University Of Khartoum

Image to image translation using generative adversarial network

Khalid Adil Mohammed Ahmed
144027

Supervised by
Dr. Hiba Hassan

A thesis submitted in partial fulfillment of the requirements for the degree of
B.Sc (HONS) Electrical and Electronic Engineering
(SOFTWARE ENGINEERING)

Faculty of Engineering
Department of Electrical and Electronics Engineering
October 2020

Declaration of Authority

I declare this report entitled "**Image to image translation using generative adversarial network**" is my own work except as cited in references. The report has been not accepted for any degree and it is not being submitted currently in candidature for any degree or other reward.

Name: _____

Signed: _____

Date: _____

Acknowledgements

I would like to thank my supervisor Dr.Hiba Hassan for offering her guidance and support during this project, thanks to her advises and supervision, this work saw the light. It was such a great honor to work with her.
I would also like to thank Engineer.Mustafa Algaly who used his wide technical experience to guide us throughout this project.
I also want to express my sincere gratitude to my colleagues,my seniors, and my Juniors for their emotional and technical support.
Special appreciation for my project partner Khalid Fathalrehman for his dedication, persistence and patience during this project.
Finally i should thank every single professor,doctor and tutor in University of Khartoum because they had the major role in giving us the basics upon which we build this work and hopefully many other projects to come.

Dedication

I dedicate this work to the ones who dedicated their life for me,i dedicate this work to the ones who lived their life teaching, motivating , inspiring and helping me in every single aspect to become the best possible version of me. This study is wholeheartedly dedicated to my beloved parents.

Abstract

Sketches are an excellent technique for visual explanation, criminals identification, design prototyping and many other useful applications. Since sketches have relatively low details, relying on human drawing these sketches require considerable skill from the human operator to make sure it represents the actual idea they want to deliver. Thus an automated approach for converting sketches -regardless of their detail level- to actual realistic images is needed. In the past years many solutions for converting sketches to images were proposed an important one of which is the Pix2Pix model which uses adversarial training to generate realistic looking images, in this work Pix2Pix model was reviewed and modified to reproduce previous results with considerably less resources and to generalize the model to more complex data.We used three different approaches, in the first one we ran Pix2Pix model without modifications on a low-resource machine, which achieved impressive results proving the flexibility of the model. Learning from the hyper-parameters selected in the first approach, we continued in the second one first by automatically removing background of face images using a pre-trained model then we perform five different experiments that extensively analysis the model to determine which configuration produces best results, one of these experiments consisted of modifying the model by adding an extra input layer so as to add random noise which helps in making the model generalize more easily. Results show that our modified approach achieves better inception score, less generator loss and cleaner, more crisp output images. Finally in the last approach we go a step further by trying to generate realistic human images knowing only the pose they were in, eventhough we did the necessary pre-processing by converting the images to poses using the pre-traines OpenPose model, our model failed to generate realistic results because the dataset acquired by concatenating CoCo images with the output of OpenPose was too noisy. The results obtained during the development of the three approaches showed the effect of different hyper-parameters selection on the generation accuracy of the Pix2Pix model, and it also highlight the advantages and limitations of each approach to point out possible further improvements.

المستخلص

تعد الرسومات التخطيطية تقنية ممتازة للشرح المرئي وتحديد المجرمين وتصميم النماذج الأولية والعديد من التطبيقات المفيدة الأخرى. نظراً لأن الرسومات تحتوي على تفاصيل منخفضة نسبياًا ، فإن الاعتماد على الرسم البشري لهذه الرسومات يتطلب مهارة كبيرة من المشغل البشري للتأكد من أنها تمثل الفكرة الفعلية التي يريد تقديمها. وبالتالي هناك حاجة إلى نهج آلي لتحويل الرسومات - بغض النظر عن مستوى تفاصيلها - إلى صور واقعية فعلية. في السنوات الماضية ، تم اقتراح العديد من الحلول لتحويل الرسومات إلى صور ، ومن أهمها نموذج (بيكس-تو-بيكس) الذي يستخدم التدريب العدائي لتوليد صور ذات مظهر واقعي ، وفي هذا العمل تمت مراجعة وتعديل النموذج لإعادة إنتاج النتائج السابقة بموارد أقل بكثير و لتعميم النموذج على بيانات أكثر تعقيداً. استخدمنا ثلاث طرق مختلفة ، في الطريقة الأولى قمنا بتشغيل نموذج (بيكس-تو-بيكس) بدون تعديلات على آلة منخفضة الموارد ، مما حقق نتائج مبهرة تثبت مرونة النموذج. تعلما من المعلومات المحددة في النهج الأول ، واصلنا في الطريقة الثانية أولاً عن طريق إزالة خلفية صور الوجه تلقائياًا باستخدام نموذج مدرب مسبقاًا ، ثم اجرينا خمس تجارب مختلفة تحلل النموذج على نطاق واسع لتحديد العوامل التي تنتج أفضل النتائج ، إحدى هذه التجارب عبارة عن تعديل النموذج بإضافة طبقة إدخال إضافية لإضافة ضوضاء عشوائية تساعد في جعل النموذج يتعمم بسهولة أكبر. تظهر النتائج أن أسلوبنا المعدل يحقق نتيجة بداية أفضل ، ويقلل من فقد المولد ، وصور إخراج أكثر وضوحاًا . أخيراً في النهج الأخير ، ذهبنا خطوة إلى الأمام من خلال محاولة إنشاء صور بشرية واقعية بمعلوميه الوضع الذي كانوا فيه فقط، على الرغم من أننا قمنا بالمعالجة المسبقة اللازمة عن طريق تحويل الصور إلى أوضاع باستخدام نموذج (اوبن-بوس) فشل النموذج في تحقيق نتائج واقعيه لان البيانات كانت مليئه بالضوضاء. أظهرت النتائج التي تم الحصول عليها أثناء تطوير الأساليب الثلاثة تأثير اختيار المعلمات المختلفة على دقة التوليد لالنموذج ، كما أنها تسلط الضوء على مزايا وقيود كل نهج للإشارة إلى التحسينات الإضافية الممكنة.

NN	Neural Network
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
ConvNets	Convolutional Neural Network
RELU	Rectified Linear Unit
GAN	Generative Adversarial Network
SBIR	Sketch Based Image Retrieval
ROI	Region of Interests
MNIST	Modified National Institute of Standards and Technology
CBIR	Content Based Image Retrieval
SVM	Support Vector Machine
VAE	Variational Autoencoder
DCGAN	Deep Convolutional Generative Adversarial Network
PSNR	Peak Signal to Noise Ratio
SSIM	Structural Similarity Index Measure
FID	Frechet Inception Distance
IS	Inception Score
CPU	Central Processing Unit
GPU	Graphical Processing Unit
API	Application Programming Interface
CGAN	Conditional Generative Adversarial Network
SGD	Stochastic Gradient Decent

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an insight into the field of image to image translation using generative adversarial networks, the motivation behind this work, a brief description of the methodology, the objective of the study. In addition, the thesis layout, and what we intend to do with this project.

1.1 Motivation

Visual explanation plays an indispensable role in today's life, because a small drawing with a little amount of details can make people understand exactly what you are talking about, that's why sketches are very important in today's life. Particularly with the exploding development of mobile devices, tablets and other touchscreen devices, sketches are easy to draw, that what made considerable re-searchers focus on how to optimize the use of these sketches.
Being able to convert these low-level drawings to high-level images with significantly more details will definitely help in many tasks that consider sketches essential.
An example of where this might be useful is that police used to sketch criminals and distribute these sketches all around the city in order to identify the criminal, now they can convert these sketches to real images and run a search on their database hoping for a match , also home designers may draw a quick sketch and convert them to images in order to give the customer a better understanding of the design, other examples include fashion design or any other form of visual description.

1.2 Problem Statement

The purpose of this project is to develop a system that automatically converts small details drawings to an actual image that can be used to describe anything more thoroughly. Also to provide a detailed analysis of the different methods used to tackle the problem and evaluate which method will yield better results.

1.3 Methodology

The focus of this work is to use deep learning models such as convolutional neural networks and transpose convolutional neural networks with three different approaches to convert low level representation of an image to a high level representation of that same image. Each approach was extensively analyzed and evaluated.

1.4 Objectives

- To perform a literature survey of previous methods used in the field of image to image translation and classify them based on the method used.
- To automate the activity of converting drawings to images by developing a robust image generation system utilizing deep learning algorithms.
- To evaluate different deep learning methods used in the development process.

1.5 Thesis Layout

The rest of this thesis is structured as follows:

- **Chapter 2 - Literature Review:**
  Introduces a survey of previous researches in the field and presents some theoretical background knowledge needed to understand the work.
- **Chapter 3 - Methodology:**
  Provides a detailed description of the three approaches being used in this study along with their implementation.
- **Chapter 4 - Results and Discussion:**
  Presents and discusses the results and finding obtained during training the model with different approaches.
- **Chapter 5 - Conclusion:**
  Provides the conclusion and an insight into the the possible future work.

1.6 Project publication

We intend to present this thesis as a paper in the IEEE annual conference which will be held in October 2020.

Chapter 2

Theoretical Background & Literature Review

2.1 Theoretical Background

This section will discuss the basics upon which we build our work, here we will only talk about the basics of the basics. Next section will also include some background specific to the technique under discussion.

2.1.1 Neural Networks (NNs/ANNs)

Neural networks are a set of algorithms, modeled loosely after human brain, that are designed to recognize patterns. They interpret sensory data through a king of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

2.1.2 Convolutional Neural Networks (CNNs/ConvNets)

Convolutional Neural Networks (CNNs) are a special class of artificial neural networks (figure 2.1), and it's inspired by the natural visual perception mechanism of the living creatures.

ConvNets are regularized versions of multilayer perceptron, and it consists of an input and output layer, and it has multiple hidden layers called convolutional layers that performs a mathematical operation called convolution which is a special kind of linear operation, and its activation function is a RELU layer.

The CNNs went through many phases starting by its first well known architecture the LeNet [1] for hand written digits classification in 1998, then it had many advancements in the general network architecture which led to the famous AlexNet [2] in 2012, ZFNet [3] in 2013, VGGNet [4] in 2014, GoogleNet [5] in 2015, Microsoft ResNet [6] in 2015.

CNNs have been mainly used in image classification for a long time. Compared to other methods, CNNs can achieve surprisingly better classification accuracy on large scale datasets due to their capability of joint feature and classifier learning, and it also has other applications such as face recognition [7], object tracking [8], pose

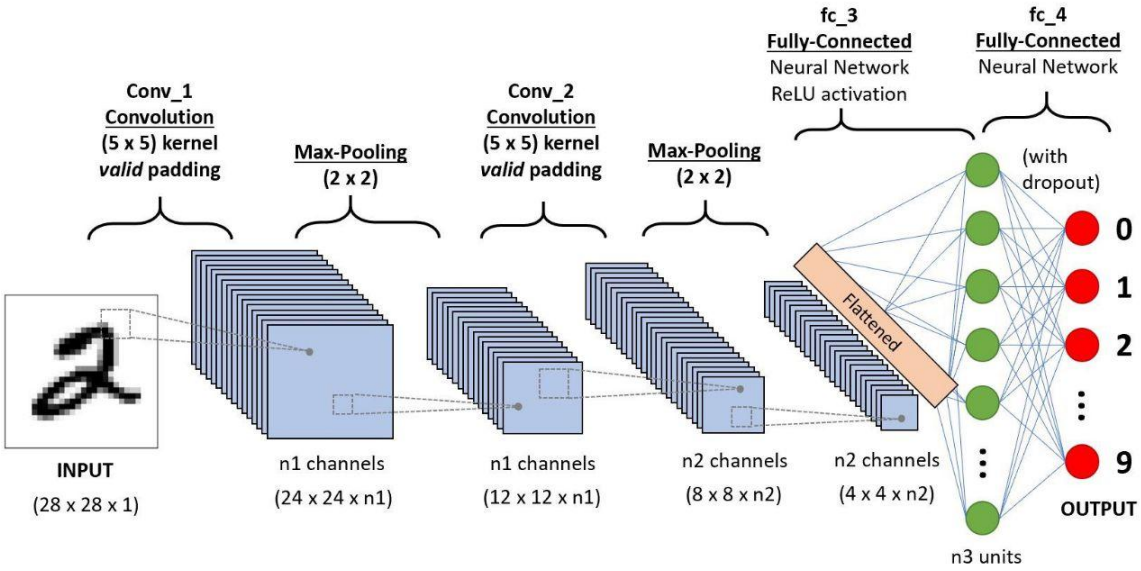


Figure 2.1: Illustration of a CNN architecture

estimation [9], text detection[10], visual saliency detection [11], action recognition [12], scene labeling [13], speech and natural language processing [14].

2.2 Literature Review

Sketches are intuitive models used for communicating everyday concepts ranging from describing a cat to a child to describing how space ships work, this has been the subject of many different studies because it would be convenient and time saving if we could turn low-level sketches drawn in no time to a realistic detailed image that describes the concept we are trying to communicate precisely.

Apart from generative techniques using GANs [15] some research have been conducted in sketch based image retrieval systems, these systems primarily depend on feature extraction using CNNs because the more complete the extracted sketch features are, the more the real content of the sketches can be expressed, then the hand-drawn sketch can be matched accurately [16].

Different approach suggested by Tao Chen et al [17] was a system that composes a realistic looking images from a simple free-hand sketch annotated with text as labels, these images are found by doing a quick online search, Tao Chen et al were able to overcome the fact that internet search sometimes results in an inappropriate results -such as the case when you search for "crane" meaning the bird but it results in "crane" meaning the heavy lifting machine- using a filtering scheme to exclude unwanted images.

All systems mentioned above were able to convert a low-level sketch to a detailed image to some extent. However, although generative models are tricky to train, they have many advantages over other techniques including that they don't search a database or the internet; they generate their own unique never seen image that matches the desired characteristics.

So, in short, the main techniques used for converting sketches to images are:

- Sketch based image retrieval (SBIR)
- Internet montage
- Generative models (GANs)

2.2.1 Sketch based image retrieval techniques (SBIR)

2.2.1.1 Overview

The sketch-based image retrieval (SBIR) algorithms[18] find the natural images according to the features and rules defined by human beings. The retrieval results are generally similar in contour; however, their complete semantic information of the image is missing. From the user's point of view, the same hand-drawn image may represent many different things, due to the semantic "one-to-many" category mapping relationship between the hand-drawn image and the natural image.

2.2.1.2 Review of Sketch based image retrieval techniques (SBIR)

The main problem in SBIR is to localize a region in an image which would be matched with the query image in contour. Rong Zhou et al [19] tackle this problem using human perception to identify two types of regions in an image. First, the main region defined by a weighted centre of image features, suggesting that we could retrieve objects in images regardless of their sizes and positions. Second, region of interests (ROI), is to find the most salient part of an image and is helpful to retrieve images with objects similar to the query in a complicated scene. Using this two-region topology as a feature extractor increases the rate of retrieval and the retrieval speed dramatically. They first extract orientation features and then organize them in a hierarchical way to generate global-to-local features. Based on this characteristic, a hierarchical database index structure could be built which makes it possible to retrieve images on a very large-scale image database online.

Another approach was proposed in [20], they introduced a convolutional neural network based on Siamese network for SBIR, the main idea was to pull output feature vectors closer for input sketch-image pairs that are labeled as similar, and push them away if irrelevant. This is achieved by jointly tuning two convolutional neural networks which linked by one loss function, by learning from the positive and negative sketch-image pairs samples; it results in a small Euclidean distance between a query sketch and a candidate real image if they are similar. Experiments on the Flickr15K dataset shows that this method yields better results than other state of the art methods.

Q.Qi [16] focused on solving the problem of the ambiguity of hand written images (the user drawing's may have many different characteristics) resulting from the one to many relation between the sketch and the matched image, he introduced a personalized SBIR architecture that consists of deep convolutional model as a general model and a personalized model using transfer learning to achieve fine-grained image semantic feature.

This paper proposed a dual-input shared full CNN structure for image visualization feature extraction [21], feature vector, category label supervision information and backpropagation algorithm extracted by the CNN are used to reduce the value of the joint loss function to dynamically adjust the parameter information of each

layer of the network then it used transfer learning to further learn the distribution change from shape contour feature to semantic feature space. Noting that the two most important characteristics of an SBIR algorithm are performance and speed [22] focused on surpassing the state of the art performance on SBIR benchmarks by introducing a hybrid multi-stage training network that exploits both contrastive and triplet networks.

This hybrid network is used to learn a joint sketch-photo embedding suitable for measuring visual similarity in SBIR, for the training it used the TU-Berlin dataset of Eitz et al [23]. And the sketchy dataset of Sangkloy et al [24]. The model was trained using exemplar triplets formed using these query sketches augmented by positive and negative training photos from the internet. The optimal network configuration comprised triplet architecture with branch structure derived from GoogLeNet [5], this technique achieved more than 17% increase in performance accuracy over the published state of the art methods at that time.

An out of the box approach suggested in [25] was to use adversarial training for sketch retrieval, it considered documents that contain unlabelled Merchant Marks, sketch-like symbols that are similar to hieroglyphs.

This approach is particularly interesting because it solves a major problem in machine learning and data science generally, which is the availability of labeled data, because this method doesn't require labeled data, instead it uses the ability of GANs [15] to learn representations that are suitable for image retrieval.

Similar to the approach suggested above, Huang in [26] proposed another adversarial model; the generative model selects well-matched images and passes them to the discriminative model. The discriminative model judges the selected images and sends feedbacks to the generative model. This way they play the minimax game that helps them to train in an adversarial manner.

They conducted a set of experiments on MNIST [1] and CIFAR10 [27] and the results show that the proposed unsupervised adversarial image retrieval framework is effective and robust in image retrieval.

General Disadvantages of using SBIR algorithms

1. Need to localize a region in an image which would be matched with the query image in contour.
2. Needs a database, Database can get large, inefficient.

2.2.2 Internet montage techniques

2.2.2.1 Overview

These techniques are similar in nature to the SBIR techniques discussed above; the major difference is the database we are working with.

In SBIR the database is a dataset of images, sometimes a sketch-image pair and sometimes only images, but in internet montage techniques there is no explicit database, instead the algorithm searches the internet for a candidate image that best matches the required features.

2.2.2.2 Review of Internet montage techniques

An interesting approach using this technique was developed in [16], in which the composed picture is generated by seamlessly stitching several photographs found

Ref	Method	Disadvantage	Comments
[1]	Uses human perception	Impractical	Cannot work with large datasets because it depends on human perception
[20]	Based on Siamese network	Inefficient	Focuses on minimizing the euclidean distance.
[16]	Deep convolutional model and a model using transfer learning	Mislabeling of the label information or the incomplete information has a high probability of affecting the final precision	Needs a highly accurate dataset to avoid mislabeling
[4]	Hybrid multi-stage training network	Complicated architecture.	Each model is trained separately
[20]	Adversarial training using sketch-GAN, thin-GAN\	Difficult to train	Doesn't need labeled data.
[21]	Adversarial training	Difficult to train	Doesn't need labled data

Table 2.1: A comparison of the different Sketch-based image retrieval approaches for sketch to image translation.

online in agreement with the sketch and text labels. For example, you casually draw a beach, a sailboat, a seagull, a bride and a groom, and then you label each one of them. The system will generate a photo for every label then use its blending scheme to generate the one desired photo that contains all the labels.

There were two key contributions to achieve this goal.

- First, they used a novel filtering scheme to select images with simple backgrounds to exclude undesirable found images.
- Second, they used a hybrid blending architecture to provide improved image composition results.

The latter also gives a numerical measurement of composition quality allowing automatic selection of optimal image combinations.

The performance of a CBIR system crucially depends on the feature representation and similarity measurement. For this reason, Ouhda et al [28] introduced a simple but effective framework based on Convolutional Neural Networks (CNN) and Support Vector Machine (SVM) for fast image retrieval composed of feature extraction and classification. Their approach can be summarized as follows:

- Feature extraction using CNNs: the image is transformed into a small dimension feature, which extracts feature that are very relevant. It reduces the size of the problem. Since the extraction of features is only performed once per image, it can be performed quickly on the CPU.
- Classification using SVM which make use of the extracted features.

General Disadvantages of using internet montage techniques
1- Needs internet.
2- Inappropriate results returned by internet.
3- Performance speed trade-off since internet is very large scale.

REF	Method	Disadvantage	Comments
[17]	Filtering scheme + Hybrid blending architecture	1-Need to annotate every sketch you draw. 2-If the filtering scheme fails to filter the image, it may result in a very irrelevant image	Although this is a very interesting approach that has many fantastic results, it sometimes generates very irrelevant images.
[28]	CNNS + SVM	Focuses only on very relevant features.	Even though the performance is very good the quality is fair.

Table 2.2: A comparison of the different internet montage approaches for sketch to image translation.

2.2.3 Generative models techniques

2.2.3.1 Overview

Generative models [29] generally and GANs specifically are undoubtedly one of the most researched ideas in machine learning in recent years, it found major success in sketch to image translation problems.

The fundamental difference between generative models and methods discussed above is that generative models don't need any database, instead it takes the sketch as an input and generates a never seen before image as an output, and this output -if the model is trained correctly- is the desired image that fully represents the input sketch with all of its details.

2.2.3.2 Background of Generative models

2.2.3.2.1 Autoencoders Autoencoders were first introduced in the 1980's [30] with Hebbian learning rules [31] to use the input as a teacher to solve the unsupervised backpropagation problem, and they are unsupervised ANNs that aims to learn how to compress the input data and encode it and then learns to reconstruct the output back from the reduced encoded representation to a representation that is close to the input with the least possible amount of distortion.

The autoencoders reduce the dimensions of the data by learning to ignore the noise. And to achieve this purpose it consists of the following components (Figure 2.2)

• Encoder

This is the first part that the input passes through and it's a fully connected ANN in which the input data compression happens and the data turns into an encoded representation called the latent-space representation.

• Bottleneck

Is the most important layer and it contains the compressed representation of the input data with the lowest possible dimensions, and its importance comes from the fact that this layer gives you a low dimensional representation of high dimensional data in a way that tries to minimize the information loss.

• Decoder

Just like the encoder it has a similar ANN structure (fully connected) and here the model learns to reconstruct the data from the encoded representation with the least distortion possible.

• Reconstruction Loss

It is a function that measures how well the decoder is performing and how close the output is to the original input.

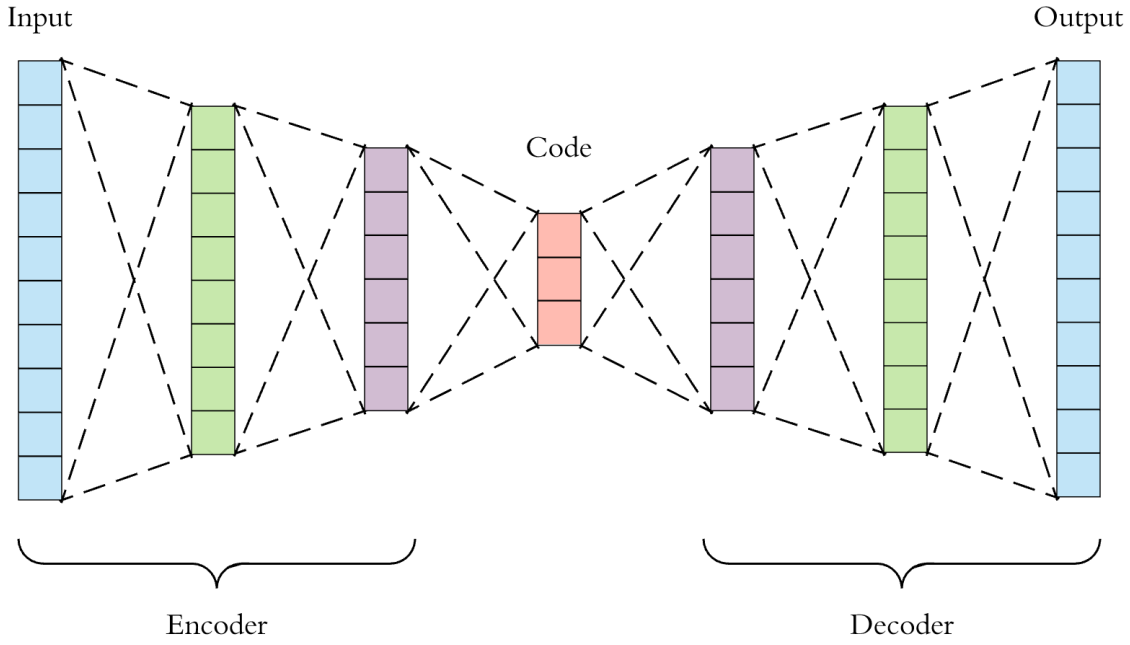


Figure 2.2: illustrates basic structure of an autoencoder

The main problem with this autoencoder architecture is that their latent space which is converted from the input may not be continuous or allow easy interpolation, and when you are building a generative model you want to sample from the latent space randomly or generate variations on an input image from a continuous latent-space [32] and here where the variational autoencoders comes in hand.

2.2.3.2.2 Variational autoencoders (VAEs) They were discovered simultaneously by Kingma and Welling in December 2013 [33] and Rezende et al in January 2014 [34], and they are generative models that falls into the autoencoders group because of its architecture (the final training objective has an encoder and a decoder scheme) but with a different mathematical formulation [35], and its training is regularized to avoid overfitting and ensure that the latent-space has good properties to enable this generative process.

In variational autoencoders the latent space is represented as a normal distribution with a learned mean and standard deviation (Figure 2.3) rather than a single point vector as in the standard autoencoders [36].

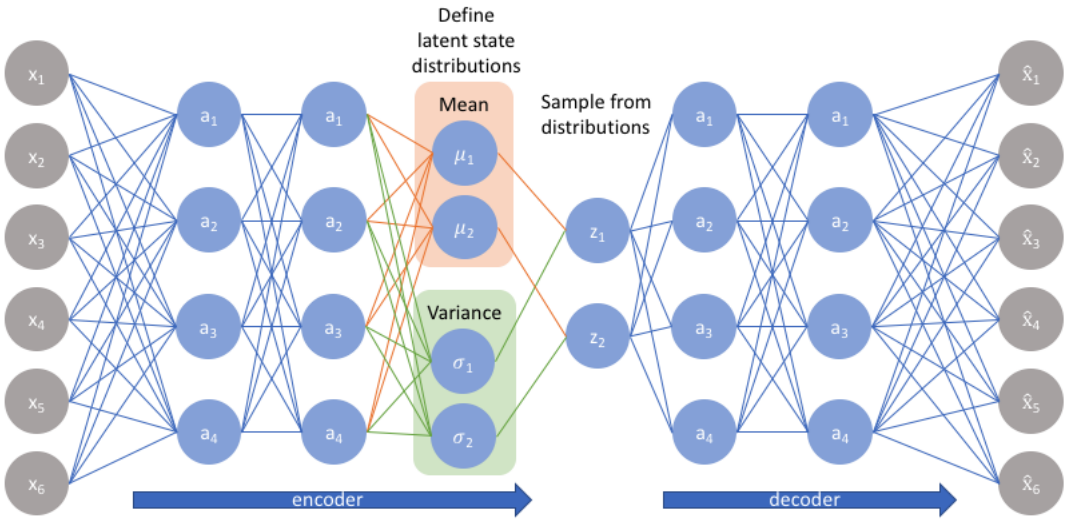


Figure 2.3: Variational auto-encoder architecture diagram

The only fact that variational autoencoders encode input as distributions is not sufficient to ensure continuity and completeness for a proper generation, so a well-defined regularization term is added to prevent the model from encoding the data far apart in the latent space and encourage as much as possible returned distributions to overlap.

$$(f^*, g^*, h^*) = argmax_{(f,g,h)\epsilon FxGxH}(E_{z\sim q_x}(-\frac{||x - f(z)||^2}{2c}) - KL(q_x(z), p(z))) \quad (2.1)$$

where:
$E_{z\sim q_x}(-\frac{||x-f(z)||^2}{2c})$ is the reconstruction loss, and $KL(q_x(z), p(z))$ is the regularization term.

Although the variational autoencoders solved the lack of standard autoencoders ability to generate, but still it has a major problem coming from the fact that it always assumes that the input is distributed as Gaussian distribution (Figure 2.4) and here came the development of the generative adversarial networks.

2.2.3.2.3 Generative Adversarial Networks (GANs) Generative adversarial networks were first introduced by a researcher named Ian Goodfellow in 2014 [15]. They are a class of machine learning techniques that consist of two simultaneously trained models (Figure 2.5): one (the Generator) trained to generate fake data, and the other (the Discriminator) trained to discern the fake data from real examples [36].

GANs use two neural networks pitting against each other to generate new synthetic instances of data that can pass for real data, and those two neural networks are:

- The generator:
  The Generator is a neural network that takes in a vector of random numbers as an input and produces fake examples as an output, its goal is to create examples that are indistinguishable from the real input data from the training

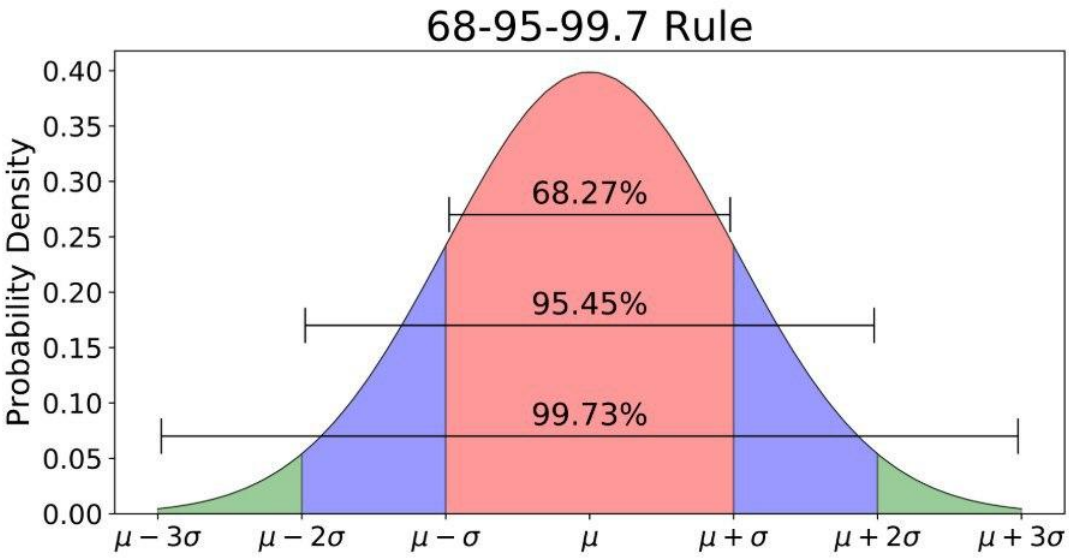


Figure 2.4: Shows a detailed description of the gaussian distribution

set, and it can be - for example - an inverse convolutional network which takes a vector of random noise and upsamples it to an image.

- The discriminator:
  The discriminator is a neural network that takes its input either from the training set labeled as a real example or from the generator's output labeled as a fake example and produces a probability of whether the example is real, so its goal is to distinguish the fake Generator's output from the training dataset real examples, and it's - for example - a standard convolutional network that can categorize the images fed to it, a binomial classifier labeling images as real or fake.

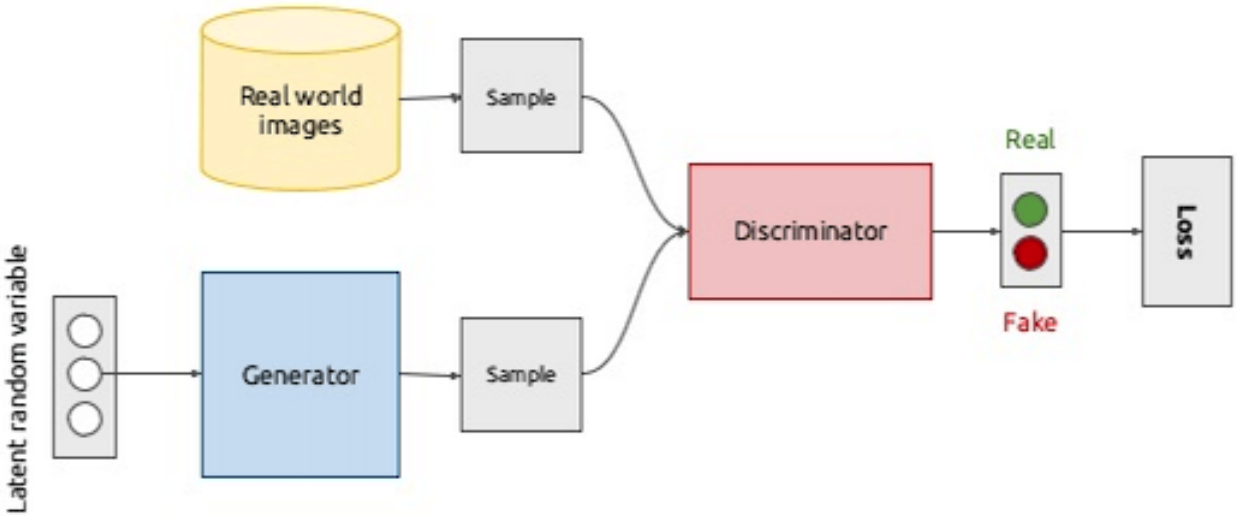


Figure 2.5: Shows a high-level architecture of a GAN network

In GANs both networks are trying to optimize a different and opposing objective function or loss function. We train the discriminator to maximize the probability of assigning the correct label to both training examples and samples from the generator, and simultaneously train the generator to minimize the probability of assigning the correct labels to fake samples as shown in equation 2.2

$$min_G max_D V(D,G) = E_{x\sim P_{data(x)}}[\log D(x)] + E_{z\sim P_z(z)}[\log(1 - D(G(z)))] \quad (2.2)$$

2.2.3.3 Review of Generative models techniques

The single most popular paper on Image to Image translation using conditional generative adversarial networks was introduced by Isola et al [37]. Their work was so popular because of 2 main reasons.

First, they introduced a general purpose solution to image-image translation problem using DCGANs, which means that these networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping, which makes it possible to apply a generic solution to problems that may require different loss functions.

Second, a software called pix2pix [38] was developed associated with this paper which lets users to experience the system to further show the ease of adaptation of the system without the need of parameters tweaking.

Then [39] focused on face-generation from sketches acknowledging that it is difficult to generate a satisfactory face images because they need more texture details and color information. But solving this problem as a face hallucination super-resolution reconstruction they propose a more suitable network. The architecture of the generator was composed of a feature extracting network and downsampling–upsampling network (which is known as Unet), both networks use skip-connections to reduce the number of layers. The discriminator network works as expected as it is designed to decide whether the generated face is real or not. This proposed method achieved 16.3069 PSNR and 0.579 SSIM comparing to the above discussed paper (pix2pix) 14.5834 PSNR and 0.5682 SSIM which indicates better results even comparing to one of the most popular image-image translation papers (pix2pix).

The dataset required to train a GAN to solve a sketch-image problem is usually a sketch-image pair, which is difficult to obtain because we need artists to draw a sketch of every single image in the dataset. Considering this problem, [40] proposed an unsupervised algorithm to sketch-image problem claiming that the key to solving this problem is dividing it into two sub problems, shape translation and colorization. That's why they proposed two models, one solving each task.

- Shape translation:
  The key to this step is removing the factor of color and forcing the network to focus on shape producing a grey scale image. The proposed shape translation network is developed based on CycleGAN [41].

- Colorization:
  from here on the problem is reduced to only colorizing the grey scale image generated from the translation step. The network adopts an encoder-decoder structure but modified to add the adversarial loss.

This approach was evaluated using the FID score [42], scoring 50.56 compared to the 65.09 scored by the pix2pix method discussed above.

Then [43] focused on making the generation process interactive, as the user starts to draw a sketch, the network automatically auto-completes the shape with a plausible image this allows a feedback loop, where the user can edit their sketch based on the networks recommendations. Another point of strength in this paper is that it also enables generation of multiple classes with a single generator network without feature mixing using a gating-based approach for class conditioning. This gating network was designed using Resnet blocks with 1D convolution.

General Disadvantages of using GANs

1. They are notoriously difficult to train.
2. Training takes significant amount of time.
3. Complex architecture.

REF	METHOD	Disadvantage	Comments
[37]	Conditional Generative Adversarial networks	Limited number of classes	Added feature of pix2pix software which allows users to experiment the system.
[39]	Standard discriminato Feature extraction +skip connections + Unet generator	Although the performance is very good, it only focuses on face generation	Comparing to pix2pix , it has better performance but much less number of classes
[40]	Shape translation model + colorization model	1/ the generative model fails to translate some sketches 2/ uses a single-class setting	Uses unsupervised training.
[43]	gating-based approach for class conditioning using Resnets.	Focuses on shape completion rather than the quality of the generated shape	Interactive generation process

Table 2.3: A comparison of the different generative techniques for sketch to image translation

Chapter 3

Methodology

3.1 Introduction

This chapter will discuss:

- The implementation environment, associated tools and libraries that were used for image processing, training the deep learning models, source code editing and packages installation.
- thorough explanation and discussion of the model used (pix2pix[37] model) architecture, layers, activation functions, overfitting avoidance techniques and hyper-parameters tuning.
- Implementation of three different approaches using the pix2pix GAN model which consists of 2 CNN models (the generator and the discriminator).
- Discuss each approach's dataset, data acquisition and data preprocessing techniques in details.
- Discuss each approach's collection, labeling, preprocessing and details of the dataset used for training and evaluation.
- Discuss the drawbacks and advantages of each one of the three approaches to be covered.
- Implementation of the open-pose open source pre-trained model that is used for object detection and pose estimation and discuss it's justification, data representation and the hyper-parameters that were used.

3.2 Implementation Environment

3.2.1 Tools and Software Libraries

Python is an object oriented, high level programming language that is very powerful in deep learning research field, the large user base of python provides good support for researchers and developers, so it is mainly adopted throughout this research.

PIP is a package management tool used to install and manage software packages written in python. Many deep learning packages can be found in the default source for packages and also their dependencies. Pip's advantages include ease of its command line interface, which makes installing Python software packages as easy as issuing one command also it can manage full lists of packages with just their corresponding version numbers.

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. It includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real time vision applications and takes advantage of SSE instructions when available.

Atom is a free and open source text and source code editor for mac OS, Linux and Windows with support for plug-ins written in Node.js, and embedded Git Control, it was developed by GitHub.

Google colab is a free online cloud service that supports free GPU which enables you to write, compile and run python to develop deep learning applications using popular libraries such as Keras, Tensorflow, Pytorch, and OpenCV.

Tensorflow is an open source software library for high performance numerical computation. It's flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs).

TensorBoard is a tool for providing the measurements and visualizations needed during the machine learning workflow. It enables tracking experiment metrics like loss and accuracy, visualizing the model graph, projecting embeddings to a lower dimensional space, and much more.

Keras is a high-level neural networks API, written in python and capable of running on top of Tensorflow. It was developed with a focus on enabling fast experimentation. Some of the advantages using keras:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

3.2.2 Processing Environment

Training and testing the models was done in one local machine and one online cloud service (google colab). Some of the experiments were done on an MSI gaming laptop incorporated with Intel Core i7 processor, 16GB RAM, Nvidia GeForce RTX 2060, VRam 6GB, 2.6 Ghz that goes up to 5Ghz with gaming boost, 64 based processor system and windows 10 Pro Operating. The other part of the experiments were performed on google colab which offers many GPUs variants including K80,P4, T4, P100 with K80 being the slowest and P100 being the fastest, but the only downside is that you don't have control over which GPU you get, the distribution occurs depending on how much you are using the GPU.

3.3 Pix2pix model discussion and explanation

Pix2pix model which was presented in [37] (although "pix2pix" is not an official name and, it is used by the AI community, we will be using it throughout this chapter) is used to convert low level representation of an image to a high level representation of that same image, it's network is basically a special type of generative adversarial network (GAN) that is called a Conditional GANs (CGANs), they have some conditional settings and they learn the image-to-image mapping under this condition. Whereas, basic GAN's generate images from a random distribution vector with no condition applied.

3.3.1 Network architecture

Pix2pix model is a GAN model, so it consists of a generator and a discriminator and their networks are as follows:

3.3.1.1 The generator's architecture

The generator network is using a U-Net based architecture (see figure 3.1) that performs better than the basic autoencoder because a U-Net architecture allows low-level information to shortcut across the network throw the skip connections. This network can be further sliced into two networks:

- The Encoder

The encoder network is a down-sampler and has seven convolutional blocks, and each convolutional block has a convolutional layer, followed a LeakyRelu activation function and it also has a batch normalization layer except the first convolutional layer.

- The Decoder

The decoder network is an up-sampler and it has seven up-sampling convolutional blocks Each up-sampling convolutional block has an up-sampling layer, followed by a convolutional layer, a batch normalization layer and a ReLU activation function.

3.3.1.2 The discriminator's architecture

Discriminator network uses of PatchGAN architecture, the PatchGAN network contains five convolutional blocks, and a 30*30 output layer that serves as the decision upon which the network judges the generator's output as fake or real (see figure 3.2).

3.3.2 Activation function

The model uses the Rectified Linear Unit (RELU) activation function on the generator decoder, uses the leaky Recited Linear Unit activation function on the discriminator and the generator encoder, and uses the sigmoid activation function on the discriminator's output patch layer and these functions are defined as follows:

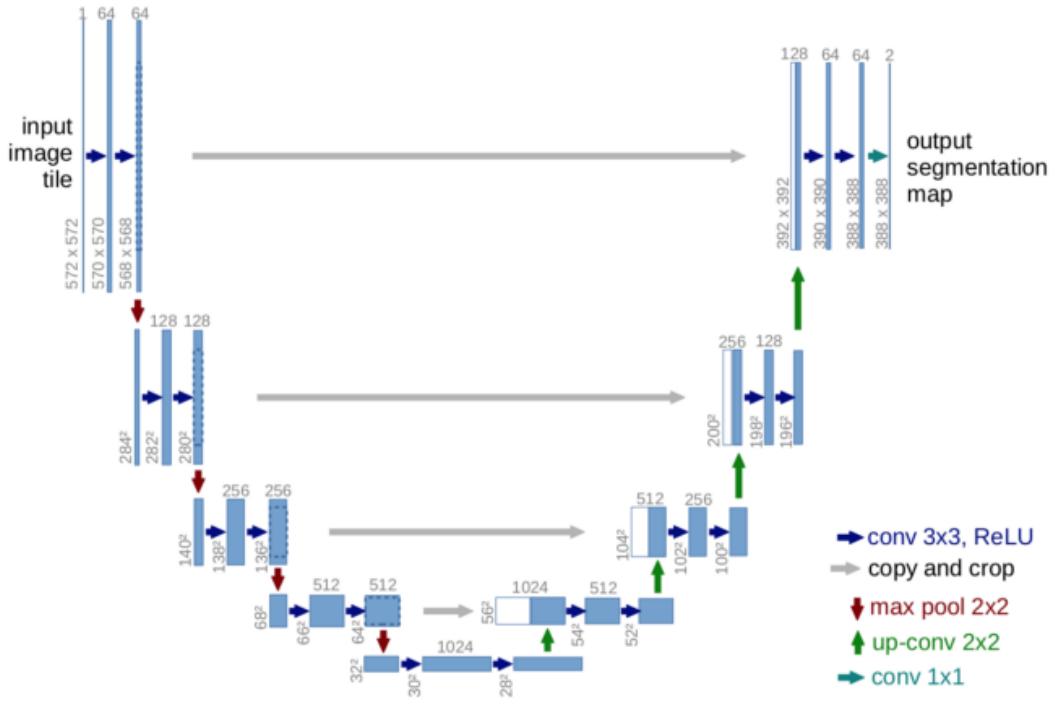


Figure 3.1: basic U-Net architecture

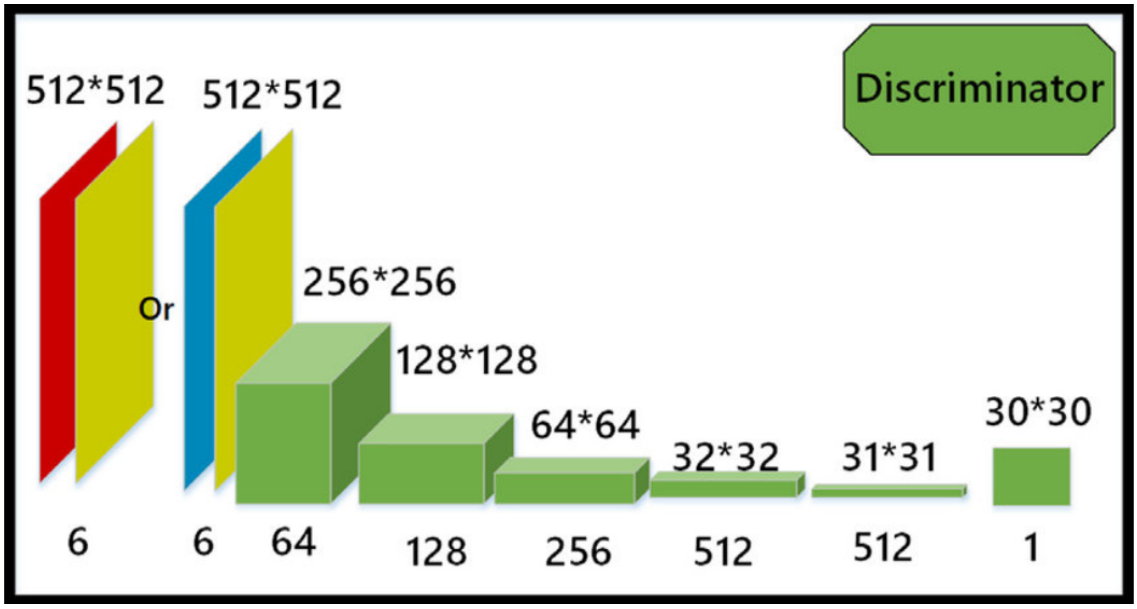


Figure 3.2: discriminator network architecture which is a PatchGan

- **Rectified linear**

$$f(x) = max(0, x) \tag{3.1}$$

- **Leaky rectified linear**

$$f(x) = x \quad if(x \geq 0), f(x) = \alpha * x \quad if(x < 0) \tag{3.2}$$

- **Sigmoid**

$$f(x) = \frac{1}{(1 + e^{-}x)} \tag{3.3}$$

3.3.3 Loss function

Since we have two networks (the generator and the discriminator) we will have a loss function for each network to act as a metric by which the network knows how much mistaken its output was in order to improve in next iterations.

3.3.3.1 Discriminator loss

This loss acts as a judge to how much the generated image was real or fake.

$$J^{(D)} = -\frac{1}{2}E_{x\sim p_{data}} \log D(x) - \frac{1}{2}E_z(1 - D(G(z))) \tag{3.4}$$

Since somethings are better explained in a programming way, this loss can be described using pseudo code as:
> Loss_func = Cross_Entropy_Loss()
> Disc_loss=loss_func(real_img,array_of_ones)+loss_func(fake_img,array_of$_zeros$)
Where the cross-entropy loss in equation 3.5

$$Binary\,Cross\,Entropy\,Loss = -(p(x) \log y + (1 - p(x)) \log (1 - y) \tag{3.5}$$

3.3.3.2 Generator loss

It has two losses

- The GAN loss $J^{(G)} = -J^{(D)}$ which is exactly the opposite of the discriminator loss since they are competing to fool/discover each other.

- L1 loss $L1\,loss\,function = \sum_{i=1}^{n} |y_{true} - p(x)|$ between the output of the generator and the original image (which acts as a labels), this L1 loss helps the generator output to be structurally similar to the target image (label).

in pseudo code:
> Loss_func = Cross_Entropy_Loss()
> gen_loss = loss_func(fake_img,array_of_ones) + L1_loss(fake_img,real_img)

3.3.4 Optimization algorithm

To optimize our networks, we follow the standard approach from [15]: we alternate between one gradient descent step on D (the discriminator), and one step on G (the generator). As suggested in the original GAN paper [15], rather than training G to minimize $\log(1 - D(x, G(xw, z))$, we instead train to maximize $\log D(x, G(x, z)$ [15]. In addition, we divide the objective by 2 while optimizing D, which slows down the rate at which D learns relative to G. We use mini-batch SGD and apply the Adam solver [44], with a learning rate of 0.0002, and momentum parameters $\beta 1 = 0.5$, $\beta 2 = 0.999$.

3.3.5 explanation

After the dataset is acquired, preprocessed and made completely ready for training (dataset acquisition and preprocessing will be discussed in detail in next section),we will discuss how all these points fit together.

- **Step1:** First we feed the generator network with a batch of the low-level image, maybe a facade drawing, a sketch, or any other low representation of an image.
- **Step2:** The generator outputs a batch_size*256*256*3 (RGB) images that should be -ideally- a high-level representation of the input image.
- **Step3:** Then the discriminator network takes the output of step2 and concatenates it with the actual image (label) and produces a decision on how much the output of step2 is real as a 30*30 output, each one of these outputs judges a 70*70 portion of a single image (more on this in later sections).
- **Step4:** We continue training the discriminator with a real image this time, so it takes a a batch of real low-level images and it's corresponding high-level batch images (labels) and produces a decision on how much the input images is real, again as a 30*30 output for every single image.
- **Step5:** We calculate the discriminator and generator losses as discussed in subsections 3.3.3.1 , 3.3.3.2 respectively.
- **Step6:** Then we calculate the gradient of these losses with respect to the network parameters and propagate them back throw the network using Adam optimizer.
- **Step7:** Repeat steps1-6 using the specified number of epochs until we get the desired output, while keeping in mind to monitor the performance because parameter tuning or over-fitting prevention techniques may be needed.

We use this knowledge and apply it to 3 different approaches that will be discussed in detail in the next three sections.

3.4 First approach

This approach is a straightforward implementation of the pix2pix model that produces very impressive results.

(a) handbags dataset

(b) shoes dataset

Figure 3.3: Example of edge-image dataset

3.4.1 Dataset

With the popularity of machine learning and deep learning in particular the amount of available datasets significantly increased. From one side more people are interested in using and analyzing different sources of data. From the other side the modern technology allows to collect and store huge amounts of data. The problems start to rise, when a labeled dataset for a specific task is required.In particular a famous datatset from Berkeley university is used for sketch-image translation tasks, it consists of edge-image pair (see figure 3.3)

3.4.2 Data acquisition

Pictures of shoes and bags were taken, and then manually drawn. Since this approach is merely a test for how good the model and our hyper-parameter tuning techniques are, we used the dataset we referred to in the previous sub-section as is.

3.4.3 Data prepossessing

1. We apply random jittering and mirroring to the training dataset as mentioned in [37].

random jittering steps:

- The image is resized to 286 x 286

- Then randomly cropped to 256 x 256
- Finally the image is randomly flipped horizontally (left to right)

2. Then we normalize the data in the (-1 - 1) range.

3.4.4 Implementational details

This approach was trained and tested using Google Colab's p100 GPU, 12GB RAM, 30GB disk storage. Since we are using a low-specification machine we needed to do some optimizations on the implementation steps discussed in 3.3.5.

- First we choose the batch size to be 1 since this will take the minimum amount of RAM, but will risk having unstable training depending on the noise present in the dataset (a trade-off that we had to make because of the lack of training resources).
- Second we used tensorflow's input pipeline which optimizes the use of GPU, RAM and disk space.

3.4.5 Disadvantages and limitations

Although this approach achieves very realistic results, it's dataset requires an extremely difficult procedure in which we take pictures of the required object and then manually draw every single one of them. Obviously this is a time consuming tasks since the dataset may contain more than 10,000 sample, and this is exactly why we applied the second approach.

3.5 Second Approach

This approach is an attempt to generalize the previous approach and to overcome it's disadvantages.

3.5.1 Dataset

Here we used the CelebA Dataset[45] available on Kaggle website, it is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations.(see figure 3.4)

3.5.2 Data acquisition/prepossessing

As discussed earlier, our model needs a low-level, high-level representation of an image pair dataset in-order to learn the mapping between them. To acquire this pair:

- We first load CelebA dataset from Kaggle
- Then we use the Canny edge detector filter to get the edges of CelebA dataset.

Figure 3.4: CelebA dataset example

(a) single image from CelebA

(b) after applying Canny edge detector

Figure 3.5: image to edge using Canny filter

This way we get the desired pair as shown in figure 3.5.

After the dataset pair is ready we preprocess by :

- Random jittering (see figure 3.6)
- Normalizing data to [-1 - 1] range.

Figure 3.6: Doing random jittering 4 times

3.5.3 Implementational details

Here we also use the input pipeline and a batch size of 1, but this time the pipeline only contains the high-level image and not the edge, because the edge will later be processed using Canny filter which is provided by OpenCV library. So the logic goes as follows:

- Define the input pipeline and fill it with CelebA dataset.
- Do the necessary preprocessing.
- In the training loop, when the input pipeline outputs an image, use Canny filter and train with the image-edge pair.

3.5.4 Disadvantages and limitations

This approach solves the problem of having to draw all the dataset manually since it uses image processing techniques to create an estimation of an image edge. But it produces relatively less realistic results.

Figure 3.7: CoCo Dataset examples

3.6 Third Approach

This approach is similar to the previous one in the sense that it uses only a high-level image then use some techniques to create the low-level pair. Except that this approach focuses on pose generation.

3.6.1 Dataset

This time we need a dataset that consists of people with different poses so we use some of the CoCo dataset [46] which is an excellent object detection dataset with 80 classes, 80,000 training images and 40,000 validation images. In particular we use 20,000 images for training, validation, and testing(see figure 3.7).

3.6.2 Data acquisition/prepossessing

We choose 20,000 sample from the datatset then we cropped it in order to ensure that every sample has only one person in it, then we pass these samples to OpenPose model [47] which is a pre-trained widely used pose detection model. At this point we have both the image and it's pose which will act as our training pair(see figure 4.11).
For the preprocessing we also do both of the random jittering and the normalization steps.

3.6.3 Implementational details

Implementation logic goes as follows:

- First we put the downloaded dataset on a google drive folder.
- Then we apply pose detection to all images on that folder using OpenPose model, this creates another folder that contains the model's output.
- We use the input pipeline to store the image-pose pair, we also choose the batch size to be 1.
- Then we do all necessary preprocessing to the image-pose pair.
- Finally we start the training using this data.

3.7 Disadvantages and limitations

In this approach we also do not need to manually sketch all of the dataset, instead the OpenPose model takes care of creating the low-level representation of the dataset. But this approach can only work on humans since we can't get the pose of anything else.

3.8 Comparison

See table 3.1 for full comparison between the three approaches discussed.

	Advantages	**Disadvantages**	**Comments**
Approach1	The simplest, gives best results.	Very difficult to optain the dataset	Direct implementation of pix2pix model
Approach2	Easy to optain the dataset since it uses image	Produces less impressive results.	Uses Canny filter which is implemented in OpenCv library
Approach3	Easy to obtain dataset since it uses Pose estimation techniques	Works only on humans because we can't estimate the pose of other objects	Uses OpenPose pre-trained model for pose estimation.

Table 3.1: Comparison between different approaches

Chapter 4

Results and discussion

In this chapter we will point out a number of experiments that were carried out during the development of the three approaches, also we present the results of each experiment and discuss its results.

4.1 Overview

Generative adversarial networks are notoriously difficult to train since they consist of two networks competing against each other and they both need to be optimized.
In normal neural networks, low training error combined with low validation error always means good results, but in GANs even if one network gives very low training and validation loss that doesn't necessarily mean good results, it could simply mean one network is dominating the other. And since our goal is to have a good generator network, this means -ideally- the discriminator should not be able to classify which image is real and which is fake.
In the next four sections we will talk about our evaluation methods and then discuss in detail how our experiments resulted in achieving the goal discussed above while applying our evaluation methods to backup our claim, we will talk about each approach's experiments and results separately in sections 4.3, 4.4, 4.5 .

4.2 Evaluation metrics

Using the right evaluation metric for any system is crucial. This metric influences how the performance of different deep learning networks is measured and compared, it also influence how much is the importance of the different characteristics in the results. GANs evaluation is particularly difficult because how can we judge to what extent a generated image is good or not. Generally both subjective and objective evaluation methods are used.

- **Subjective measures:**
  These measures rely on people judgment, we bring many people to view our results and then numerically give it a score depending on a pre-chosen criteria.

- **Objective measures:**
  In GANs objective measures are a large research area since it is not clear exactly how a number can determine the realism of a generated image, but some metrics were developed, here we will use bot the **Inception score(IS)** and the **Fréchet Inception Distance score (FID)**.

4.2.1 Inception Score

The Inception Score, or IS for short, is an objective metric for evaluating the quality of generated images, specifically synthetic images output by generative adversarial network models.
The inception score involves using a pre-trained deep learning neural network model for image classification to classify the generated images. Specifically, the Inception v3 model described in [48]. A large number of generated images are classified using the model. Specifically, the probability of the image belonging to each class is predicted. These predictions are then summarized into the inception score.
The score seeks to capture two properties of a collection of generated images:

- **Image Quality**: Do images look like a specific object?
- **Image Diversity**: Is a wide range of objects generated?

4.2.2 Fréchet Inception Distance score

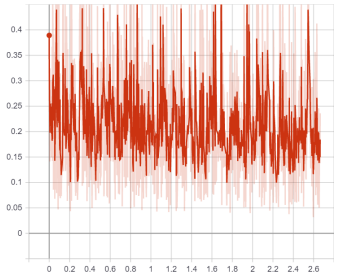
The Fréchet Inception Distance, or FID for short, is a metric for evaluating the quality of generated images and it was proposed in [42], also to evaluate the performance of generative adversarial networks. The score was proposed as an improvement over the existing Inception Score.
The inception score does not capture how synthetic images compare to real images. The goal in developing the FID score was to evaluate synthetic images based on the statistics of a collection of synthetic images compared to the statistics of a collection of real images from the target domain.

4.3 Sketch-image(first approach) experiments

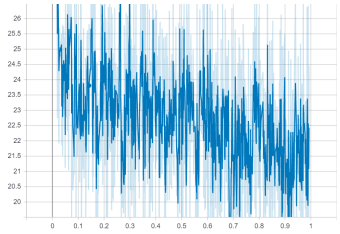
We targeted the effect of tuning different hyper-parameters in our model and it's effect on our evaluation metrics.We performed 3 experiments:

- **Experiment 1:** Adam optimizer with batch size of 1
- **Experiment 2:** Adam optimizer with batch size of 32
- **Experiment 3:** SGD optimizer with batch size of 1

In the next two subsections we will discuss the results that lead us to choose the optimizer and batch size.



(a) Generator loss using Adam optimizer



(b) Generator loss using SGD optimizer

Figure 4.1: Generator loss using Adam and SGD optimizers

(a) Adam optimizer sample result

(b) SGD optimizer sample result

Figure 4.2: Sample results for both SGD and Adam optimizers

4.3.1 Optimizer Selection

An optimization algorithm is used to minimize the loss function of both the generator and the discriminator networks and to update the network learnable parameters (i.e the weights and biases) to the direction of the optimal solution. The choice of optimization algorithms for a deep learning model can play a big role in producing optimum and faster results.
In this experiment we tried to find the optimum optimization algorithm for our training process. We trained the networks using two different optimizers: Adam and SGD.
We notice that Adam optimizer produces more realistic (see figure 4.2) results but both optimizers suffer from non-stable training (see figure 4.1)

4.3.2 Batch Size Selection

Batch size hyper-parameter represents the number of training samples that going to be propagated through the network in order to make one update to the network parameters. It controls how many predictions must be made at a time. The batch size impacts the CNN training both in terms of the time to converge and the amount of overfitting i.e., smaller batch size yields faster computation but requires visiting more examples in order to reach the same error, since there are less updates per training iteration.
In this experiment we tried to find the optimum batch size for our training process. We used two values 1 image per batch and 32 image per batch, in the experiment we are using Adam optimizer since the first experiment revealed it has better performance.
Here we find that even-though a batch size of 1 is cheaper and more memory efficient, it produces significantly better results (see figure 4.3)

(a) Batch size = 1 sample result

(b) Batch size = 32 sample result

Figure 4.3: Sample results for both Batch size of 1 and 32

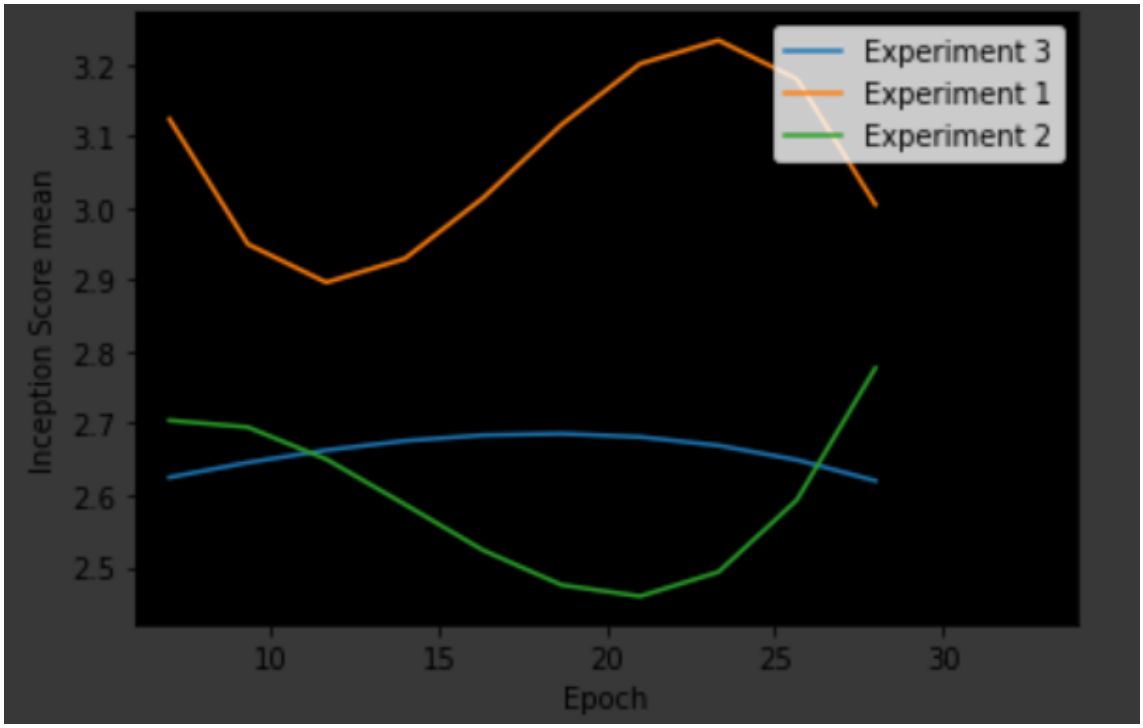


Figure 4.4: Inception score mean for the three experiments

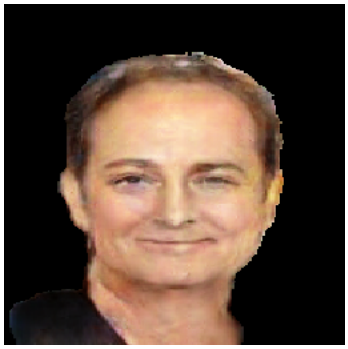
4.3.3 Final result selection

So far, Adam optimizer with batch size of 1 (experiment 1) proved to out perform other configurations -subjectively-. Next we calculate the inception score for all 3 experiments.
We notice that experiment 1 indeed outperforms the rest scoring noticeably higher Inception score (see figure 4.4)

4.4 Edge-image(second approach) experiments

Learning from first approach's experiments we automatically choose Adam optimizer and a batch size of 1 and then proceed with our experiments.
Since we are working with more complex dataset we increased the dataset size from 2k to 7k (eventhough the first two experiments where done in 2k dataset, subsequent discussion will prove that using less dataset will lead to less accurate results).
We perform 5 experiments:

- **Experiment 1:**
  Dataset size 2k, no background removal, no random noise, GAN loss + L1



(a) Using background removal sample image.

(b) Not using background removal sample image.

Figure 4.5: Sample results for with/without background removal preprocessing.

loss.

- **Experiment 2:**
  Dataset size 2k, **background removal**, no random noise, GAN loss + L1 loss.

- **Experiment 3:**
  Dataset size 7k, background removal, no random noise, GAN loss + L1 loss.

- **Experiment 4:**
  Dataset size 7k, background removal, no random noise, **GAN loss + L2 loss**.

- **Experiment 5:**
  Dataset size 7k, background removal, **random noise**, GAN loss + L2 loss.

We will perform these experiments one by one and always pick the best configuration and move on.

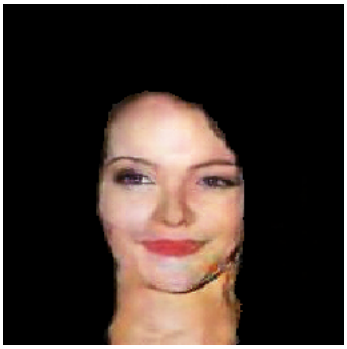
4.4.1 Background Removal/No Background Removal Selection

Backgrounds are considered unwanted random noise since they may be different in every single sample in our dataset, and our goal is to output a face regardless of the background, so the logical conclusion would be removing the background would result in better model output. In this experiment (experiment 1 VS experiment 2), our goal was to prove the importance of removing the background of the dataset. So we trained our model on a dataset with background, then we trained the same model on the same dataset but with removed background(see figure 4.5).
Results clearly show that removing background lead to cleaner more accurate output.

4.4.2 Losses function Selection

Loss functions represent the error surface in which the optimization algorithm operates on to optimize the hyper-parameters, so it must be chosen carefully. Here (experiment 3 VS experiment 4) we try two different loss functions :

(a) Using L1 loss sample image.



(b) Using L2 loss sample image.

Figure 4.6: Sample results for L1/L2 loss functions.

- GAN Loss + L1 loss.
- GAN loss + l2 loss.

The two losses -L1 ,L2 - are similar in the sense that they are both used to keep the structure of the output image similar to the input image, the only difference between them is that the L2 loss penalizes the model more when it outputs a wrong output (since it has a square term).
Generally, L2 Loss Function is preferred in most of the cases. But when the outliers are present in the dataset, then the L2 Loss Function does not perform well. The reason behind this bad performance is that if the dataset is having outliers, then because of the consideration of the squared differences, it leads to the much larger error. Hence, L2 Loss Function is not useful here. Prefer L1 Loss Function as it is not affected by the outliers or remove the outliers and then use L2 Loss Function. Figure 4.6 and 4.7 shows two sample images and the inception score respectively.

Results show better inception score when using L2 loss and subjectively better looking images.

4.4.3 Random noise

Now we investigate the effect of adding a random noise layer to the model since some scholars belief that it is a good way to increase the generalization of the model(experiment 4 VS experiment 5).
The random noise is normally distributed across the dataset, and it helps the model not to memorize patterns in the dataset thus reduces the possibility of overfitting. Figure 4.8 and 4.9 shows two sample images and the inception score respectively.

Looking at figure 4.8 we see that using random noise gives cleaner and more realistic looking output as expected, but looking at 4.9 we see that the inception score of experiment three and five is nearly similar, which is not surprising since the inception score only tries to give an evaluation of how realistic the image is and it may not be very accurate. Figure 4.10 shows the generator loss in experiment 2, 3, 4, 5.
Since we are dealing with GANs and adversarial training, we should mention that the generator loss is not an accurate estimator of how good the network is, since it may be very low because the discriminator network is bad and not because the

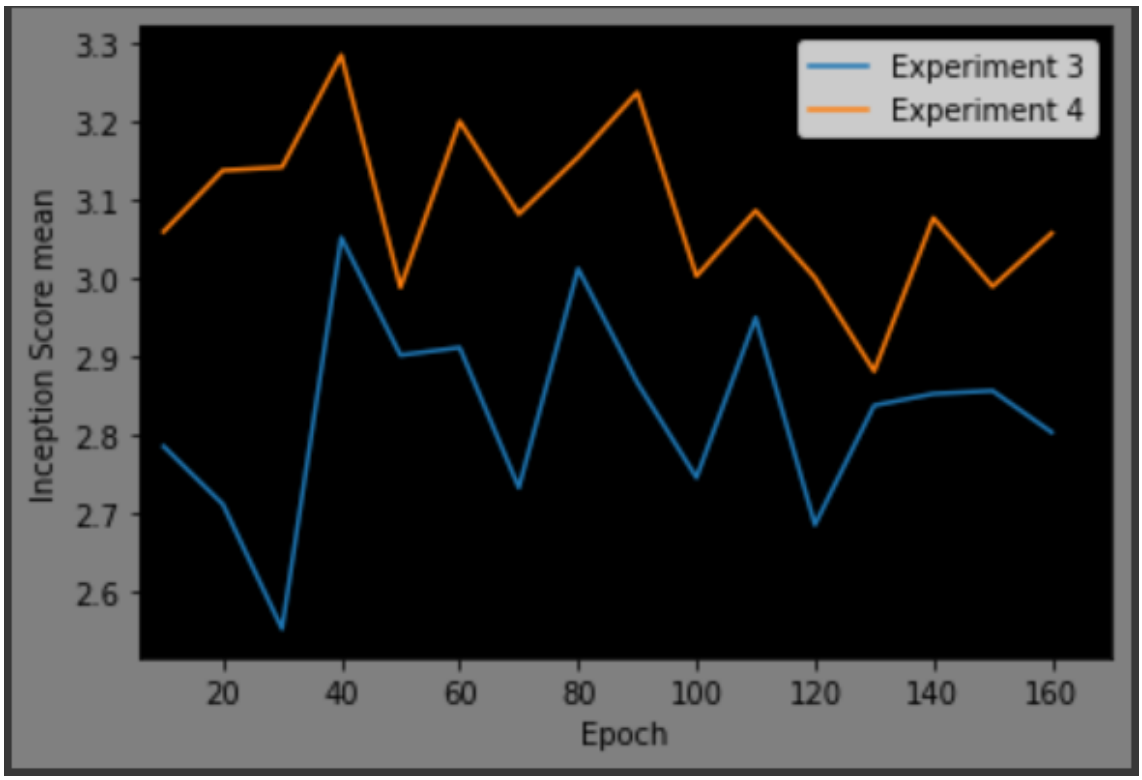
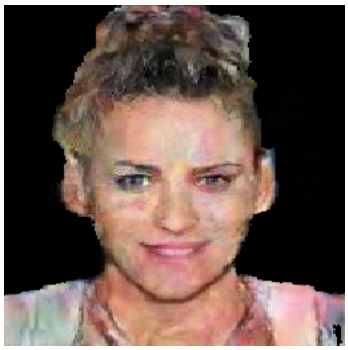


Figure 4.7: Inception score mean for experiment 3 and experiment 4.



(a) Without Random Noise sample image.

(b) Using Random Noise sample image.

Figure 4.8: Sample results With/without Random noise.

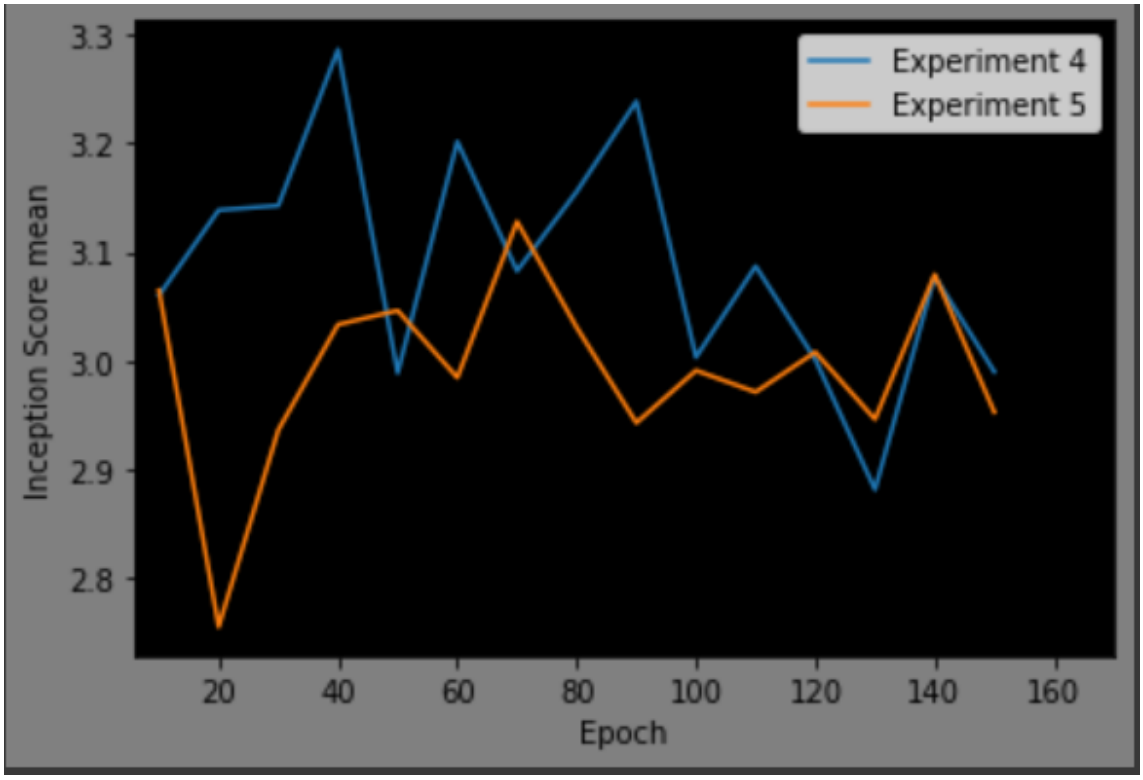


Figure 4.9: Inception score mean for experiment 4 and experiment 5.

generator network is good, but then again not even the inception score is an accurate metric since it tries to convert subjective measures to objective ones. So we use all these metricise with our subjective sense to get an indicator of the quality of the generator network.

We notice that experiment 5 yields best results in the inception score, generator loss and even in even subjectively it produce cleaner more crisp output.

4.5 Pose-image(third approach) experiments

The above discussed approaches demonstrate how powerfull the pix2pix model is, so we go one step further by attempting to generate realistic looking people knowing only the pose they are in.
The training pair consists of image-pose, eventhough we completed the necessary preprocessing code (using OpenPose model) the data acquisition phase was not complete since there is no available data for such tasks and it must be collected manually (will be addressed more thoroughly in the next chapter).
Figure 4.11 shows the result of the OpenPose model which is responsible for preprocessing.

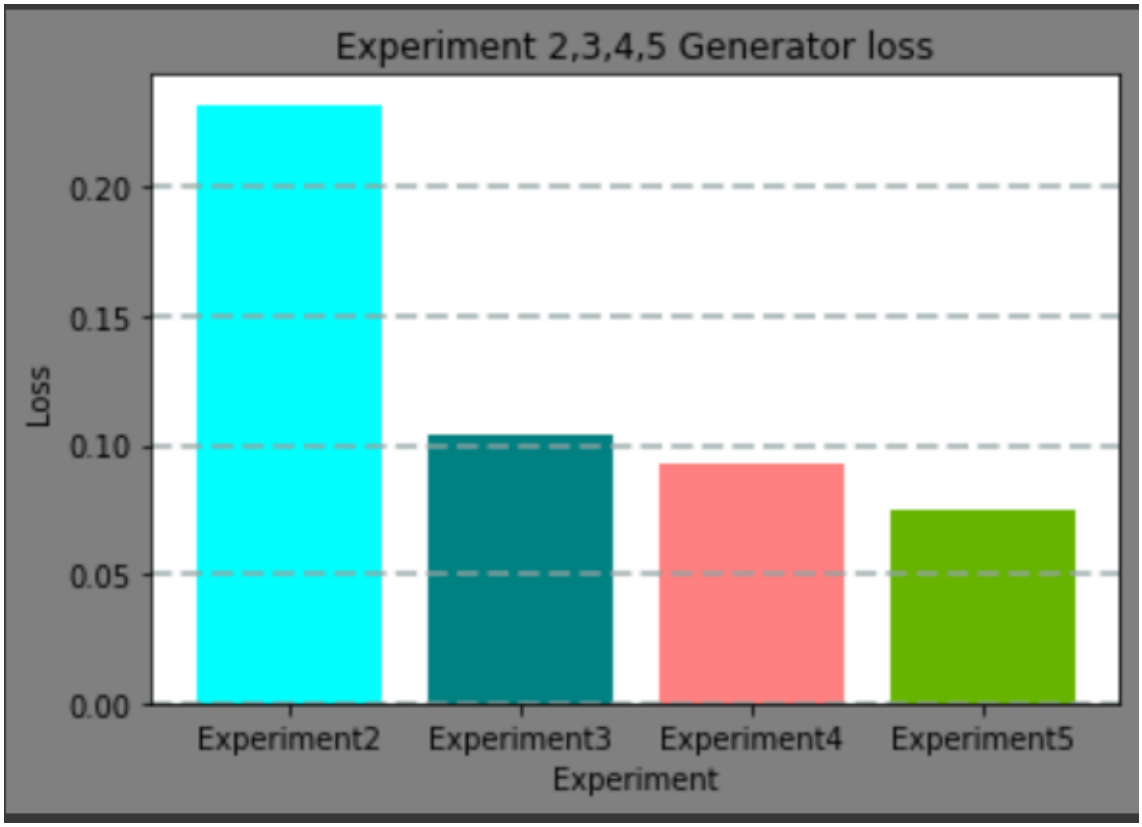
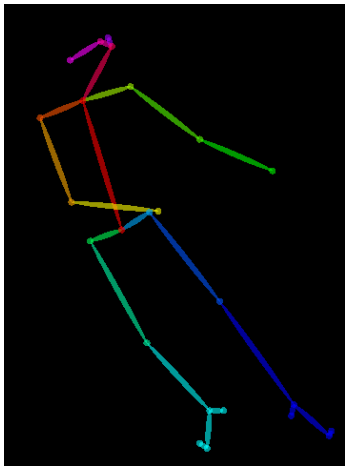


Figure 4.10: Generator loss for experiments 2,3,4,5

(a) single image form CoCo datatset



(b) after using OpenPose for pose detection

Figure 4.11: image to pose using OpenPose model

Chapter 5

Conclusion

5.1 Summary of Work and Findings

The aim of our work was to use the pix2pix model to translate low level sketches to high level images, and develop ways to overcome some of the model limitations. To achieve those goals, we worked with three approaches.

In the first approach we implemented the model to recreate the results with the computational power we have, and it produced very impressive result which shows the efficiency of the model.

Then in the second approach we overcame the generalization limitation of the previous approach by applying the canny edge detector to get sketches from the high-level images instead of manually drawing each one of them, and then modify the model by adding extra input layer to add random noise.To get to the result, we ran five different experiments in the CelebA Dataset[45].Pix2pix model modified with extra input layer, L2 loss, and 7k dataset where the configurations that we ended up using since it led to the best results. The only drawback of this approach is that it produces relatively less realistic results than the first approach because the dataset is more complex.

In the third approach we tried to go even further by letting the model generate high-level images from only poses, we achieved this by letting the OpenPose model take care of creating the low-level representation of the dataset for the training. But this approach only work with humans, and does not produce good results since there is no dedicated dataset for this task and we need to collect it manually.

5.2 Future Work

From the results we got from the three approaches we recommend that:

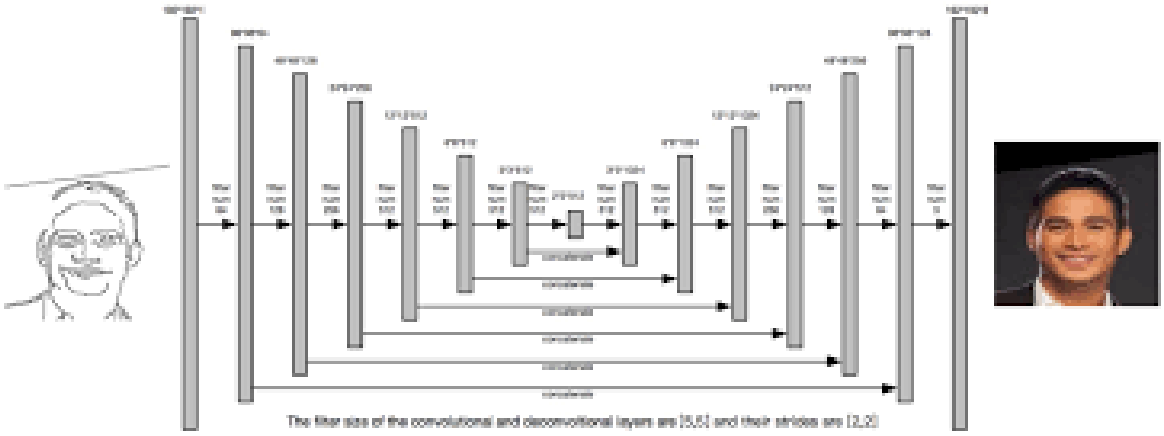
- Using Pix2pixHD model instead of Pix2pix model to get more realistic results .
- Finding a suitable dataset to train the model on poses (third approach).
- Generalize from image-to-image tasks by applying the third approach to solve Video-to-Video tasks.

- Preprocess the training dataset images with more powerful and better performing pre-trained models.

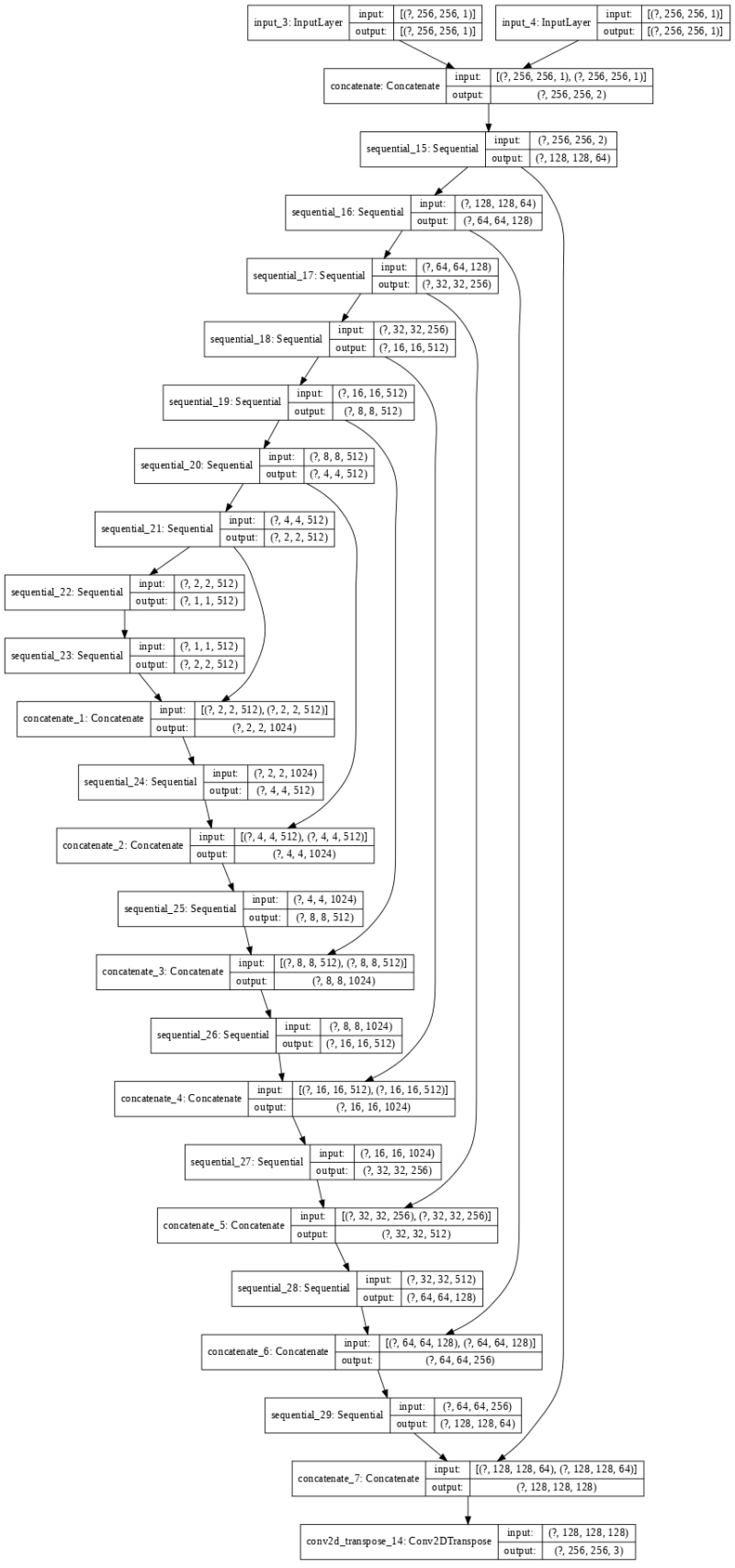
Appendix A

Appendix A : architecture listings

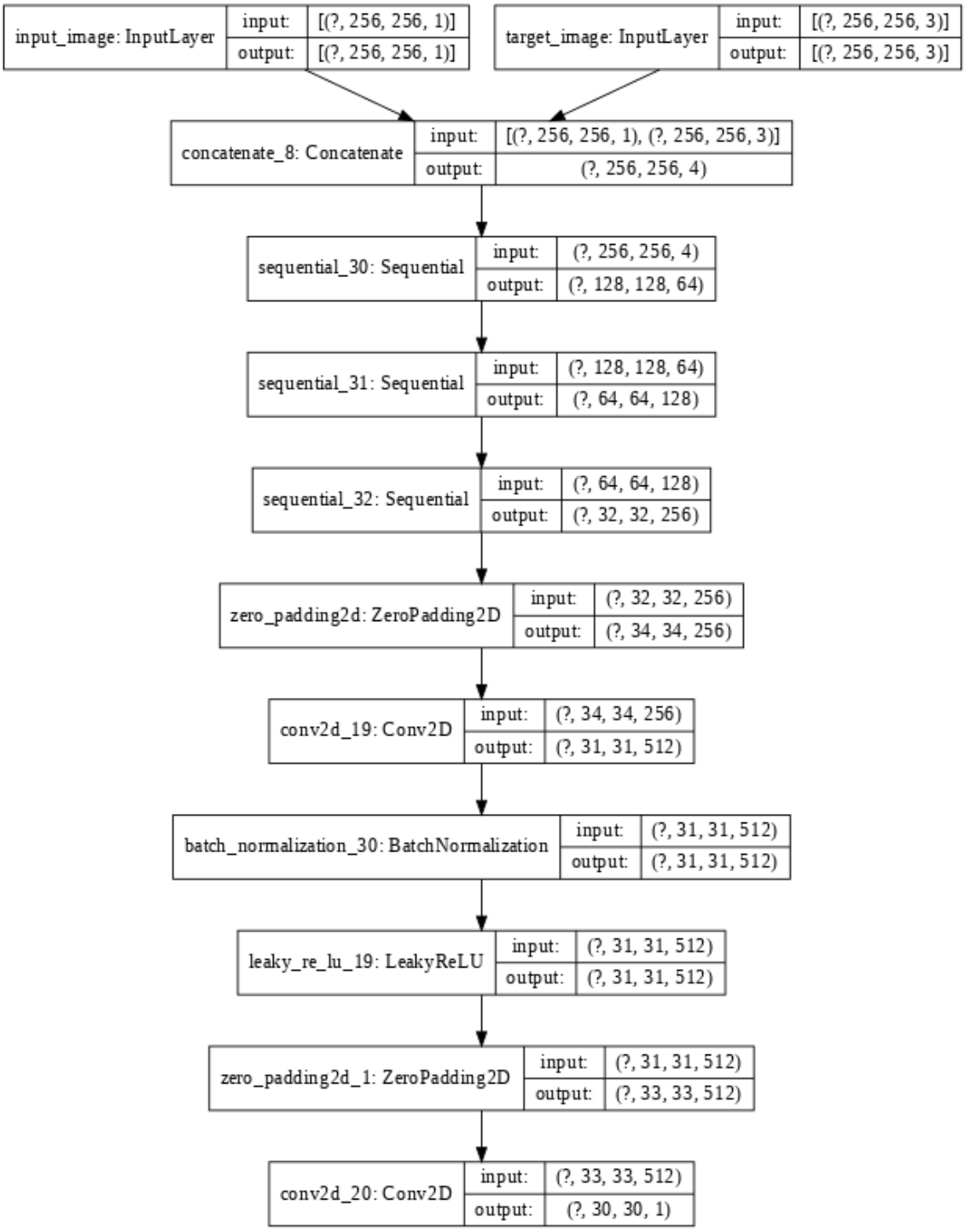
—Pix2Pix model architecture—



—Generator with random noise layer architecture—



—discriminator architecture—



Bibliography

[1] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[3] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[4] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[5] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[6] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[7] Steve Lawrence et al. “Face recognition: A convolutional neural-network approach”. In: *IEEE transactions on neural networks* 8.1 (1997), pp. 98–113.

[8] Shengfeng He et al. “Supercnn: A superpixelwise convolutional neural network for salient object detection”. In: *International journal of computer vision* 115.3 (2015), pp. 330–344.

[9] Yu Xiang et al. “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes”. In: *arXiv preprint arXiv:1711.00199* (2017).

[10] Tong He et al. “Text-attentional convolutional neural network for scene text detection”. In: *IEEE transactions on image processing* 25.6 (2016), pp. 2529–2541.

[11] Guanbin Li and Yizhou Yu. “Visual saliency detection based on multiscale deep CNN features”. In: *IEEE transactions on image processing* 25.11 (2016), pp. 5012–5024.

[12] Shuiwang Ji et al. “3D convolutional neural networks for human action recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2012), pp. 221–231.

[13] Pedro Pinheiro and Ronan Collobert. “Recurrent convolutional neural networks for scene labeling”. In: *International conference on machine learning*. 2014, pp. 82–90.

[14] Baotian Hu et al. "Convolutional neural network architectures for matching natural language sentences". In: *Advances in neural information processing systems*. 2014, pp. 2042–2050.

[15] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[16] Qi Qi et al. "Personalized sketch-based image retrieval by convolutional neural network and deep transfer learning". In: *IEEE Access* 7 (2019), pp. 16537–16549.

[17] Tao Chen et al. "Sketch2photo: Internet image montage". In: *ACM transactions on graphics (TOG)* 28.5 (2009), pp. 1–10.

[18] Mathias Eitz et al. "Sketch-based image retrieval: Benchmark and bag-of-features descriptors". In: *IEEE transactions on visualization and computer graphics* 17.11 (2010), pp. 1624–1636.

[19] Rong Zhou, Liuli Chen, and Liqing Zhang. "Sketch-based image retrieval on a large scale database". In: *Proceedings of the 20th ACM international conference on Multimedia*. 2012, pp. 973–976.

[20] Yonggang Qi et al. "Sketch-based image retrieval via siamese convolutional neural network". In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2016, pp. 2460–2464.

[21] Pier Paolo Ippolito. *Feature Extraction Techniques*. URL: `https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be`.

[22] Tu Bui et al. "Sketching out the details: Sketch-based image retrieval using convolutional neural networks with multi-stage regression". In: *Computers & Graphics* 71 (2018), pp. 77–87.

[23] Mathias Eitz, James Hays, and Marc Alexa. "How do humans sketch objects?" In: *ACM Transactions on graphics (TOG)* 31.4 (2012), pp. 1–10.

[24] Patsorn Sangkloy et al. "The sketchy database: learning to retrieve badly drawn bunnies". In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), pp. 1–12.

[25] Antonia Creswell and Anil Anthony Bharath. "Adversarial training for sketch retrieval". In: *European Conference on Computer Vision*. Springer. 2016, pp. 798–809.

[26] Ling Huang et al. "Adversarial learning for content-based image retrieval". In: *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE. 2019, pp. 97–102.

[27] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[28] Ouhda Mohamed, Ouanan Mohammed, Aksasse Brahim, et al. "Content-based image retrieval using convolutional neural networks". In: *First International Conference on Real Time Intelligent Systems*. Springer. 2017, pp. 463–476.

[29] Ruslan Salakhutdinov. "Learning deep generative models". In: *Annual Review of Statistics and Its Application* 2 (2015), pp. 361–385.

[30] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[31] Donald Olding Hebb. *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall, 1949.

[32] Irhum Shafkat. *Intuitively Understanding Variational Autoencoders*. URL: `https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf`.

[33] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[34] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *arXiv preprint arXiv:1401.4082* (2014).

[35] Carl Doersch. "Tutorial on variational autoencoders". In: *arXiv preprint arXiv:1606.05908* (2016).

[36] Jakub Langr and Vladimir Bok. *GANs in Action: Deep Learning with Generative Adversarial Networks*. Manning Publications, 2019.

[37] Phillip Isola et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[38] Irhum Shafkat. *pix2pix software*. URL: `https://phillipi.github.io/pix2pix/`.

[39] Jian Zhao et al. "Generating photographic faces from the sketch guided by attribute using GAN". In: *IEEE Access* 7 (2019), pp. 23844–23851.

[40] Runtao Liu, Qian Yu, and Stella Yu. "An unpaired sketch-to-photo translation model". In: *arXiv preprint arXiv:1909.08313* (2019).

[41] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

[42] Martin Heusel et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems*. 2017, pp. 6626–6637.

[43] Arnab Ghosh et al. "Interactive sketch & fill: Multiclass sketch-to-image translation". In: *Proceedings of the IEEE international conference on computer vision*. 2019, pp. 1171–1180.

[44] Kingma Da. "A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[45] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[46] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. arXiv: `1405.0312 [cs.CV]`.

[47] Zhe Cao et al. *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. 2018. arXiv: `1812.08008 [cs.CV]`.

[48] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv: 1512.00567 [cs.CV].