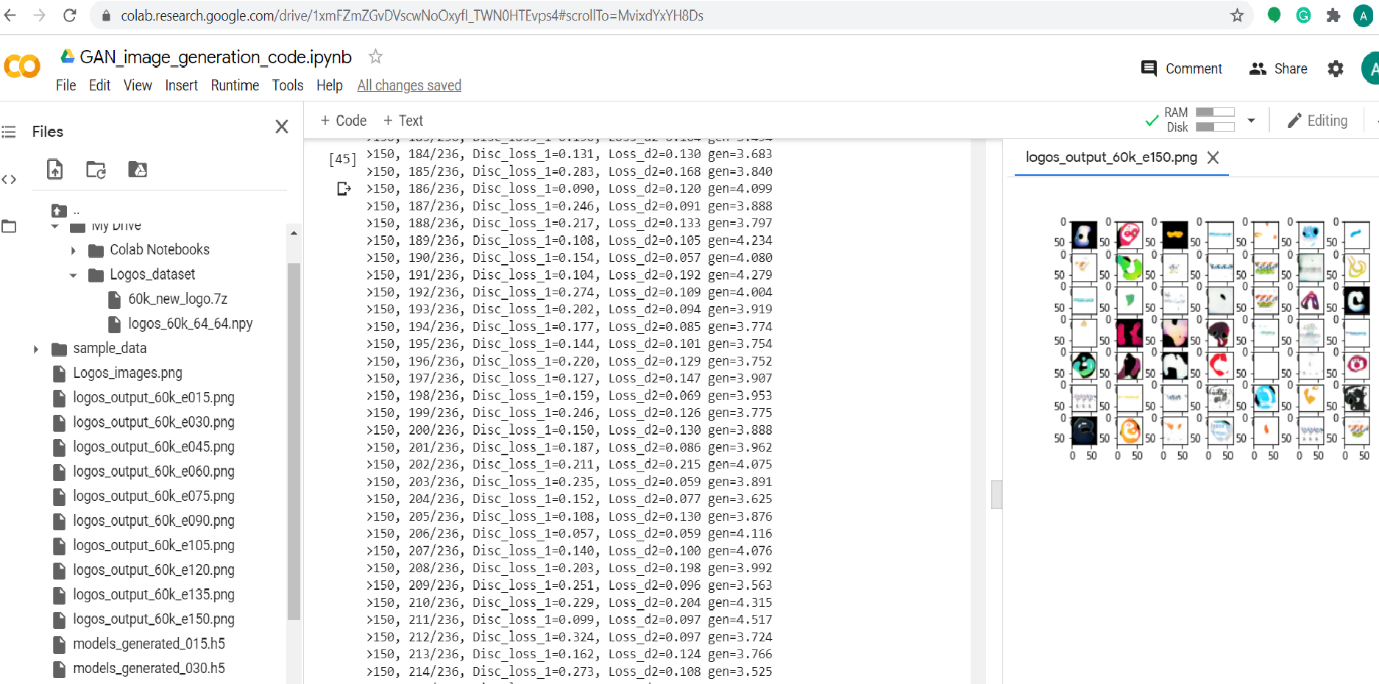
**Table 4.3 Hyper-paramter setting for GAN model.**

The table 4.4 represents the hyper-paramter used for VAE model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Paramters | Configurations |  | Training samples | |
| Learning Rate  Batch Size | 0.0002  128 |  | 30k | |
|  | 256 |  | 60k | |
| Epochs | 150 |  | 60k | |
|  | 200 |  | 30k | |
| Optimizer | Adam |  |  | |
| Beta value for Optimizer | 0.5 |  |  | |
| Loss | KL divergence |  |  | |
|  |  |  | |  |

**Table 4.4 Hyper parameter settings for VAE model**

Sample of Logos Dataset trained with 60k dataset, 150 epochs and batch size of 256 screen shot is attached. Which shows the Saved models, Output images saved and Figures. Below is theattached fig.4.1 for reference.



**Fig.4.1 Details of the Google Colab file , Execution for Logos Dataset images,Results and Saved model.**

### **4.2 Accuracy Measurement:**

Mostly, generative model are difficult to measure as these model generate images which can be classified and identified by human eyes. Most of the generative model can be classified on the basis of the image quality. Here, for generative model the generated images was saved during epochs. For every 10-20 epochs the images generated by the model was saved. Also, the generated models were saved with the epochs. This model saved can generate the images based on the input random noise. The best model can be used to generate images. Also, for the VAEs the model was tested on the different images for cartoon and logos. The test set was created with slice of 10k and 30k with the similar dimensions for both dataset. Table 4.3 describes the different training dataset for VAE model.

|  |  |  |
| --- | --- | --- |
| Method | Part | Training samples |
|  |  |  |
| VAE | Cartoon  Cartoon | 10k  30k |
| VAE | Logos | 10k |
|  | Logos | 30k |
|  |  |  |

**Table 4.5 Test set for cartoon and Logos dataset for VAE**

# **CHAPTER 5 – RESULTS**

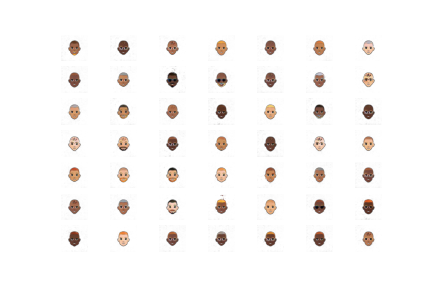
This chapter consist of the results and analysis generated from the the previous chapter of experimental set-up. In this section the results obtained from Generative model from the cartoon and Logos dataset are shown. The generated images are presented based on the parameters and experiment set-up. Also, the images generated using test dataset for VAE model is explained and images are displayed. Furthermore, the section also list the test result of images generated from the model.

## **5.1 Image generation**

In this section, samples of the generated images based on different paramters are displayed. For the different dataset, each subsection in here will display the results from two different models used in the experiment. Firstly, Cartoon images generated using GANs and VAEs are covered. Later in the subsection images of Logo dataset are presented.

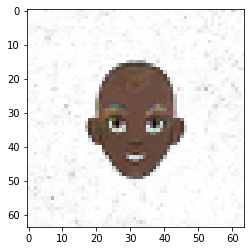
### **5.1.1 Cartoon Images**

Here, the generated images of cartoon dataset are presented. Fig.5.1 shows the image generated using 30k slice of dataset, Number of epochs 200 and batch size of 128 on GAN model.



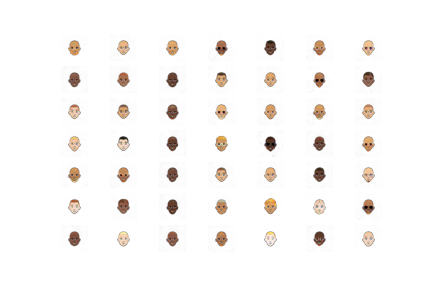
**Fig.5.1 sample of cartoon images generated with 30k sample GANs.**

Then next image is generated using the saved model of GANs. This image was generated using the random noise which was fed to the model. Fig.5.2 desccibe the kind of cartoon character which can be generated.



**Fig.5.2 Images generated using saved model**.

Then the next fig.5.3 shows the ouput for the slice 60k dataset,number of epochs 150 and batch size of 256.



**Fig.5.3 Images generated 60k slice,150 epochs and batch size of 256**.

**VAE Model:**

Now, this section displays the result from VAE models on cartoon dataset. The results displays the images on 30k and 60k slice with different paramters. The images consist of the original images and the generated images. Refering to fig.5.4 show the images with 30k slice, Epochs 200 and batch size of 128 .



**Fig.5.4 Generated images using 10k slice test dataset.**

The fig.5.4 was generated using the training dataset of 10k. These are predicticted images based on the test dataset. It depicts the Original images on the top and predicted images below it in the image.

The fig.5.5 was generated using training dataset of 30k slice. This model was earlier trained on dataset of 60k,epochs of 150 and batch size of 256. The above images are the original images from the dataset and below are the predicted images/generated images by the VAE model.

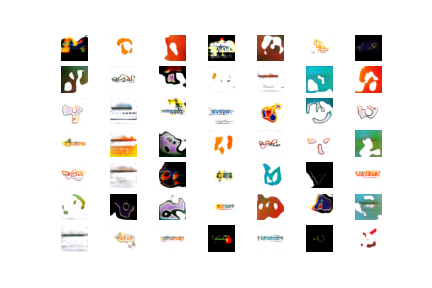


**Fig.5.5 Images generated using slice of 60k,epochs 150 and batch size of 256. These images are predicted based on 30k slice data for training**

Thus, the result from both the models are displayed for cartoon characters with different paramters and slices of dataset . Also the validation for generated images are measured based on testing slice dataset with 10k and 30k images whose output for VAE models are shown. Also, GAN models are measures with the saved models and have resulted in unique cartoon character.

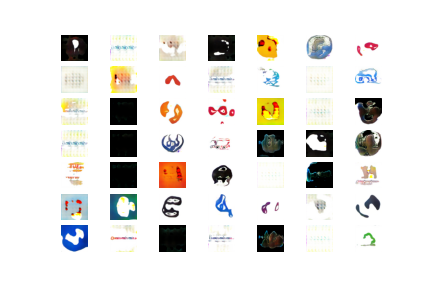
**Logo Dataset:**

Here, the images generated using GANs model are presented. Fig.5.6 represents the images generated with the slice of 30k ,Epochs 200 and batch size of 128. Followed by fig.5.7.



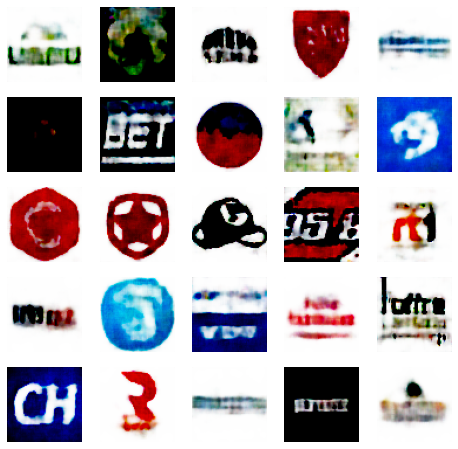
**Fig.5.6 logos generated using GANs model with 30k slice, 200 epochs and batch size of 128.**

The next image shown in the fig.5.7 is generated using 60k slice, number of epochs 150 and batch size of 256. These results were obtained by GAN model.



**Fig.5.7 Logos generated by GAN with 60k slice, 150 epochs and batch size of 256**

Images generated using VAE models are shown in the fig.5.8 these images were generated based on the experimental set-up with 30k slice ,150 epochs and batch size of 128.



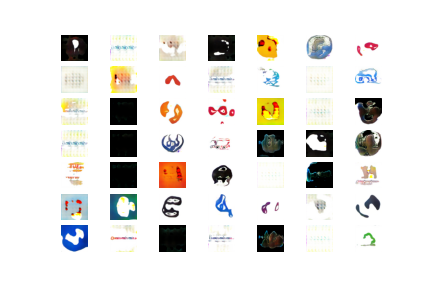
**Fig.5.8 Generated using VAE model with 30k slice , 150 epochs and batch size of 128.**

The fig.5.9 is generated using the saved model for Logos dataset using GAN model. This was the logo generated based on sample noise. Which is much more realistic.

****

**Fig. 5.9 Logo Generated using GAN model. 60k images and 150 epochs**

The fig.5.10 represents the output of the model for VAE, Generated using 60k,epochs 150 and batch size of 256. The images are more realistic and shows a better resolution.



**Fig.5.10 Generated logos with 60k,epochs 150 and batch Size of 256.**

# **CHAPTER 6: Anaysis and Disscussion**

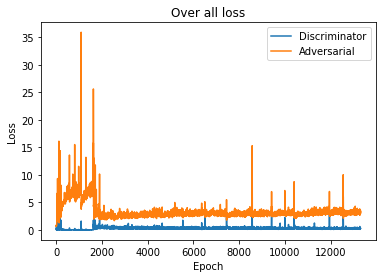
During the experimental set-up lot of variations on the model and dataset were performed. This aspects are carefully analyzed after each experiment for different sets of parameter. The interpretation of data has been discussed below in the sub-sections. This also included the comparision between both GANs and VAEs.

## **6.1 Generation of Images:**

The results for two different dataset had performed differently on both the models used in the experiment. However, Cartoon dataset was reasonably easier for generating more realistic images which can be seen the fig.5.1. Actually the results were good even when a sample of 10k images where used for GAN model. Ofcourse with the training sample increased the GAN model become better in generating images. The researcher observed that with the setting of 30k images and batch size of 128 and 150 epochs the images generated were realistic and similar to the ground truth. However minor errors were visible. Best results were obtained when the paramters and dataset was increased to 60k, with batch size of 256 and no of epochs 200. Since google Colab was specifcaly used for performing the models as it required GPU and RAM. However, generative model uses GPU and without the use of GPU the processing time is effectively more compared to when used GPU. For 60k images with 200 epochs the model took 4 hours of time to train and give the generated images. The images generated had minor error but results were observed to be much better.

Again, images for cartoon dataset generated using VAE model was much more easier. However, the same paramters were used similar to GAN. Results of the VAE model with 30k images , batch size 128 and epochs 200 generated better images which can be seen in the fig.5.4 The researcher observed that the visual quality of the images can be greatly affected by the batch size. When the batch size if set to 128 compared to 256 the visual quality is more superior. Moreover, when the training set is small number of training epochs also have an great impact on the quality of the results.

When it comes to the logo dataset, The GAN and VAE model had some complications to generate realistic images smilar to the ground truth. There are multiple reasons for the complication. Fistly, the dataset has some noise. Also, if the dataset is more explored then it was noticed that most of the images contains plain text rather then images or symbols. The dataset was therefore tested on multiple slice. For 30k slice , batch size 128 and epochs 200 the images with GAN model can be seen in fig.5.8 these images are much better but are not of super quality. Another slice with 60k images was performed with showed much better improvement with 256 batch size and 150 epochs. Images generated using VAE model the training is stable.The over all loss is shown in the fig.6.1 for the GAN for Adversarial and discriminator loss during the training which shows the loss was higher in the initial stages of the epochs and then gradually was stable through out the training. However the image generated by VAE for Logos dataset were blur when the datasize was 30k. However with larger dataset and 150 epcochs the generated logo were less blurry and can be identified as some sort of new logo creation. The image quality is not super since the traing time and epochs was perofmed with in limited space and time frame. This can be improved further with large dataset and increasing the number of epochs how ever this will required much more stable hardware and specification and GPU.



**Fig.6.1 Over all Loss for GAN logos.**

# **CHAPTER 7 CONCLUSIONS AND FUTURE WORK**

In this chapter, the author has drawn the conclusion based on the experimental set-up for both the generative model. Also, the possible future work is proposed in section 7.2

## **7.1 Conclusions**

The research work has been based on the generative models for generating realistic images. Which included the two different set of data. One being an simple dataset of cartoon comparative to Logos dataset since logos images had text and symbolic representation. When compared with the GANs images generated were very much realistic and similar to ground truth. Also, the GANs model may requires large number of training samples depending on the dataset to be more realistic. On the other hand, Pre-processing of data also plays a big roll on the performance. GANs model however required more training time with compared to the VAEs model with similar parameters. Also, images generated with VAE model very trained much faster and resulted in realistic outputs for cartoon dataset. Images for Logos images trained with VAE were observed to be blurry and smooth edges as it’s the drawback of VAE architecture. However, Logos images with GANs showed a better performance and much realistic.

The author has compared the two models and its analysised that GANs perfomed well the VAEs. However, GAN models had problem some problems with the stablisation their training. Also, the parameter are sensitive and can cause lot of changes if are tunned wrongly. The GANs require lot of computation power and with this the large computation the training time and process will be more time consuming.Which makes difficult to fine tune the model. However, GANs are more powerful and are efficiently used in task such as image generation, Text to image translation reconstruction of 3D object from images and many more, Seeing the future of Deep learning, GANs and VAEs will be play an crucial role in desiging fields,Graphics and arts.

## **7.3 Knowledge Gained**

Research work on the topic have been helpful to gain knowledge on the modern deep learning technique. The Generative models are gaining popularity in the recent years and still there is lot of research work been in progress. This pices of work has enlighten my vision and knowledge to an great extend. Now I am well aware to the variations and conepts of deep learning generative models. How this area can be explored and areas can be the future of Deep laraning and Artifical Intellegence. The basic generative models , the working and principle of these models are learned in the research work. As known deep learning generative models has given many of the innovative. Since, Generative model requires a high specification and GPUs. I have worked with Colab. Also, explored different libraries for GANS and VAEs. How the Models internally works , the sensitivity and limitations of the generative models. Comparing the different generative models have enlighted me with the specific use and sectors were these models are best suited and how this can be utilised to the maximum capacity for the requirements.

## **7.3 Future Work**

In the research, different model and architecture was used for generating images. Using the GANs and VAEs images have been generated and which proved to be effective with realistic effects. However, these can be more tuned and different architecture can be used for generative more realistic images. Moreover, This two models can me combined and hybrid model can be constructed for better results and images. Also, cartoon and human faces can be merged and new images can be generated. In Future work, Application can be created for generating Logos online according to the sectors and area of the company. This logo can be actioned online. Customers can you the website to design and generated similar logos which are already famous in industries. Moreover, improvising the architecture can be more effective to get realistic images and using hybrid architechure can be an possible way. Also, with Generative model there are a number of application which can be implemented and smart application can be generated. This sector being a large scope for development and research deep learning will play an significant role to automate technology in coming future.

# **References**

1. Arya, S. and Singh, R. (2019). ‘*A Comparative Study Of CNN And Alexnet For Detection Of Disease In Potato And Mango Leaf - IEEE Conference Publication’*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8977648> (Accessed 7 July 2020).
2. Batra, D. (2016). ‘*Discriminative Vs Generative Models’*. [online] Deveshbatra.github.io. Available at: <http://deveshbatra.github.io/Generative-vs-Discriminative-models/> (Accessed 1 August 2020).
3. Bombuwala, G. and Poravi, G. (2019).  ‘A Review Of Generative Image Modeling Techniques’ - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/9033846> (Accessed 8 July 2020).
4. Cole, F., Et al. ‘*Cartoon Set: An Image Dataset Of Random Cartoons’*. [online] Google.github.io. Available at: <https://google.github.io/cartoonset/download.html> (Accessed 15 May 2020).
5. Datum, D. (2018). ‘*Deep Learning In 5 Minutes Part 3: Discriminative Vs Generative Models’*. [online] Medium. Available at: <https://medium.com/@data\_datum/deep-learning-in-5-minutes-part-3-discriminative-vs-generative-models-8b27c616b8af> (Accessed 5 August 2020).
6. Doersch, C. (2016). ‘*Tutorial On Variational Autoencoders’*. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/1606.05908.pdf> (Accessed 9 July 2020).
7. Gonog, L. and Zhou, Y. (2018). ‘*A Review: Generative Adversarial Networks - IEEE Conference Publication’*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8833686> (Accessed 5 July 2020).
8. Güzel Turhan, C. and Sakir Bilge, H. (2018). ‘*Recent Trends In Deep Generative Models: A Review - IEEE Conference Publication’*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8566353> (Accessed 29 July 2020).
9. Hubens, N. (2018). ‘*Deep Inside: Autoencoders’*. [online] Medium. Available at: <https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f> (Accessed 10 July 2020).
10. Han, C. (2018). ‘GAN-Based Synthetic Brain MR Image Generation - IEEE Conference Publication’. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/8363678> (Accessed 22 July 2020).
11. Hou, X., Sun, K., Shen, L. and Qiu, G., (2019). ‘*Improving Variational Autoencoder With Deep Feature Consistent And Generative Adversarial Training’*.[online] Arxiv.org. Available at: <https://arxiv.org/pdf/1906.01984.pdf> (Accessed 19 July 2020).
12. J. Goodfellow, I., Mirza, M. and Xu, B. (2014). ‘*Generative Adversarial Nets’*. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/1406.2661.pdf> (Accessed 10 July 2020).
13. Jordan, J. 2018. ‘*Variational Autoencoders.*. [online] Jeremy Jordan’. Available at: <https://www.jeremyjordan.me/variational-autoencoders/> (Accessed 20 July 2020).
14. Kos, J., Fischer, I. and Song, D. (2018). ‘*Adversarial Examples For Generative Models - IEEE Conference Publication’*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/8424630> (Accessed 4 August 2020).
15. Kalchbrenner, N., Grefenstette, E. and Blunsom, P. (2014). ‘*A Convolutional Neural Network For Modelling Sentences’*. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/1404.2188.pdf> (Accessed 19 June 2020).
16. Khan, S., Hayat, M. and Barnes, N. (2018). ‘*Adversarial Training Of Variational Auto-Encoders For High Fidelity Image Generation - IEEE Conference Publication’*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8354252> (Accessed 1 August 2020).
17. Kurt, W. (2017). ‘*Kullback-Leibler Divergence Explained — Count Bayesie’*. [online] Count Bayesie. Available at: <https://www.countbayesie.com/blog/2017/5/9/kullback-leibler-divergence-explained> (Accessed 15 August 2020).
18. Liu, Z., Siu, W. and Chan, Y. (2019). ‘*Photo-Realistic Image Super-Resolution Via Variational Autoencoders - IEEE Journals & Magazine’*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9121327> (Accessed 28 July 2020).
19. MissingLink.ai. (2015). ‘*Convolutional Neural Network Architecture: Forging Pathways To The Future - Missinglink.Ai’*. [online] Available at: <https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-architecture-forging-pathways-future/> (Accessed 12 August 2020).
20. Mehrasa, N., Et al. (2020). ‘A Variational Auto-Encoder Model For Stochastic Point Processes’. [online] arXiv.org. Available at: <https://arxiv.org/abs/1904.03273> (Accessed 19 July 2020).
21. Rosca, M., Lakshminarayanan, B., Warde-Farley, D. and Mohamed, S., (2017). ‘Variational Approaches For Auto-Encoding Generative Adversarial Networks’. [online] arXiv.org. Available at: <https://arxiv.org/abs/1706.04987> (Accessed 22 July 2020).
22. Sage, A., Agustsson, E., Timofte, R. and Van Gool, L. (2017).  ‘*LLD - Large Logo Dataset’*. [online] Data.vision.ee.ethz.ch. Available at: <https://data.vision.ee.ethz.ch/sagea/lld/> (Accessed 11 June 2020).
23. Saha, S. (2016). ‘*A Comprehensive Guide To Convolutional Neural Networks — The ELI5 Way’*. [online] Medium. Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> (Accessed 6 July 2020).
24. Semeniuta, S., Severyn, A. and Barth, E. (2017). ‘*A Hybrid Convolutional Variational Autoencoder For Text Generation’*. [online] Arxiv.org. Available at: <https://arxiv.org/pdf/1702.02390.pdf> (Accessed 9 July 2020).
25. Shafkat, I. (2018). ‘*Intuitively Understanding Variational Autoencoders’*. [online] Medium. Available at: <https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf> (Accessed 9 August 2020).
26. Vishwakarma, D. (2020). ‘*Comparative Analysis Of Deep Convolutional Generative Adversarial Network And Conditional Generative Adversarial Network Using Hand Written Digits’ - IEEE Conference Publication*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/9121178> (Accessed 30 July 2020).
27. W3schools.com. (2018). ‘*Introduction To Numpy’*. [online] Available at: <https://www.w3schools.com/python/numpy\_intro.asp> (Accessed 5 June 2020).
28. ZHANG, T., TIAN, W. and ZHENG, T. (2018). ‘*Realistic Face Image Generation Based On Generative Adversarial Network - IEEE Conference Publication’*. [online] Ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9067742> (Accessed 1 August 2020).

# **Appendices**

This section will guide through over all content of the Artifacts and the process to followed to implement the python code for the disseratation project names “A comparative study on Generative Adversarial Network and Convolutional Variational Auto-encoder for generating cartoon and logo images”.

**Artefact Contents**

1. **Python code:**

* **VAE\_Images\_generation\_code.html**: Contents the python code generation of images on both the dataset using Variational Auto-encoder generative model. All the results are displayed int file.
* **GAN\_Images\_generation\_code.html**: Contents the python file for generation of images using Generative Adversarial Nertworks. This file contains the total code for Generator and Discriminator model.
* **Data Pre-processing GAN and VAE (Logos).html:** This file contains the pre-processing of images of Logo dataset. Reading the images from directory, Creating Binary and Pickle files which are used later for generation.
* **Data Pre-processing GAN and VAE (Logos).html:** This file also contains the data pre-processing, Reading and creating dump files as per the experimental set-up and slices of the dataset.

2. **Results GANS:**

* **Outputs\_cartoon\_dataset\_30k\_images**: This folder contents the results generated using GANs with the dataset of slice 30k. This has the images generated and the saved models during the epochs.
* **Outputs\_cartoon\_dataset\_30k\_images**: This folder contents the results with 60k dataset. Images and saved models.
* **Outputs\_logos\_dataset\_30k\_images**: This folder holds the result images and models for logos dataset with slice of 30k.
* **Outputs\_logos\_dataset\_60k\_images**: This folder contains the Logo images and saved models. This results were generated using slice of 60k Images from Logos dataset.

3. **Datasets:**

* Cartoon Character: The dataset was not uploaded in the artifact as the size of dataset when zipped was more then 2GB. The link for the dataset has been provided in the Dataset Section and references. Site for downloading the dataset: <https://google.github.io/cartoonset/download.html>
* Logos Dataset: Similar to the Cartoon dataset , size of logos dataset was 2GB. Hence the link for reference has been provided:

https://data.vision.ee.ethz.ch/sagea/lld/

4. **Readme:** This file contains the steps and details of the file in artifact, how to implement the python code.

**Note:** Please refer to the readme file in the artifacts to understand the use of python files, dataset and other files mentioned. It also explains the different planforms used for executing the python files.

## **LIST OF ABBREVIATIONS**

1. **GAN** Generative Adversarial Networks.
2. **VAE** Variational Auto Encoders
3. **GPU** Graphics Processing Unit
4. **CNN** Convolutional Neural Networks.
5. **DCGAN** Deep Convolutional Generative Adversarial Network
6. **MLP** Multi-Layer Perceptron
7. **KL** Kullback-Leibler Divergence
8. **ReLU** Rectified Linear Unit
9. **VM** Virtual Machine
10. **K Thousands**
11. **BS Batch Size**
12. **Opt Optimizer**