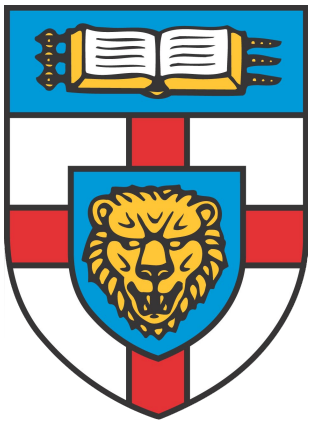

GOLDSMITHS, UNIVERSITY OF LONDON

FINAL YEAR PROJECT

Deep Image Colourisation: comparing AutoEncoders and Conditional Adversarial Networks

Author:
Muneeb Rehman

Supervisor:
Dr. Tim Blackwell

A thesis submitted in fulfillment of the requirements for BSc Computer Science Degree

May 6, 2022

Declaration of Authorship

I, Muneeb REHMAN, declare that this thesis titled, "Deep Image Colourisation: comparing AutoEncoders and Conditional Adversarial Networks" and the work presented in it are my own. I confirm that:

- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: Muneeb Rehman

Date: 05/05/2022

UNIVERSITY OF LONDON

Abstract

Computing Department

BSc Computer Science Degree

Deep Image Colourisation: comparing AutoEncoders and Conditional Adversarial Networks

by Muneeb REHMAN

Image colorization is the process of taking a grayscale image and adding color information to it in a way that looks realistic. Early methods for colorization relied on user-guided techniques, such as scribbling and using reference images. However, nowadays deep learning techniques for image colorization have been introduced, which eradicates the need for user guidance. There are several types of deep learning architectures that can achieve image colorization, such as AutoEncoders and conditional adversarial networks. This investigation only focuses on deep learning techniques for image colorization, specifically AutoEncoders and cGANs. The investigation aims to compare the performance between AutoEncoders and cGANs using quantitative metrics. In addition to this, the investigation also looks into whether objective metrics, such as PSNR and SSIM, correlate with human opinion in regard to image colorization and determines which method of evaluation is best suited when analyzing image colorization methods. The limitations suffered from each method and possible future research are also mentioned. The investigation concludes that using objective metrics for evaluation is unreliable for measuring colors from the image and using user studies is the optimal solution. The cGAN models performed overall the best, by producing more realistic colorization and adding a higher variety of colors.

Keywords - Image Colourisation, Convolutional Neural networks, AutoEncoders, cGAN, evaluation

Acknowledgements

Firstly, I would like to thank my supervisor, Tim Blackwell, for providing me with support and guidance along the way. I would also like to thank my parents for their support throughout my studies.

Contents

List of Figures

List of Abbreviations

ANN	**A**rtificial **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
GAN	**G**enerative **A**dversarial **N**etwork
cGAN	**C**onditional **G**enerative **A**dversarial **N**etwork
GFE	**G**lobal **F**eature **E**xtractor
PSNR	**P**eak **S**ignal to **N**oise **R**ationetwork
SSIM	**S**tructural **S**imilarity **I**ndex **M**easure

Chapter 1

Introduction

1.1 Purpose and Motivation

The main **purpose** of this report is to investigate various techniques used for the application of image colourisation and restoration purposes using deep learning tactics such as AutoEncoders and Conditional Generative Adversarial Network (cGAN). These purposes may include applying colour to an old black and white image or restoring an old coloured image to its original form as accurately as possible.

The **motivation** for my project came from watching historical documentaries that were recently colourised using artificial intelligence. The colourisation added a lot more sense of immersion; it made every human being, every car, every horse, and everything look more real and alive. What was more impressive was that it was all done by AI. This fascinates me because, for the human mind, the idea of image colourisation is a simple task. We learn this from a very early age—to fill in the missing colours from a colouring book, by knowing the fact that the sky is blue, the grass is green, clouds are white, apples are red or green. Using deep learning, we can replicate that mindset for a machine to automate the process of colourising photos for us.

On top of that, I also thought that having such a model would allow it to be a useful tool for users to apply colours to old black and white image artefacts or restore videos and images that have suffered deterioration in quality over time.

Moreover, AI image colourisation can be used to reveal semantic meaning or details that may have otherwise been left out or lost from the image. Besides that, having an automated colourisation process can eliminate the need to hire an artist (or can be used to assist them) to colourise or restore your photos, as such a model will be able to do what an image restoration artist can, for free and fast.

1.2 Aims

The **aim** of this project is to determine the best image colourisation method by **carrying out analysis** of **deep learning architectures such as AutoEncoders and Conditional Adversarial Networks** using **quantitative measures such as human assessment and objective metrics**.

In addition to this aim, the project will investigate whether the **objective measures correlates with human assessment** results and **determine which measure matters more for the purpose of image colourisation**.

1.3 Scope

The **scope** of this research project is to conduct an analysis and evaluation of architectures such as AutoEncoders and cGAN for the task of image colourisation, and determine which method is best suited for this task. A total of four existing method architectures will be used for this study. The first two methods are AutoEncoders whereas the second two are cGANs. All methods will be trained using a dataset, Places365 which is split into 60:20:20 for training, testing and validation respectively. PSNR and Naturalness perception study will be the metrics used to evaluate each method's performance in the test set. Visualisations will also be used to showcase the colourisation ability.

1.4 Section Overview

Chapter 2: Background Research: This chapter goes over the intuition about the history and process of image colourisation, as well as the area of research within deep learning and similar projects.

Chapter 3: Methodology This section of the report will go over the step by step methodology used to conduct my research, including the data preprocessing. Here, I will be describing the methods I have used to experiment with image colourisation.

Chapter 3 Research results: In this chapter, I will be going through the results gathered from the implementation and training. Each method will be tested against unseen test data. Their results will be in the form of visualisations and quantitative metrics used to evaluate each method's performance and determine which class method is best suited for the purpose of image colourisation. In addition to that, I will discuss the results along the way and go over any existing limitations suffered from each method.

Chapter 4 Conclusion future work: The final chapter of the report will conclude the investigation by going over what has been accomplished along the way but also possible improvements which may yield better results for this investigation as future work.

Chapter 2

Background research

2.1 AI image colourisation

The problem of automated image colourisation from the previous century did not receive much attention. The process was carried out manually, mostly by artists. Back then, early photography lacked the developing agent which as a consequence made it more prone to quality deterioration due to the conditions of the environment. These environmental factors can be the humidity from the air, dust particles, temperature and mould. Other factors can be due to inappropriate storage, mishandling of images or improper image restoration process [48]. Additionally, early photography was in black and white, and the need for colourising them was in demand. Image restoration artists were often required to maintain these images and also enhance them (e.g add colourisation).

It wasn't until this century when in 2004 when Anat Levin and Dani Lischinski proposed an automated solution [33] that required users to scribble colours on a grayscale image, and then the process will apply the colour to regions across the image.

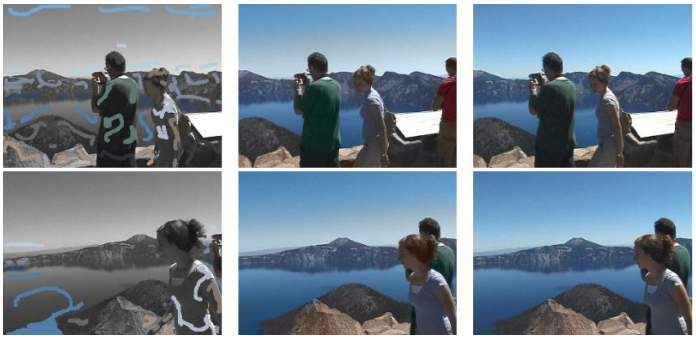

FIGURE 2.1: Anat Levin and Dani Lischinski's image colourisation method

This in turn significantly reduces the need for human intervention in the process of image colourisation. However, the limitation of Anat Levin and Dani Lischinski's method was that it was considered slow in producing the results. Over time, later research provided more automated methods for image colourisation. Whilst the methods provided by Anat Levin and Dani Lischinksi required one or several referenced images (provided by users or received automatically) as source data. The model learns to transfer colour across regions as shown in figure 2.1.

On the other hand, newer methods involve a function that learns predictions from colour images

from an extensive dataset during training, and the colourisation problem will either be a regression within the continuous colour space or a classification of colours with associated objects in the image.

2.1.1 Image colourisation projects

This section goes over several existing projects relating to image colourisation.

Deoldify

Deoldify is a well known black and white image colourisation library developed by Jason Antic[1]. This project uses state of the art techniques such as GANS and has impressive colourisation results. Recently, Deoldify has developed a new type of GAN training method called No-GAN which is a model that addresses the many issues faced by his previous GAN model. A few of these issues are disruptive visual artefacts, bugs and inaccurate colouring. According to Jason Antic, the use of No-Gan has reportedly shown impressive results and achieves realistic colourisation.

FIGURE 2.2: Image colourised using Deoldify

Scribbler

Scribbler (not to be confused with Anat & Dani's scribble method) is an interesting technique that involves turning pictures of sketches into realistic images using generative techniques[43]. Even though their aim doesn't directly align with mine (image colourisation), their model can still generate realistic images with colours. They used a technique similar to [33], which involves user guidance by scribbling colours onto a sketch and will systematically apply those colours, whilst at the same time transforming the sketch into a realistic image. Their method is trained using in a GAN manner.

FIGURE 2.3: Turning sketches into realistic images using Scribbler

Let there be Color!

Similarly, "Let there be Color!" is another project that uses deep learning techniques to colourise black and white images, and is developed by [25]. They use an AutoEncoder which has been slightly modified to include a global feature extractor model. The addition of this modification meant the colourisation could be more accurate by applying relevant colours of the predicted class of the image. Their model has shown impressive colourisation results, as seen below in figure 2.3:

FIGURE 2.4: Image colourisation results

End-to-End Image Colourisation

The End-to-End conditional GAN-based image colourisation paper by [8]. They use a conditional GAN architecture which is based on a well known cGAN architecture "Pix2Pix".

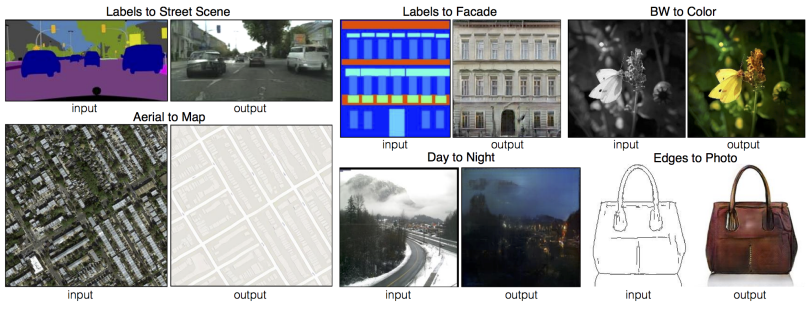

FIGURE 2.5: Pix2Pix image translation examples

Pix2Pix is a type of generative adversarial network used for image-to-image translation. Pix2Pix was presented by Phillip Isola in his 2016 paper [26]. This architecture is used for many purposes that involve image-to-image translation as shown in the figure above. The author of End-to-End Image Colourisation decided to use this base architecture for image colourisation.

ChromaGAN

The ChromaGAN is a conditional generative adversarial network proposed by the students from the University of Pompeu, Spain 2020 [44]. Their proposal is primarily used for colourising grayscale images from semantic details engraved within the scene. Their method is an adaptation of **Let there be Color** since the generator of the architecture is inspired in the sense that it utilises a global feature extractor, but they decided to train this model in a GAN setting alongside a discriminator.

FIGURE 2.6: examples of ChromaGAN image colourisation

2.1.2 Comparative literature review

Title	Category	Dataset	Dataset size	Accomplishment	Limitation
Deep colorization	plain CNN network	SUN dataset	1519 images (33 object categories)	- Simple network and minimises user effort	- Limited performance: Can only colourise simple scenes. - Visual abstracts: produces interfering abstracts which rely on bilateral filtration to remove them.
Scribbler	AutoEncoder	CUHK dataset	1869 images	- User can control the colourisation process. -Can transform sketches into realistic photos	- Results are usually subject to user preferences. - Blurry edges and leaking colours.
Let there be color!	AutoEncoder	Places365	2,327,985 (19,1546 for testing) (205 categories)	- Incorporates transfer learning technique to improve colourisation process. - Can colourise without user intervention.	- Complex architecture: Contains many parameters which may significantly increase the training time. - Biased: The results are subject to the knowledge seen within the training set.
End-to-end (pix2pix)	cGAN	ImageNet	50,000 images (10,000 for testing) (50 categories)	- Solves vanishing gradient problem seen with prior image colourisation methods. - Uses skip connections, allowing it to use RGB images for the generator instead of LAB and YUB colour space. - Uses PatchGAN as a discriminator which has significantly improved GAN performance.	- Some patch sizes may produce visual abstracts. - Complex architecture and slow training
ChromaGAN	cGAN	ImageNet	1,300,000 images	- Performs realistic colourisation and significantly reduces visual abstracts. - The model can be trained unsupervised	- Complex architecture and slow training. - Computationally expensive
Deoldify	cGAN	ImageNET	260,000 images	- The use of "NoGAN" significantly speeds up training and has resulted in reduced visual abstract and improved colourisation. - Provides many modes: Stable, Artistic, Video.	- No official paper has been released yet

FIGURE 2.7: Advantages and disadvantages of each author's method

2.1.3 RGB and LAB mapping

In the task of image colourisation, the aim is to predict the colour channel of the given grayscale images. However in order to predict, a mapping between the grayscale pixel and the corrosponding coloured pixels is needed to be found defined colour space.

FIGURE 2.8: Image colourisation is a task of finding a mapping

As seen in the above figure, the task mainly involves finding a mapping such that a grayscale image (input) can be mapped to its corresponding coloured output. This mapping can be achieved by using a training set to discover all the related patterns used for mapping between grayscale and coloured. Here, the mapping can be achieved using two existing colour space representations: RGB and LAB.

FIGURE 2.9: examples of ChromaGAN image colourisation [46]

One of the common representations is the RGB. This representation is widely used in electronic displays, such as computers, televisions, and cell phones. The RGB color model is an additive model in which red, green, and blue light are added together in different combinations to create a wide range of colors [30]. For the task of colourisation, this representation will involve finding a mapping function to the three channels (as seen in the figure above).

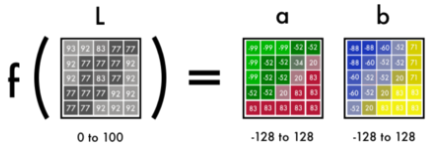

FIGURE 2.10: examples of ChromaGAN image colourisation

There are other kinds of representations like LAB where L stands for luminance, and A and B for the colour spectra green–red and blue-yellow [35]. Using the LAB colour channel in terms of image colourisation offers convenience as the L* channel represents the grayscale of the image and can be used as a conditional input. So while using the LAB representation, only 2 channels need to be mapped.

2.2 Similar Work

2.2.1 Human VS Objective evaluation of colorisation performance

Sean Mullery and Paul F. Whelan wrote a research investigation into evaluating image colourisation results using human and objective measures. Their research concerns more on whether the objective measures correlate with human opinion on images colourised using deep learning. They conducted a human evaluation test using Amazon mechanical Turk using 19 participants, and using their responses, they compared that against a dozen objective measures, to see if any of the objectives have a strong correlation with human opinion. The investigation concluded that the objective measures utilised in the colourisation literature did not correlate well with human opinion, with MS-SSIM showing the highest correlation, however, the correlation was too low to be considered an appropriate solution. They also reported on the behaviour of the observers, and noticed how most observers were tolerant to minor changes, but penalised heavily on medium to large changes [38].

2.2.2 Image Colorization: A Survey and Dataset

A comprehensive survey of various image colourisation methods was investigated by Saeed Anwar and Muhammad Tahir. This investigation was to address the need for a survey in the field of image colourisation. The survey involves the results of experiments on existing image colourisation methods. Here, they performed an extensive evaluation on several classes of architectures which are: Plain networks, user-guided networks, domain-specific networks, text-based networks, diverse colourisation networks, multi-path networks, and exemplar-based methods. Their investigation also discusses the many benefits and limitations of each existing colourisation method as well as mentions the same for datasets. Their report concluded that the current use of metrics used by many existing colourisation methods is inadequate to evaluate the colourisation, and also discovered how most image colourisation methods performed, such as how GAN-based methods delivered diverse colourisation whereas CNN-based methods tend to deliver sub-optimal results for complex scenes [4].

2.2.3 A review of image colourisation

A review of image colourisation was a report carried out by Bo Li, Yu-Kun Lai and Paul Rosin. Their paper discusses the history of image colourisation and goes on to mention automated-based image colourisation methods, with which they experimented. They perform an extensive evaluation of reference-based, user guidance-based and deep learning-based image colourisation methods and compare each method to see look for similarities and differences. Their report concluded that deep learning methods produced robust and meaningful results, however, were difficult to control. They also reported how no quantitative metrics exist to measure the quality of colourised results, mainly because how current metrics do not provide accurate measures for coloured images. They plan to investigate and develop a metric useful for coloured images as future work[34].

2.3 Generative Neural Networks

In deep learning there are neural networks and then there are generative neural networks. Here, I will only be mentioning the latter, so refer to Appendix B for information regarding Neural Networks.

2.3.1 Auto encoder

The autoencoder is a class of CNN, it is used to learn efficient coding of unlabeled data (unsupervised learning). Autoencoders attempt to learn representations (encoding) for a given set of data by decomposing the data into small bits of data (bottleneck). Then, the Autoencoder will use that representation to reconstruct (decoder) the original data [29].

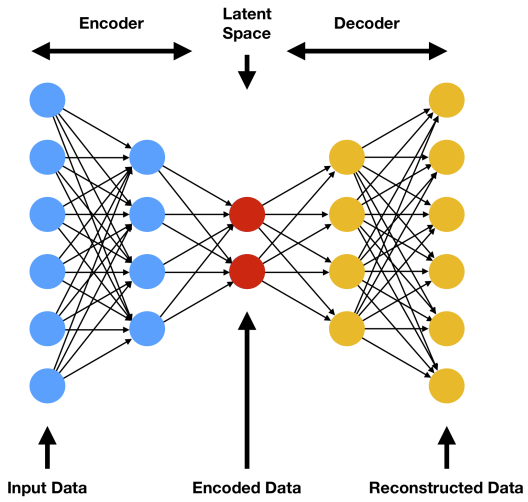

FIGURE 2.11: Example of an AutoEncoder model [18]

2.3.2 Generative Adversarial Networks

A generative adversarial network (GAN) was introduced in 2014 by Goodfellow et al[20]. The process of GAN involves a generator that generates an image, and its job is to fool the discriminator into thinking the generated image is real. As mentioned, the discriminator is used to determine whether the images presented by the generator are real or fake. If the image is deemed to be fake, the discriminator will pass feedback to the generator to suggest what the generator must do to improve its skill of making images that are indistinguishable from the real ones[9]. Both the generator and the discriminator are trained to outperform one another, eventually resulting in images that are increasingly realistic, making it impossible for the discriminator to tell the generated and the real images apart[13].

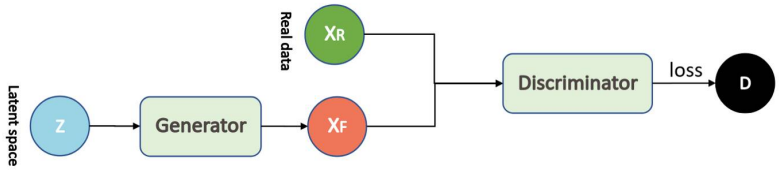

FIGURE 2.12: Example of a gan model [6]

2.3.3 conditional GAN

A cGan or conditional GAN is a generative adversarial network (GAN) that is conditioned on an additional input, typically an image. The cGan is different from a regular GAN in that it can generate images that are conditioned on a specific input, giving it the ability to generate images that are more realistic and diverse. The first cGan was developed by Mirza and Osindero in 2014. Some use cases for a cGan include generating realistic images, synthesizing images from text, and creating photo-realistic images[19].

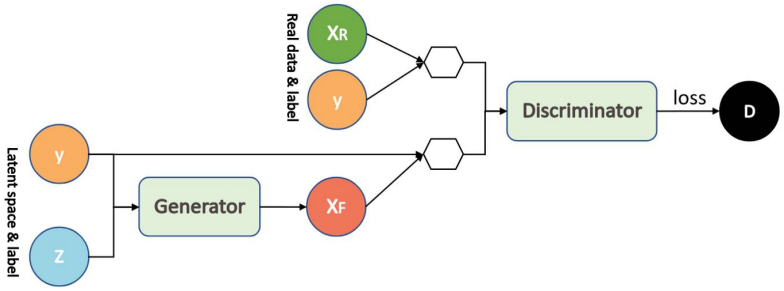

FIGURE 2.13: Example of a cGan model [6]

2.4 Dataset

In order for Machine learning models to understand how to perform various tasks, they'll first need to be trained using a sufficient amount of dataset. These datasets can be in various forms, such as numerical, categorical or time series, in my case I will be using numerical as it is a way of representing images. Datasets are often split up into 3 partitions: Training set which will be used to train the model; Validation set will be used to evaluate the model's performance during training; Testing set will be used to test the model's accuracy and performance after it has been trained. Since the models will perform image colourisation tasks, it'll need to be trained using an image dataset. Below are a few image datasets I have looked into that I can use to train the models:

2.4.1 SUN dataset

The Scene UNderstanding (SUN) dataset is a well-sampled dataset that contains about 130,519 annotated images spread over 899 categories [49]. The dataset is used by researchers specialising in the fields of computer vision, human perception, cognition, machine learning, and data science. Typically, the images in the dataset contain environmental scenes and everyday objects, which are all annotated with their corresponding category and name. The number of images in the SUN dataset is expanding constantly as new images are being annotated every day. In my case, this dataset provides a rich amount of images, which is already beyond the perfect amount needed to train the models.

2.4.2 Places dataset

The places dataset is an open-source large-scale image dataset used mainly for scene recognition in computer vision. The dataset contains 10 million scene photographs with annotations which makes it the largest existing scene-centric image dataset. This dataset is used in many applications within the world of computer vision, such as classification problems. The dataset contains a large number of natural images from a variety of places, including indoor and outdoor scenes. The dataset is divided into 365 categories, and each category contains a large number of images.[53].

2.4.3 Open Images Dataset

The open images dataset is an open-source image dataset from Google. This dataset contains a significant amount of over 9 million images with annotations. It contains a total of 16 million boxes for 600 object classes on 1.9 million images, which makes it the largest existing dataset with object location annotation [32]. Like the SUN dataset, the open image dataset is expanding every single day, with each release adding millions of more annotated images.

2.5 Deep Learning tools

2.5.1 Tensorflow

Tensorflow is a powerful library used for **numerical tensor computation** and is primarily used for **creating and deploying large-scale Machine Learning models** and offers a plethora of tools. The library was first released for open source in 2015 by Google Brain Team and is now regarded as one of the **most popular deep learning frameworks** to this day[21].

2.5.2 Keras

Keras is an open-source, deep-learning python library **developed by François Chollet in 2015** [13]. The framework provides an interface used for defining and training any kind of deep learning model. However, for tensor computation, Keras has to rely on a **backend engine** such as TensorFlow which makes Keras a **high-level framework**.

2.5.3 Scikit-learn

Scikit-learn is an open-source machine learning library used for the Python programming language and was **released in 2007 by French Data Scientist David Cournapeau**. The library **features several machine learning algorithms for classification, regression and clustering**. Additionally, the library offers utilities for applying transformations to data such as **Lab2RGB** and **RGB2LAB** (which will be used for this throughout my research project). NumPy is utilised for performing high-performance linear algebra and array operations [39].

2.5.4 OpenCV

OpenCV is an open-source multi-platform **computer vision library** that you can use to implement **object recognition, object tracking, 3D reconstruction, image matching, and image stitching**. It provides many powerful image processing and computer vision functions. It is **widely used by developers to develop computer vision** and machine learning applications[28].

Chapter 3

Research Methodology

3.1 The dataset

Before the model can be trained, tested and validated, I will need to acquire a dataset. The dataset of choice was the "Places365" dataset [52]. This dataset contains images of places, specifically streets, buildings, mountains, glaciers, and trees, but also contains images of objects and entities such as cars, planes, people and animals. Additionally, the set contains about 105,000 256x256 images across 65 classes split 60:20:20 for training, testing and validation respectively. The dataset is primarily used for image classification and analysis, but in my case, the task will be image colourisation [36].

3.2 Data Preprocessing

To ensure the models can interpret the data for training and testing, the data should be prepossessed. In my case, a **tensorflow pipeline** was used. The pipeline takes images in batches and will apply numerous transformations. These transformations may involve:

1. Decoding batches of images into numerical tensors is achieved using OpenCV's "imread" function.
2. Creating input and targets. Inputs and outputs can be either grayscale input and RGB output for methods that perform RGB mapping, or $L*$ and $A*B$ respectively for methods that perform LAB mapping.
3. Resizing the images to a constant size of 256 x 256 to ensure consistent dimensionality.
4. Scaling each pixel of the image down to a range of [0,1]. This is done by dividing the colour channels from the images by the value of 255 (as RGB ranges between 0-255)
5. Augmentation transformations are applied. These augmentations involve random cropping, flipping and zooming. Note that augmentation isn't used when using the testing set pipeline.

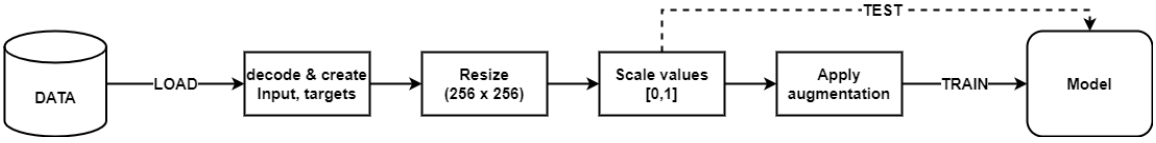

FIGURE 3.1: Preprocessing pipeline

3.3 AutoEncoder method

Here I've explored two AutoEncoder methods, Simple AutoEncoder and the Global AutoEncoder.

Both models adopt the LAB colour space mapping since this colour space minimises the correlation between the three coordinate axes of the colour space.

3.3.1 Simple AutoEncoder

Model Architecture

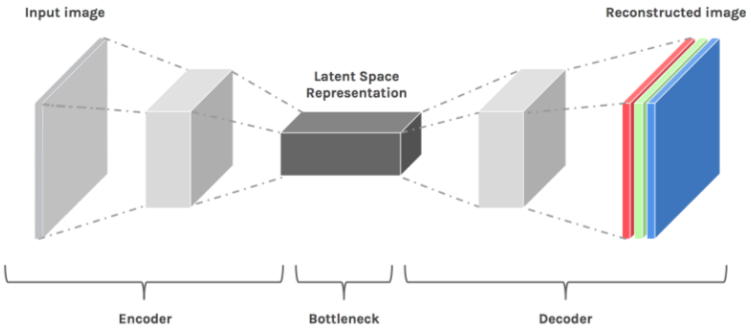

FIGURE 3.2: AutoEncoder architecture [14]

The Simple AutoEncoder is regarded as the first method I experimented with during my research into image colourisation. The Architecture being used is based on [25] but without the Global Feature Extractor. For the sake of this report, this AutoEncoder will be referred to as **Simple AutoEncoder**.

This architecture is considered simple as out of all the methods I have experimented with, this model uses the fewest amount of parameters.

The basic anatomy of this AutoEncoder involves three distinct parts: the Encoder unit, Bottleneck & the Decoder unit.

Encoder unit

The low-level features are encapsulated as part of the encoder. The encoder extracts features from the input image and comprises 10 convolution layers, each consisting of 3x3 kernels. The input of the encoder is the grayscale image, which represents H x W x 1, where H and W refer to the image's height and width. The result of the encoder represents H/8 x W/8 x 512. Setting each layer's padding to "same" helps to preserve the original image size. Strides are used at each convolution layer, this helps to reduce the output of the image by 50%. Ultimately, the output image is decomposed into a latent space representation.

Bottleneck unit

The bottleneck unit of the autoencoder is the unit in the middle of the network that has the smallest number of neurons. This unit provides the lowest level of abstraction for the input data and forces the autoencoder to learn to compress the input data into a lower-dimensional representation.

Decoder unit

The decoder takes the bottleneck output as input and uses it to estimate the colours of the image. During this process, several upsample layers are used to increase the spatial H x W of the feature maps, using the UpSample2D. The produced output is the roughly estimated A*B* channels, with the H x W x 2 volume representation. The output of the decoder is then concatenated with the input grayscale channel to form the full L*A*B* image, which is then converted to RGB representation.

3.3.2 Global AutoEncoder method

Model Architecture

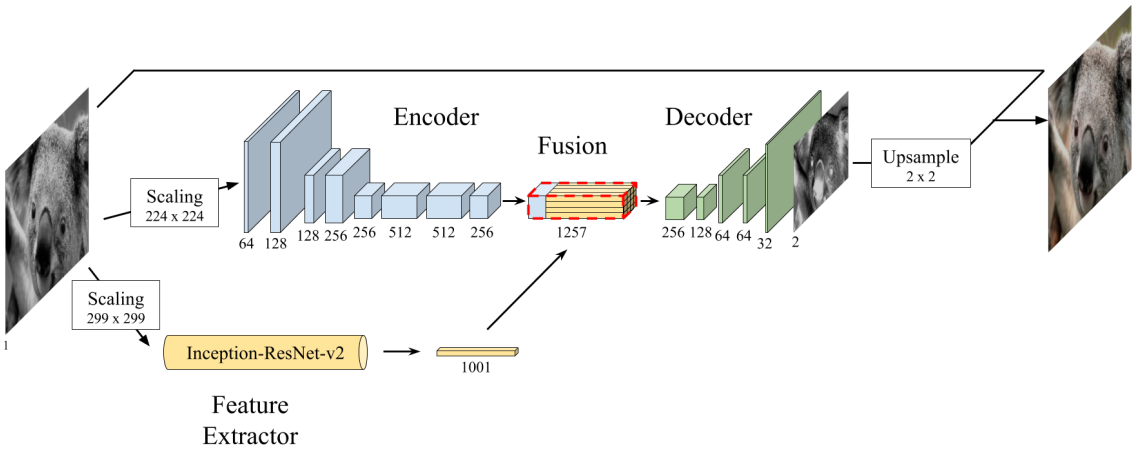

FIGURE 3.3: Deep Koalarization architecture

The base architecture used for the image colourisation model is a form of an autoencoder and was developed by Federico Baldassarre, Diego Gonzalez-Morin and Lucas Rodes-Guirao [16]. Their model was based on the authors of [25] (as seen in Figure 11).

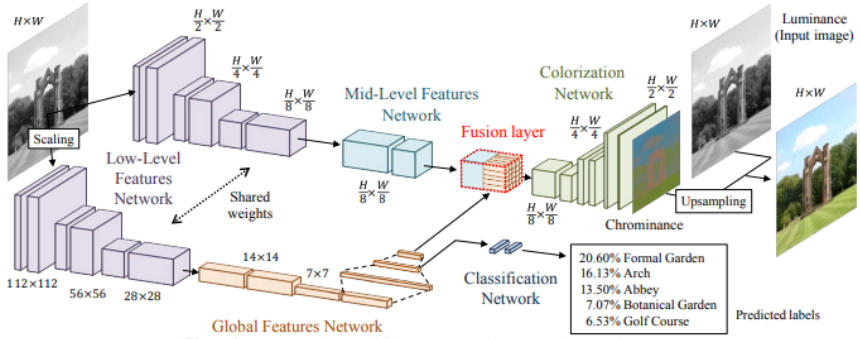

Figure 2: *Overview of our model for automatic colorization of grayscale images.*

FIGURE 3.4: Let there be color architecture

[16] (figure 3.3) use a pre-trained model (Inception-Resnet-V2) as a global feature extractor, whereas [25] (figure 3.4) use their own trained model for this purpose. The use of a pre-trained model for the global feature extractor eliminates the need to train a separate model, as the inception-ResNet-v2 has been trained over 1000 classes of images.

This model consists of four parts: the encoder, global feature extractor, fusion layer, and decoder. The encoder takes in the L channel (grayscale) and extracts low-level features. The input is also fed into a global feature extractor (GFE) which extracts global features. The fusion layer then combines

the outputs of GFE and encoder. The fusion layer's output is then fed into the decoder, which does the colourisation task by mapping outputs to their corresponding colour channels.

The base AutoEncoder (without the global feature extractor) architecture used is identical to the previous simple AutoEncoder mentioned previous section, hence I will not go over the implementation again for this method, but rather will go over the highlighted differences. For the sake of this report, I will be referring to this architecture as **Global AutoEncoder**.

Global feature extractor

The global feature extractor helps to identify and extract global features from the image and outputs a class distribution. The model is a pre-trained inception-resnet. It takes as input a grayscale image input which is upscaled to H x W of 299 x 299 respectively. The output of GFE will result in the image representing a 1000-dimensional class distribution vector. The point of the GFE is to indicate the class of the image from the given input so that it can help identify relevant colours. For instance, if the GFE identified the image as being an indoor room consisting of furniture, the GFE will mitigate the encoder from being biased, so that it will add appropriate colours for furniture, rather than sky or grass colours.

Fusion Layer

The fusion layer combines the output of the GFE (a 256 dimensional vector) and the encoder. The use of the fusion layer results with a H/8 x W/8 x 256 layer. This effectively merges the information gathered from the GFE and the encoder into one layer. The formula for fusion can be seen as:

$$y_{u,v}^{fusion} = \sigma\left(b + W\begin{bmatrix} y^{global} \\ y_{u,v}^{mid} \end{bmatrix}\right) \tag{3.1}$$

Where $y_{u,v}^{fusion}$ is the fusion of features u, v. Where y^{global} is the global feature vector and $y_{u,v}^{mid}$ is the mid level feature at (u, v). W is the weights and B is the bias. Both, W and B are trainable variables.

3.4 Conditional GAN method

I have experimented with two different types of conditional GANs, ChromaGAN and Pix2Pix, to see if they perform better than the AutoEncoder methods that were discussed earlier.

3.4.1 Pix2Pix GAN

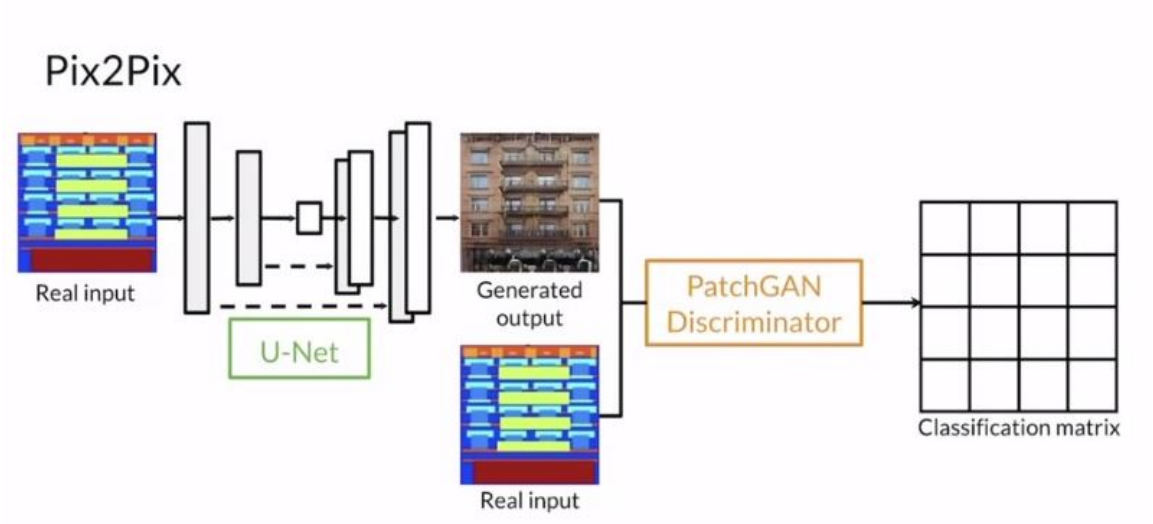

FIGURE 3.5: Pix2Pix Architecture [26]

Pix2Pix is a category of cGAN and is used for image to image translation. This network not only can learn the mapping from an input image to an output image but also learns the loss function to train this mapping.

Pix2pix was developed as a more sophisticated approach to image-to-image translation, compared to the basic CNN method which simply minimizes the Euclidean distance between the predicted and ground truth pixels. However, it was seen that this method would result in a blurry output, as the Euclidean distance is minimized by averaging up all the outputs, causing the blurring effect.

In brief, the architecture works by learning a loss function that tries to classify the output image as being either real or fake, while simultaneously training a generative model to minimise this loss. Blurry images will not be tolerated as they will easily look fake.

Generator Architecture

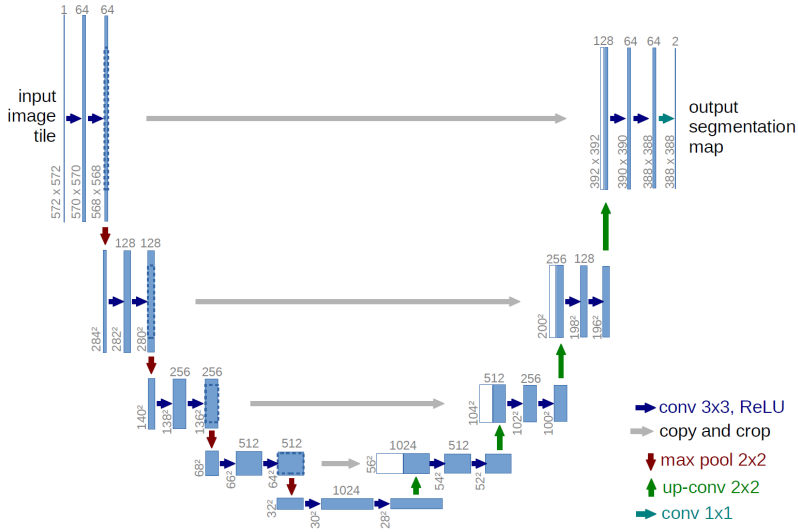

FIGURE 3.6: U-NET model [41]

The most important aspect of the cGAN is the generator, as this is responsible for generating the coloured output. As discussed earlier, Pix2Pix can only work if the output image is not blurry, and having such blurry images will only help the discriminator in identifying the generated images. The AutoEncoder method I discussed earlier uses an encoder-decoder network. In these networks, the input image is progressively decomposed through the layers until it reaches a bottleneck, and from there the process is reversed. However, since the information is constantly being decomposed, information that helped retain the structures of the input image is risked being lost.

To mitigate this issue, the generator can utilise skip connections which creates the shape of a "U-Net". This architecture adds skip connections between each *ith* and the $n - i$, where n represents the number of layers in the network. Using skip connection help pass information between the encoder and the decoder, information that involves the underlining structure, which helps mitigate the blurry effect, and also why this architecture can use the RGB mapping, previously sought to produce blurry images an AutoEncoder.

Discriminator Architecture

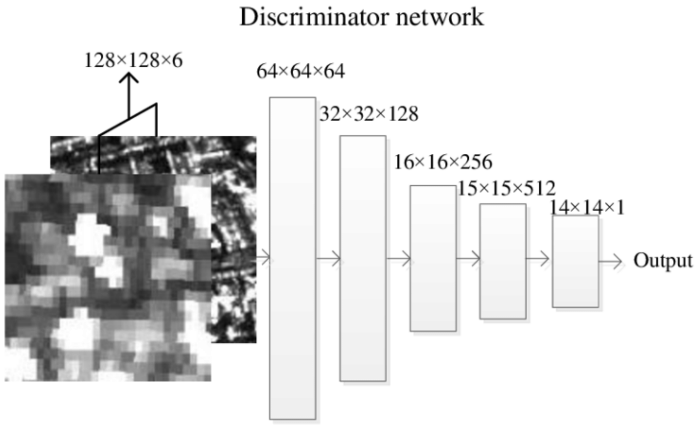

FIGURE 3.7: PatchGAN model [5]

The architecture of the discriminator used in Pix2Pix is based on the Markovian discriminator architecture, also known as patchGAN [15]. PatchGAN is used for binary classification. This discriminator is used to address blurry images when using L1 loss and L2 loss on high frequency, however, it turned out that having L1 loss switched to low frequency helped mitigate this issue. The way this architecture works is by dividing the input image into NxN patches and then identifying each patch of the image as either real or fake.

The size of the N patch was noted to influence the output of the image. Using smaller patches was reported to speed up training time, and the results often looked high quality. The discriminator is created using convolutional layers and LeakyRelU was used as this was reported to help solve the vanishing gradient problem [10].

3.4.2 ChromaGAN

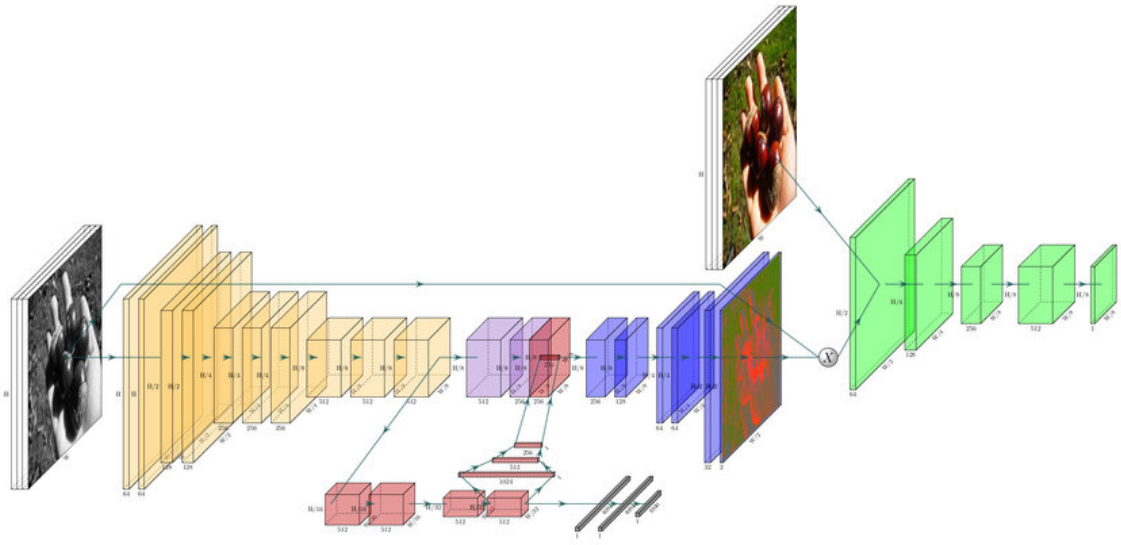

FIGURE 3.8: ChromaGAN Architecture

The ChromaGAN architecture is a conditional GAN. This architecture was developed by researchers from the University Pompeu Fabra [44]. This architecture was developed for the purpose of colourisation of grayscale images.

The difference between this architecture and the Pix2Pix method is the generator. ChromaGAN uses an AutoEncoder, which is similar to the global AutoEncoder discussed earlier. It performs LAB mapping and utilises a pre-trained global feature extractor, which is used to learn the class distribution vector. The distribution vector enables the information about the probability distribution within the semantic information contained within input L, and this helps improve the colourisation performance without the need for a labelled dataset.

Generator Architecture

The generator of the ChromaGAN is a network divided into two subnetworks: the AutoEncoder (yellow, purple and blue parts shown in figure 3.7) and the global feature extractor (red parts shown in figure 3.7). Here, I will be denoting both networks as G^1 and G^2. Both G^1 and G^2 take as input a HxW image. G^1 outputs chrominance information represented as $(a, b) = G^1(L)$, and G^2 outputs the class distribution vector $y = G^2(L)$. G^2 can only take in the input of channel size of 3, so the input image requires broadcasting, which involves concatenating the input image three times $[L, L, L]$. G^2 uses the pre-trained VGG-16 weights, but they are kept unfrozen during the process of training.

The encoder output of G^1 (represented in orange in figure 3.7) is then fused with the class distribution output produced from G^2. The fused output is then fed as input into the last stage of G^1, which is the decoder. The decoder will process this information through six modules of CNN layers with ReLu activation functions, along with two upsampling layers between them. The produced output is then concatenated with the input to form the full coloured image which will then be used against the adversary.

Discriminator Architecture

The discriminator of the ChromoGAN is the patchGAN. The patchGAN discriminator has already been discussed as pix2pix also uses the same discriminator. To be brief, it is noted that this architecture exceeds when working when capturing low-frequency structure of the input image, however, fails when working at high frequency. This process involves dividing the HxW input into patches and then classifies each individual patch of the image. The size of the patch was noted to influence the performance, as using smaller patches helped speed up performance and increase the quality of the input image produced by the generator.

3.5 Training and evaluation

Out of the **105,000** samples from the Places dataset, **84,000** was reserved for training whilst **21,000** was used for **testing and evaluation**.

All models were trained using **Slurm Facility** which is an workload manager for scheduling jobs for large scale cluster. [51]. The university provided me this service, and it allowed me to train the models practically for free.

Unfortunately, I could not leverage the time I had for optimisation. As a compromise, I will be sticking to the author's chosen hyper-parameter and will be performing quantitative testing using evaluation metrics. In the interim, I have performed tuning using a smaller dataset, please refer to Appendix B for this.

Nvidia Cuda Toolkit with **Nvidia Tesla M10** were used to enable GPU acceleration for faster training. The training time between each method was different, as some methods took 1 day to train whereas others took up to 4 days.

3.5.1 Objective functions

The objective function is a key component in deep learning because it provides a way to find the optimal solution [31]. The objectives may involve minimising or maximising a value. This value may represent an error used to quantify the distance between the predictions and the targets. The algorithm will continuously configure the model's parameters to minimise this error. Therefore, the use of an objective is essential in deep learning.

Since I have researched the 4 methods discussed throughout this section, it's known that each method has its own unique objective function.

AutoEncoder Objective function

Both AutoEncoder methods discussed so far use the same objective function. This objective function can be defined as computing the distance between the estimated output and the target output. A Mean Square Error loss function is used to quantify the error between the estimated pixel and the target pixel. The objective function can be denoted as the following formula:

$$C(X,\theta) = \frac{1}{2HW} \sum_{k\in(a,b)} \sum_{i=1}^{H} \sum_{j=1}^{W} ||X_{k_{i,j}} - \hat{X}_{k_{i,j}}||^2 \tag{3.2}$$

Where θ represents the parameters of the model, $X_{k_{i,j}}$ and $\hat{X}_{k_{i,j}}$ denote the i, jth pixel value of the kth channel value contained in the target and reconstructed predicted image respectively. The $||.||^2$ is the euclidean distance used to compute the difference between $X_{k_{i,j}}$ and $\hat{X}_{k_{i,j}}$. Both methods employ the same optimiser which is Adam to minimise this objective. This optimiser is preferred because it can handle sparse gradients on noisy problems such as natural language and computer vision [11]. Both methods have an initial learning rate set at 0.0001.

Pix2Pix Objective function

The objective function for Pix2Pix is unique compared to the AutoEncoder methods mentioned above, as both entities of the cGAN, the generator and the discriminator are competing against one

another over the same objective. The anatomy of how Pix2Pix performs is through an objective function expressed below:

$$\arg\min_{G}\max_{D}\mathcal{L}_{cGAN}(G,D)+\lambda\mathcal{L}_{L1}(G)$$

Where G represents the Generator that is tasked to minimise the objective against the discriminator D which tries to maximise it. In addition to the objective, a $\mathcal{L}_{L1}(G)$ cost is added, and can be expressed as:

$$\mathcal{L}_{L1}(G)=\mathbb{E}_{x,y,z}\left[\|y-G(x,z)\|_1\right]$$

The $L1$ distance is used instead of the $L2$ as it was discussed earlier that using $L1$ encouraged less blurring in images.

The Adam optimiser is used to minimise this objective with the learning rate set at $2e-4$ and the momentum parameters set at $\beta_1=0.5$ and $\beta_2=0.999$.

ChromaGAN Objective function

The objective of the ChromaGAN is similar to Pix2Pix, both generator and discriminator are continuously competing against one another. The objective is defined by the following formula:

$$\arg\min_{G}\max_{D}\mathcal{L}_{cGAN}(G,D) \tag{3.3}$$

However, the difference lies within the $\mathcal{L}_{cGAN}(G,D)$ objective function:

$$\mathcal{L}_{cGAN}(G,D)=\mathcal{L}_{\mathrm{e}}\left(\mathcal{G}^1_{\theta_1}\right)+\lambda_{\mathrm{g}}\mathcal{L}_{\mathrm{g}}\left(\mathcal{G}^1_{\theta_1},D_w\right)+\lambda_{\mathrm{s}}\mathcal{L}_{\mathrm{s}}\left(\mathcal{G}^2_{\theta_2}\right) \tag{3.4}$$

The first term in the equation, $\mathcal{L}_{\mathrm{e}}\left(\mathcal{G}^1_{\theta_1}\right)$, is the color error loss of the encoder-decoder part of the generator. The second term, $\lambda_{\mathrm{s}}\mathcal{L}_{\mathrm{s}}\left(\mathcal{G}^2_{\theta_2}\right)$, is the class distribution loss produced by the VGG-16 model.

This objective is minimised using the Adam optimiser which has a set learning rate of 2e-5 and has the momentum parameters set at $\beta_1=0.5$ and $\beta_2=0.999$.

3.5.2 Evaluation Metrics

Due to the black box nature of Deep Learning models, extensive testing and validation is require to gain insight into the decision-making process [54]. This can be done using various techniques. Here, I will be using PSNR, SSIM and Naturalness study to conduct me evaluation.

PSNR score

The Peak Signal-to-Noise Ratio (PSNR) is an objective metrics used to calculate the ratio in decibels between two images. This ratio is used to measure the quality between the target image and the original image. The ratio is used to indicate the quality of the output image; the higher the ratio, the better the quality of the output image. It's essentially another way of estimating the similarities between the colourised image and the ground truth [12]. The formula for calculating the PSNR score is as follows:

$$\begin{aligned} PSNR &= 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ MSE &= \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j)-K(i,j)]^2 \end{aligned} \quad [2] \tag{3.5}$$

Where MAX_I represents the maximum pixel value of the image, whereas the MSE is used to measure the difference between the original and target image.

SSIM score

SSIM (structural similarity index measure) is another objective metric used for quantifying the differences or similarity between two images. This metric is meant to serve as an alternative to the PSNR score, and unlike the PSNR score, the SSIM score can capture visible structure of the image [47].

The SSIM index is calculated as:

$$\text{SSIM}(f,g) = l(f,g)c(f,g)s(f,g) \tag{3.6}$$

where

$$\begin{cases} l(f,g) = \frac{2\mu_f\mu_g+C_1}{\mu_f^2+\mu_g^2+C_1} \\ c(f,g) = \frac{2\sigma_f\sigma_g+C_2}{\sigma_f^2+\sigma_g^2+C_2} \\ s(f,g) = \frac{\sigma_{fg}+C_3}{\sigma_f\sigma_g+C_3} \end{cases} \quad [24] \tag{3.7}$$

The first term is the luminance comparison of the two images. The second term is the contrast comparison and the last term is used for structural comparison of the two images. A SSIM of 1 means both images are the same, whereas a high SSIM denotes a high similarity and a low SSIM means there's little to nothing in common between the two images.

Naturalness study

The naturalness study is another form of quantitative evaluation, however, unlike previous objective metrics (PSNR SSIM), this method relies on human opinion. The way the natural study works is by providing a naturalness perception test to volunteers. Each volunteer was given instructions (see figure 3.8 below) and their job is to agree or disagree on whether the colours of the image looked natural or not.

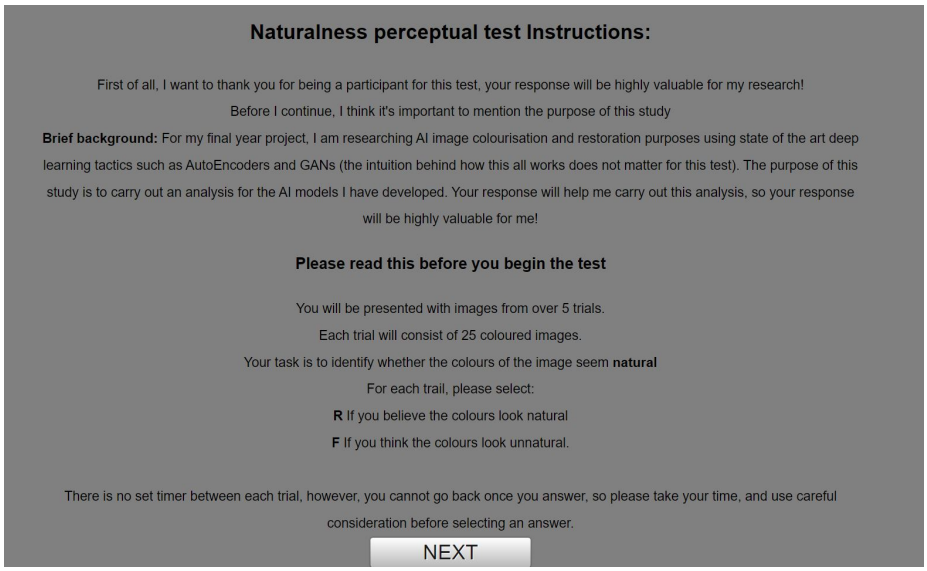

FIGURE 3.9: Naturalness Perceptual test instructions

For example, figure 3.9 is one of the many images presented to the user, they needed to press R or F to answer.

(A) (B) (C)

FIGURE 3.10: Three simple graphs

The study was carried out using a website called *testable.org* which is primarily used for experimenting with people's behaviour and perception using surveys and tests.

This technique was used as calculating accuracy turned out to be unreliable because the calculated accuracy is used to calculate the similarities between two images and does not takes into consideration the naturalness. This could only be achieved by carrying out a study, hence the purpose of this method.

Chapter 4

Results and Discussion

This chapter goes over the results gathered for each method. Accuracy, Loss, SSIM, PSNR and Naturalness study are all used to evaluate each method's performance. Each method is also tested and evaluated on the colourisation of historical photos, and restoration of old coloured photos.

4.1 Training results

After training each model for two weeks, their metrics have been recorded. Below are the results gathered from both AutoEncoders and cGAN.

4.1.1 AutoEncoder training results

Both AutoEncoders were trained for 50 epochs. Accuracy and Loss were recorded during training. The **accuracy** is calculated by totalling up the number of correctly predicted pixels and dividing it by the total number of pixels in the image. The **Loss** is another way of measuring similarity, and the goal is to minimise this loss. Both, training and **validation loss and accuracy** metrics are shown together. The recorded validation metric is used to determine the model's performance.

AutoEncoder	Val accuracy	Val loss	Optimal epoch
Simple AutoEncoder	67%	0.0095	50
Global AutoEncoder	69%	0.00917	45

Table 1. Above shows the training results gathered from two AutoEncoder methods. Out of both models, The Global AutoEncoder has performed the best in terms of accuracy (near 70%) and low loss (0.0091). This suggests that having a global feature extractor can lead to improvements in colourisation.

Simple AutoEncoder Loss/Accuracy graph

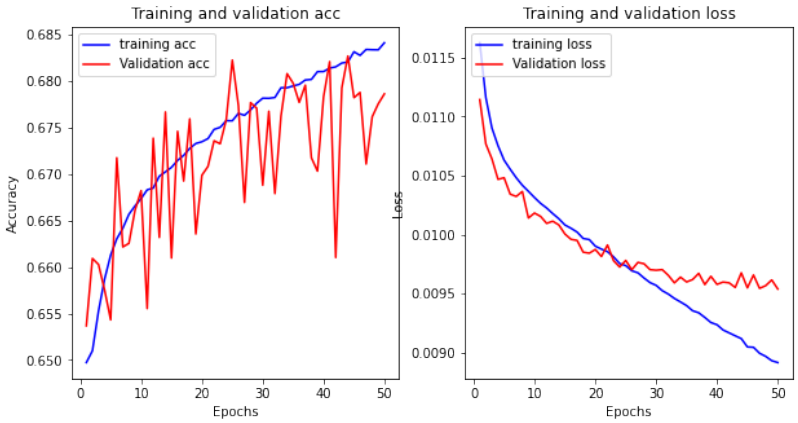

FIGURE 4.1: Simple AutoEncoder training history over 50 epochs

In Figure 4.1, you can see the training progress of the Simple Autoencoder. The model was trained for 50 epochs, which took 24 hours to complete. The left graph demonstrates the **accuracy** for both validation and training sets, while the right graph displays the **loss** values for both sets. Notice how the validation accuracy has a decent ascending trend, although there is some fluctuation. This is likely due to having a small batch size of 32. The best validation accuracy was **67%, achieved at epoch 50**. Whereas the validation loss decreases between 0 and 20 epochs, then levels off and eventually reaches a good **validation loss of 0.0095** at epoch 50.

Global AutoEncoder Loss/Accuracy graph

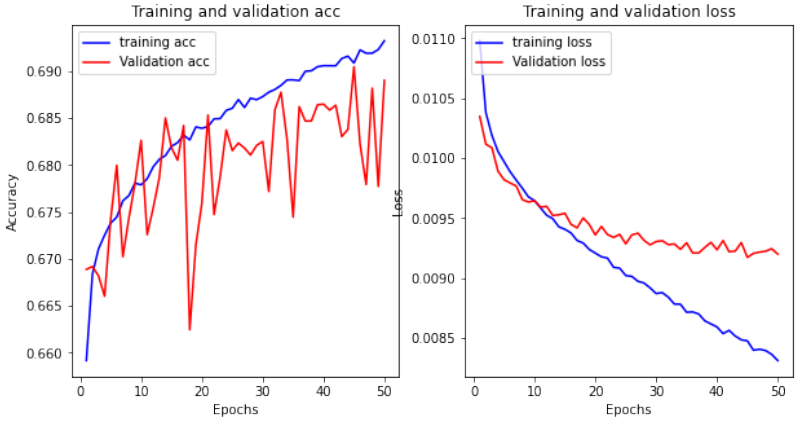

FIGURE 4.2: Global AutoEncoder training history for 50 epochs

Here, in Figure 4.2, you can see the recorded history of the Global AutoEncoder training progress. The model has been trained for 50 epochs. However, this method took more time to complete training (2 days) due to it having more parameters to tune. The left graph depicts an upward trend

in validation accuracy, though fluctuations continue to be an issue. Whereas in the right graph you can see a similar pattern to the last AutoEncoder, but it drops more quickly and levels out at an early epoch of 10. The **best validation accuracy was 69% and loss was 0.0091, which was achieved in epoch 45**.

4.1.2 cGan training results

cGan	Discriminator loss	Generator loss
Pix2Pix	0.0001	16.880
ChromaGAN	-0.0013	0.0002

Table 2. The above shows the results gathered for both cGAN methods (Pix2Pix and ChromaGAN). Unfortunately, it is difficult to compare both cGAN models since both rely on entirely different loss metrics as Pix2Pix uses $L1$ loss whereas ChromaGAN relies on Wasserstein loss for both their generator and discriminator.

Pix2Pix Loss graph

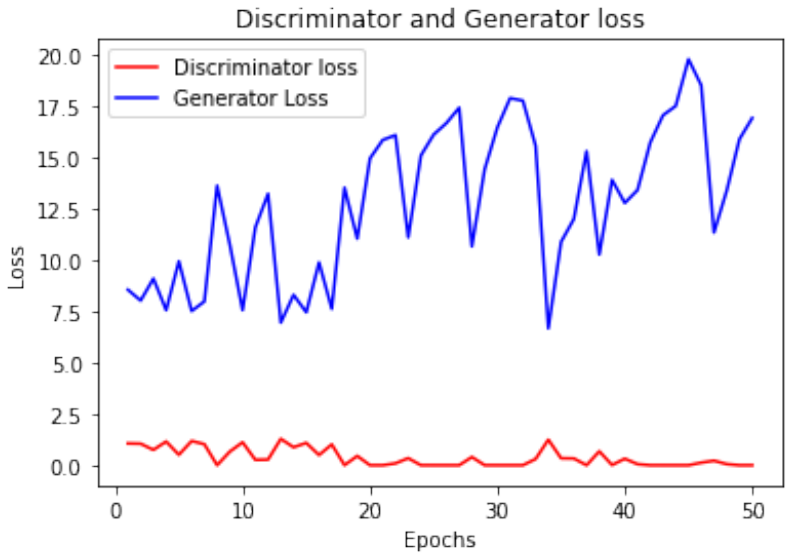

FIGURE 4.3: Pix2Pix training history over 50 epochs

In figure 4.3, the recorded history of the Pix2Pix cGAN training is shown. The cGAN was trained for 50 epochs, which took 4 days to train using Slurm facility. As observed from the graph, generator and discriminator loss have been plotted. It appears as though the generator is failing to achieve its objective against the discriminator. Despite the generator's loss having an increasing trend, at some points, it has achieved a lower loss, for example in epoch 34. If the model were to be trained for more epochs, the generator would eventually start to win against the discriminator. However, this would take more time for training. Fortunately, I have also trained this model on a smaller dataset (7,000 images) over 400 epochs (refer to Appendix B).

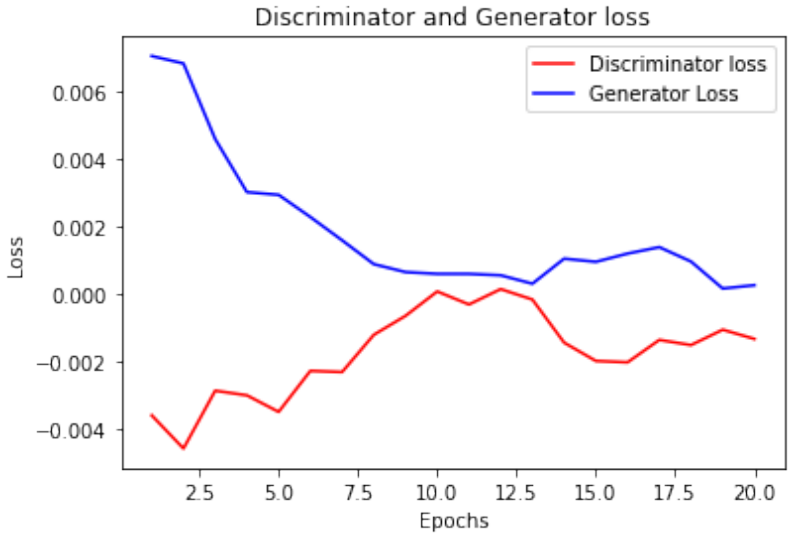

FIGURE 4.4: ChromaGAN training history for 50 epochs

ChromaGAN Loss graph

Figure 4.4 shows the training history for ChromaGAN cGAN. Unlike previous methods, ChromaGAN was only trained for 20 epochs, due to the long training time of 5 days. The generator loss shows a descending trend between the first 8 epochs, then begins to plateau until it ascends slightly between epochs 15 and 17, and then begins to dip again. The discriminator loss mirrors the generator loss. Compared with Pix2Pix, ChromaGAN shows good progress and does not inherit any of the issues seen in the Pix2Pix training history graph, such as the increasing, fluctuating generator loss.

4.2 Image colourisation showcase

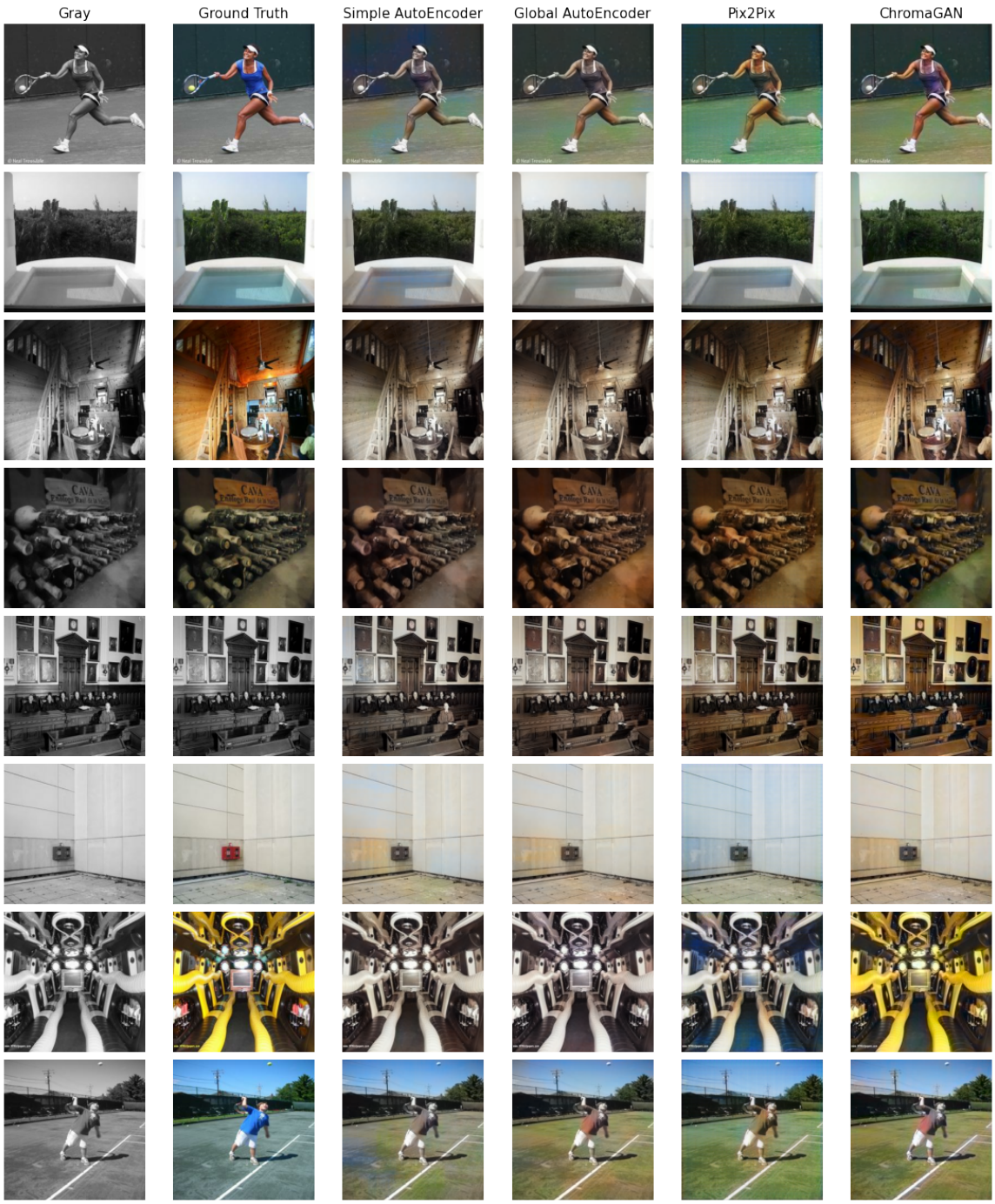

FIGURE 4.5: Eight random images from the test set are colourised. Each row represents an image. The left most column is the grayscale whereas the one next to it is the ground truth. The following three other columns are the three methods used for colourisation.

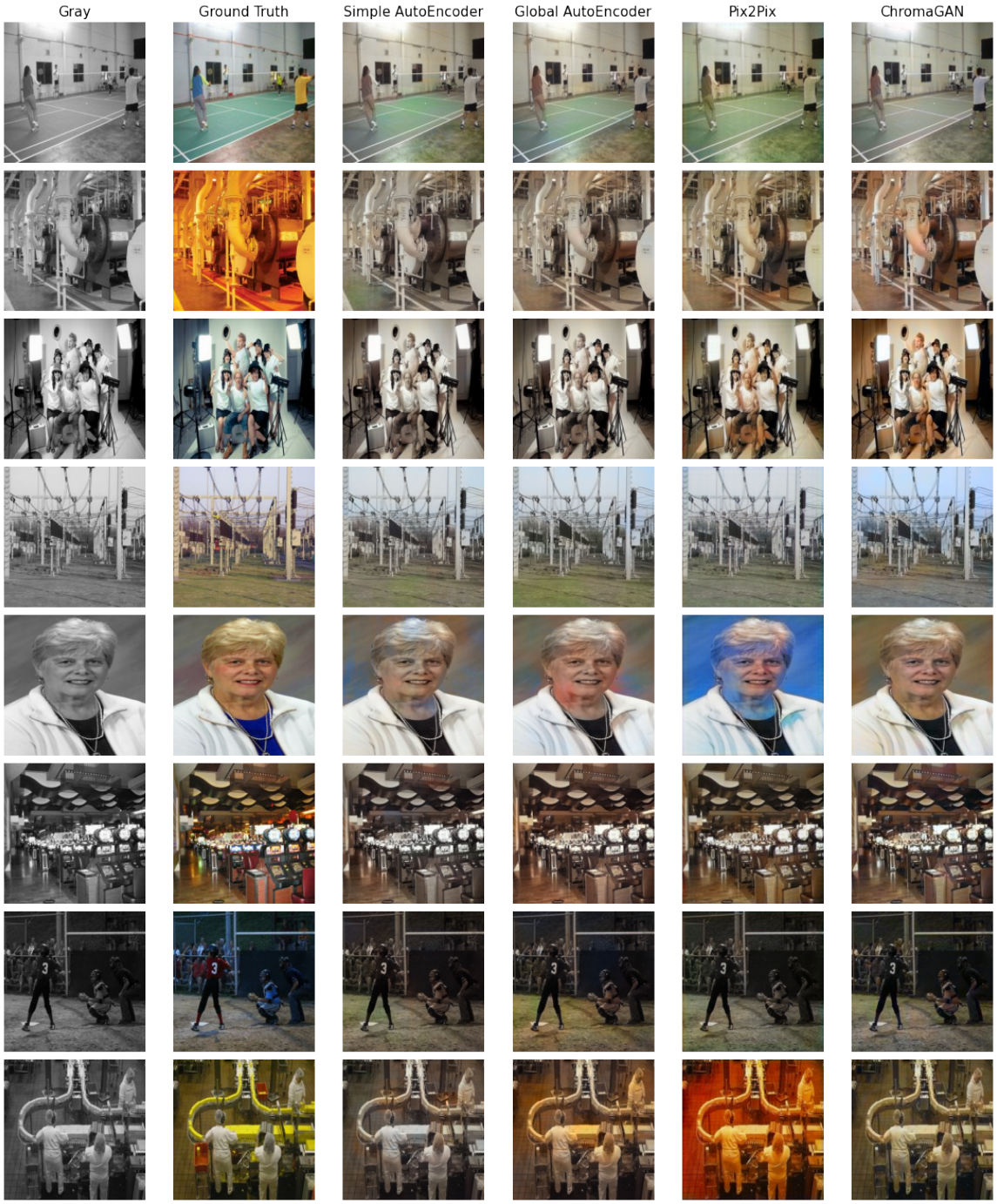

FIGURE 4.6: Further eight more random images are colourised to showcase each method's colourisation ability.

4.3 Evaluation

It was noted in the training results section that accuracy and loss are not reliable metrics to compare between methods since both AutoEncoder and cGAN methods rely on entirely different metrics for evaluation. For this reason, this section will explore evaluation metrics such as **PSNR**, **SSIM** and **natrualness study** which will assist the performance comparison between the methods.

4.3.1 Metric evaluation

Objective score evaluation

Method	PSNR (dB)	SSIM
Simple AutoEncoder	24.36	0.933
Global AutoEncoder	**24.45**	**0.936**
Pix2Pix	23.355	0.906
ChromaGAN	23.94	0.921

The PSNR and SSIM objective metrics were used to evaluate all mentioned architectures using a test set of 21,000 images. Here, you can see a correlation between both metrics. For example, the Global AutoEncoder achieves the best PSNR and SSIM score of 24.45 and 0.936 respectively. Whereas, Pix2Pix comes in last, scoring the lowest in PSNR and SSIM scores. Notice how the AutoEncoder class outperforms the cGAN, this may be due to the nature of the AutoEncoder training, which involves iteratively attempts to make the predictions similar and more inclined to the targets, and both, PSNR and SSIM rely on **similarity**. Whereas the cGAN were trained to make the resulting predictions **realistic**, which disregards the aim of making it similar, hence why the cGAN performed worse off.

Naturalness study evaluation

Method	Naturalness
Real Images	93%
Simple AutoEncoder	45%
Global AutoEncoder	51%
Pix2Pix	53%
ChromaGAN	**74**%

26 participants took part in the naturalness study. The study included real images to provide baseline accuracy. The best performing model was ChromaGAN, which achieved 74% accuracy. Pix2Pix followed with 53% accuracy. Global AutoEncoder came in third with 51%. Simple AutoEncoder had the lowest score of 45%. None of the methods, however, were able to come close to the real image score of 93%. This may be due to errors that easily gave away the naturalness of the image. For example, **visual abstracts** which were seen in the first three methods, also, photos containing humans were turned into zombies as parts of their body remained grey. This was seen throughout the first three models. ChromaGAN managed to perform reasonably well, but the reason it isn't on the same level as real images is likely due to the bland colour palette, which is seen throughout all methods. In contrast to the objective scores, cGAN outperformed the AutoEncoders in this study. This was discussed earlier that the cGAN mainly aims to perform **realistic** colourisation, which is what this study relies on.

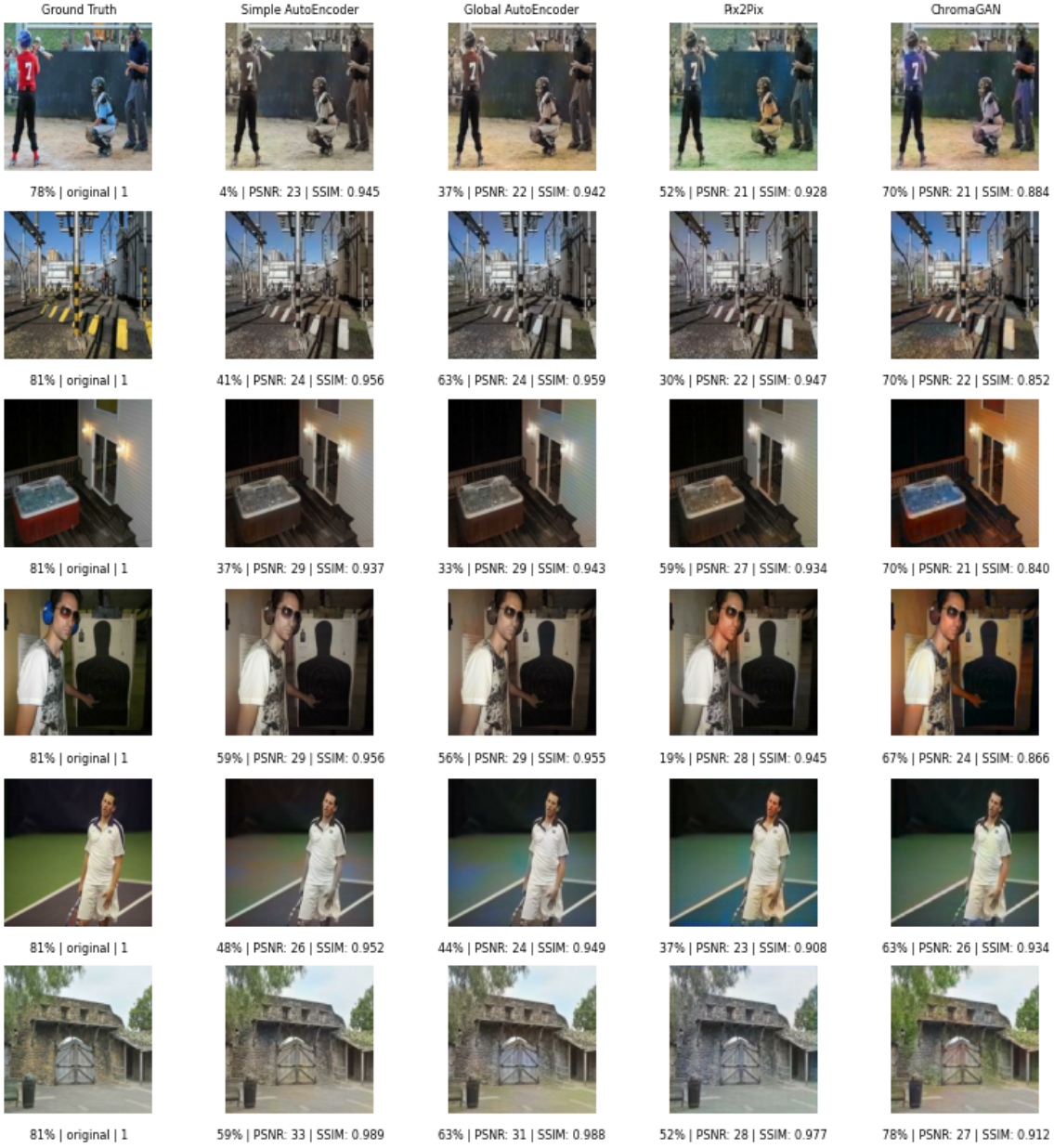

FIGURE 4.7: For each recoloured image per method, I have added a percentage of users who answered "real" or "fake".

Here in Figure 4.5, I've randomly handpicked several images used throughout the naturalness perception study. The left-most column represents the **ground truth** where the following right columns represent the four methods I have discussed so far. Here, the naturalness percentage score alongside their respective PSNR SSIM scores.

The first thing I noticed was how the results help depict the behaviour of the observers. For example, some images contain visual abstracts which makes it obvious for the observers to make this "unnatural". Also, some results seem to show little change in percentage even though the contrast

or luminosity is slightly tweaked. This shows that the observers are tolerant of minor changes.

Additionally, you can notice how the cGAN models (pix2pix & chromaGAN) have performed on average the best in terms of naturalness scores, whereas the two AutoEncoders have scored better in terms of PSNR SSIM. As discussed in the last section, there does seem to be a clear trade-off between the objective metrics and Naturalness scores. It's fair to assume that the cGAN is likely to be the preferred method even if it has a worse average PSNR SSIM score. **In my opinion the scores observed from the human eye (naturalness study) matter more than the scores computed by the computer (PSNR SSIM)**. Out of all methods, it's observed that the ChromaGAN is the best-suited model for image colourisation.

4.3.2 Historical photos analysis

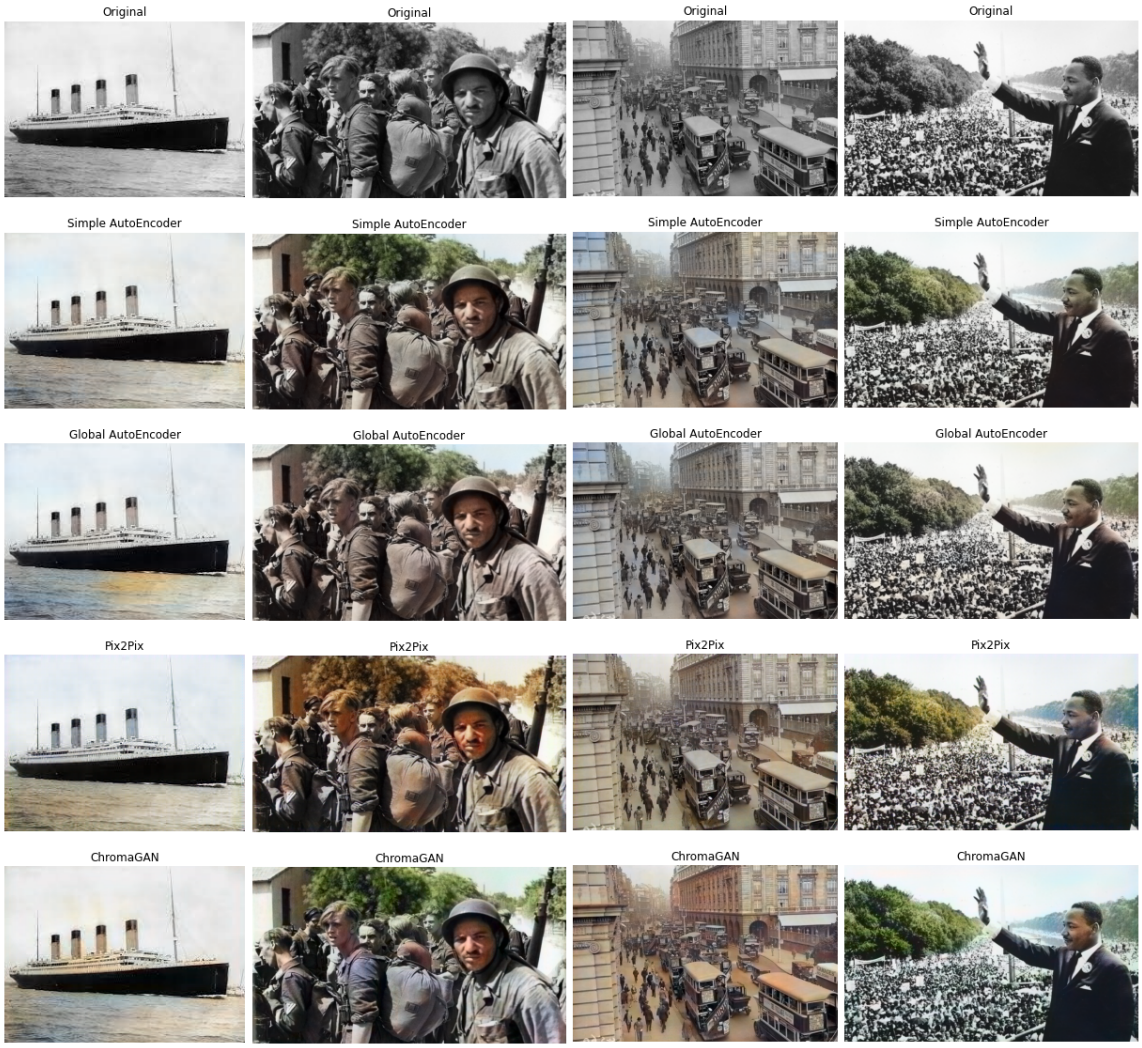

FIGURE 4.8: A few historical photos have been picked. Titanic (1912), French Algerian soldier Gaurding captured German soldiers (1944) and London (1920s), Martin Luther King giving a speech (1963).

I've tested each model on historical photos. Figure 4.8 shows the colourisation results. Since there are no ground truths to identify how accurate each model is, it is down to personal judgement. ChromaGAN seems to show the best overall colourisation results as it does a better job colourising people and their surroundings.

4.3.3 Colour restoration analysis

FIGURE 4.9: Restoring the colours of a few photos whose colours have faded

I've handpicked three photos that have suffered deterioration in quality due to environmental factors such as poor handling, poor material or ageing. The photos were used to test each model's capabilities for the purpose of restoring colours.

Out of all methods, the ChromaGAN model has done a decent job in bringing back the vibrant colours which lacked in the original photos. It even removed the orange hue seen within the first two original images.

However, not all methods are accurate, for example in the third row, Pix2Pix coloured the curtains blue instead of yellow.

It is also important to note that the models cannot remove the fade effect or any other visual abstracts seen in the original image as none of the models were trained to deal with this type of issue, hence why its presence still exists in the resulting images.

4.4 Limitations

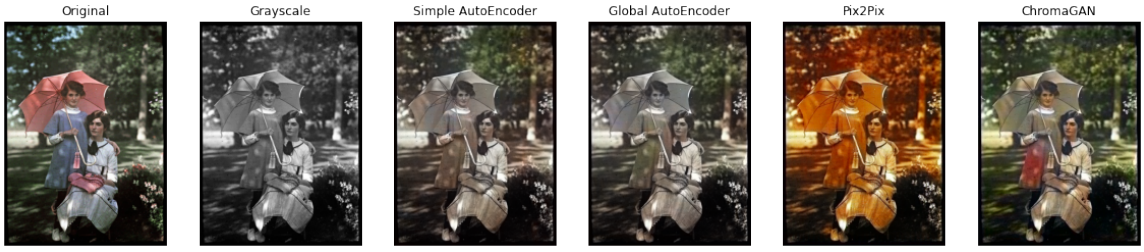

FIGURE 4.10: Limitation example 1

Here, it is important to outline a few key limitations observed from all methods. For example, figure 4.10 demonstrates some form of the limitations suffered by each model:

1. All methods were trained to colourise images of size 256x256 (except ChromaGAN - 242x242), this means that test images require resizing. Resizing causes distortion which can negatively impact the colourisation process. This is seen in the first three methods.
2. The colourisation process is biased and will not apply accurate colours to everything. For instance, it failed to colourise the umbrella or the handbag pink. This is because the algorithms were only trained on what they had seen and likely never trained on images consisting of pink umbrellas and pink handbags.
3. The predicted colours are also bland, especially complex small objects. This again relates to the last point of bias: The algorithm has a lot to learn before it can learn to colourise more entities using accurate colours.
4. Incorrect colour placement and visual abstracts are very apparent and seen throughout all models. For instance, the colourisation of Pix2Pix is completely all orange, whereas ChromaGAN has produced a slight reddish visual abstract which can be seen on the dress of the left person.

Despite mentioning all these limitations, some of the methods such as ChromaGAN manage to do a decent enough job to make the colourisation look convincing, however, still suffer from the limitations mentioned above.

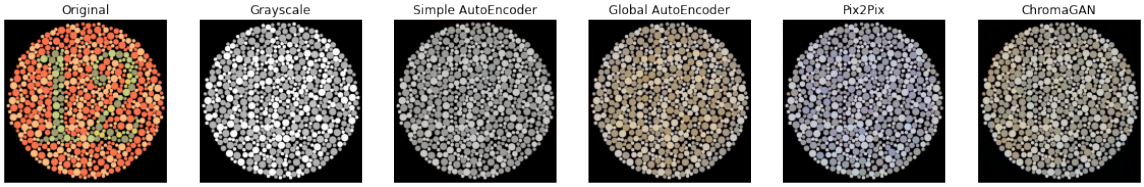

FIGURE 4.11: Limitation example 2

Furthermore, the methods are not suitable for colourising synthetic images as demonstrated in figure 4.11. For instance, the colour blind test contains a "12" which is coloured in green. However, as you can observe, the grayscale image completely loses all the information of all the synthetic colours, and if we, the viewers can not see a "12" from a grayscale image then neither can the models.

4.5 Discussion

Let's revisit the aims established at the beginning of this investigation:

1. Determine the best image colourisation method by carrying out an analysis of deep learning architectures such as AutoEncoders and Conditional Adversarial Networks using quantitative measures such as human assessment and objective metrics.
2. Investigate whether the objective measures correlate with human assessment results and determine which measures matter more for the purpose of image colourisation.

Now I believe the first aim has been addressed since judging from the results discussed earlier, I've successfully carried out an analysis of AutoEncoders and cGAN using objective measures (PSNR SSIM) and human measures (natural perception test). Here, judging from the results, it appears that the cGAN methods provide more realistic results, offering diverse colourisation than AutoEncoders, while AutoEncoders generate results that better match the targets but with colours often seen as less realistic. Here, it was revealed that the human assessment tends to rely more on realistic colourisation, whilst the objective measures relied more on similarity. And it was seen that the cGAN performed better on realism, whereas AutoEncoders were better are similarity.

So there's a clear trade-off between realism and similarity, which raises the second aim established earlier. It's clear that the objective measures do not correlate with human opinion, and this may be because the objective measures only focus on the structural and noise differences, not the colour, which the human volunteers focused on, which makes the objective metrics unreliable in conducting an evaluation for image colourisation. Unfortunately, there does not seem to be any alternative metrics, hence using human-driven study was needed.

Also, according to this study, ChromaGAN performed consistently the best as it managed to achieve the highest average naturalness study score (73%) and a reasonably good average PSNR/SSIM score, making ChromaGAN the best-suited algorithm for image colourisation.

Chapter 5

Conclusions

The purpose of the research was to study and analyse different deep learning methods, specifically AutoEncoders and cGANs. The AutoEncoders used were Simple and Global AutoEncoder, both based on the Authors of [16]. Pix2Pix and ChromaGAN were chosen to investigate the performance of cGAN and used to compare against the AutoEncoder methods explored earlier.

The Places365 dataset was used to train all methods. The dataset consisted of 105,000 samples split into a 60:20:20 ratio. A TensorFlow pipeline was used to help work with the large dataset while having limited memory space. I chose to train the models in Slurm Facility, a service provided by the university that is practically free.

After a couple of days of training, the methods were tested on the reserved test set and evaluated using objective metrics and Naturalness study. Results have shown that AutoEncoders tend to perform better than cGAN methods using objective metrics, but cGAN performed better on Naturalness Study.

This raised questions about which evaluation metric matters more. It was noted known that using objective metrics were not reliable when dealing with coloured images as it would only focus on structural and noise difference, not the colour. Hence, the only useful way of evaluating the methods was through human opinion via naturalness study. This study helped conduct a more fair comparison between all methods. The results of the study showed that cGAN was the best deep learning method out of the two compared.

5.1 Future Work

5.1.1 More Data

One of the limitations seen throughout this project was the time constraint which forced me to limit the dataset to a rather small size of 80,000. And even then, training took a considerable amount of time, taking at most 5 days for training one method (ChromaGAN) for only 20 epochs. Even though **80,000** may sound a lot in other deep learning tasks, it isn't sufficient enough for the task of Image Colourisation. Having more data will enable the model to achieve a far richer hyperspace to leverage from, which may achieve far more impressive colourisation results.

5.1.2 NoGAN training

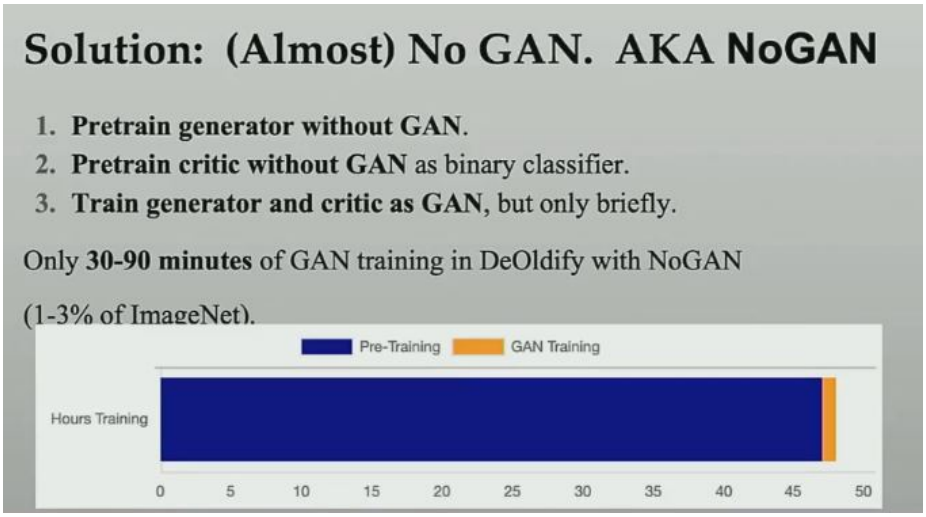

FIGURE 5.1: NoGan training process [3]

NoGAN training is another technique used by the authors of Deoldify [3]. The process is rather straightforward, you first train the generator model by itself and then generate images from that. You then proceed to use those generated images alongside real images and train the discriminator in a binary classification manner. Finally, you briefly train both the generator and the discriminator together in a GAN setting. The effects of this training method are noted to have reduced the visual abstracts seen prior to this method and have overall improved the colourisation quality.

5.1.3 User guided and Exemplar methods

FIGURE 5.2: Scribble-based colourisation method which relays on user input [50]

Lastly, I would like to experiment with models that involve user guidance. For example, the author's of Scribbler rely on user guidance. The basic process involves scribbling (hence the name) colours onto a chromatic photo. The algorithm will then distribute the colours across regions of the photo. These regions can be anything, such as T-shirts, furniture or landscapes.

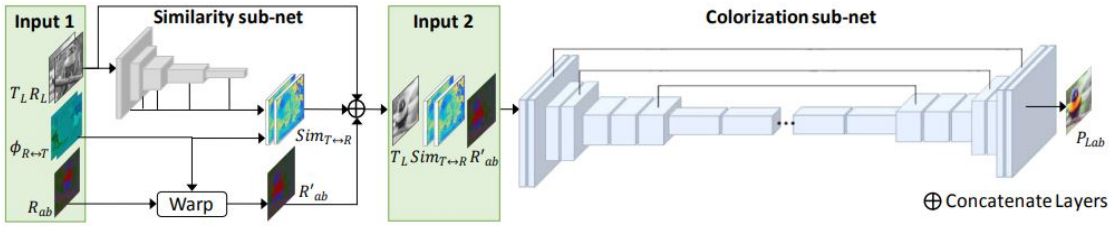

FIGURE 5.3: Exempler-based architecture for image colourisation [23]

In addition to this, I would like to explore ensemble methods that involve combining many models to seek better predictive performance. For example, the author of Deep Exemplar-based Colorization use this technique [23].

5.2 Self Evaluation

This research project has allowed me to improve my skills and deepen my interest in the topic of deep learning within the field of Artificial Intelligence. Before beginning this research, I had very little knowledge of AI image colorization or AI in general. The AI and Machine learning modules I took for my third year provided me with the basic foundations I needed to pursue this project. Of course, there were some difficulties along the way: I lacked some key concepts that made understanding certain aspects difficult. However, this was solved after I learnt some of the essential terminologies and concepts. Overall, this project has been an amazing opportunity to learn many new skills and knowledge.

A Appendix

Code repository

If you wish to reproduce the results produced from this investigation, I have a GitHub repository which can be found at the following link:

https://github.com/mrehmm001/Deep-Image-Colourisation-comparing-AutoEncoders-and-Conditional-Adversarial-Networks

A dedicated Nvidia GPU is required to enabled GPU accelaration in order to reduce training time. Also, the dataset I've used was from the places365 website which can be found at the followng URL:

http://places2.csail.mit.edu/download.html

Here, I've used "Places-Extra69" version which contains fewer images of 105,000 total. I've partitioned my owned dataset, and if you wish to use mine, here is the URL link:

https://drive.google.com/file/d/178mTebcMLPT5CixUzJOzoDYDLFpVIyNb/view?usp=sharing

I've used author's code to help conduct this research [17], [45], [27].

My research

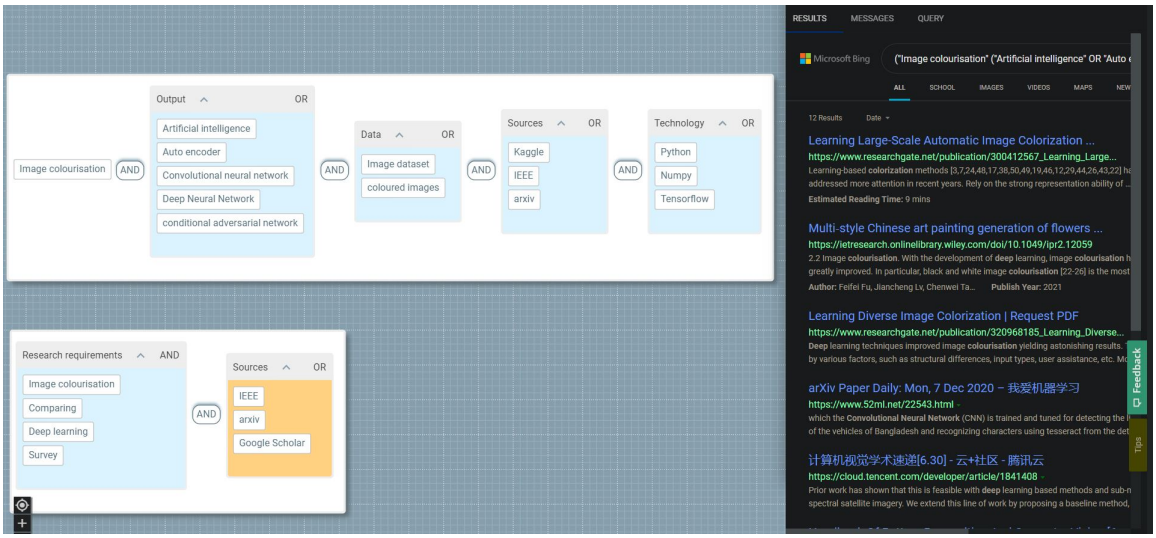

FIGURE 5.4: 2dsearch was used to help conduct my research

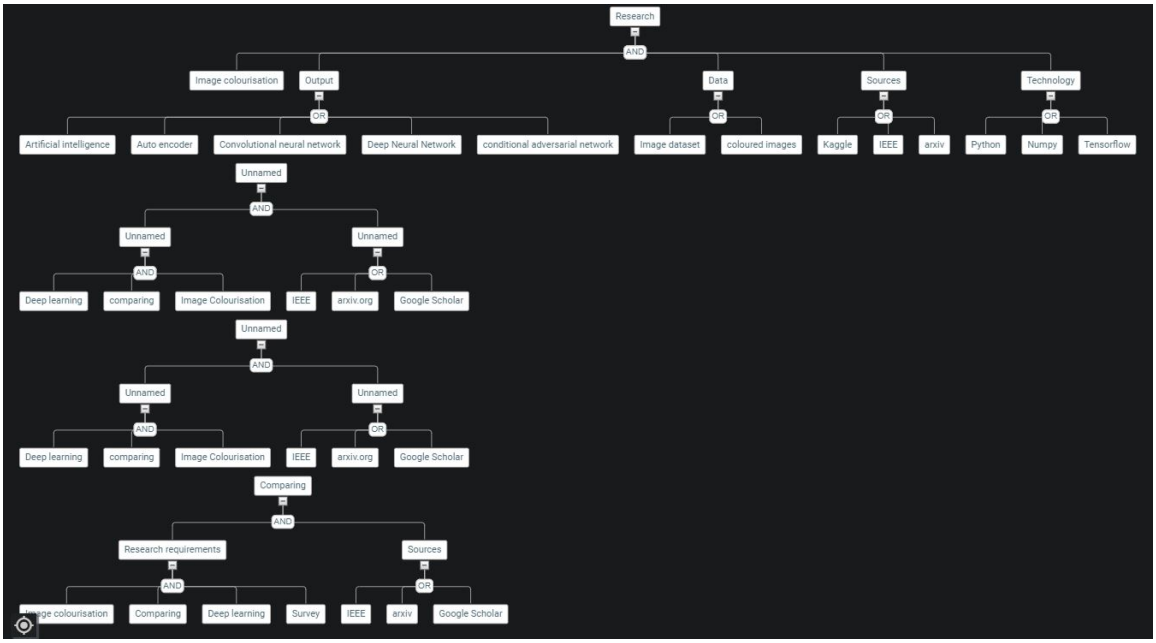

FIGURE 5.5: Literature Research tree diagram

B Appendix

Neural Networks

The Artificial Neural Network (ANN) is a structure that is inspired by the human biological brain, it consists of several processing units (neurons) that have a natural propensity for storing experiential knowledge and making it available for use. As S. Haykin puts it, an ANN resembles the human brain in two respect [22]:

1. Knowledge is acquired by the network from its endowment through a learning process.
2. interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

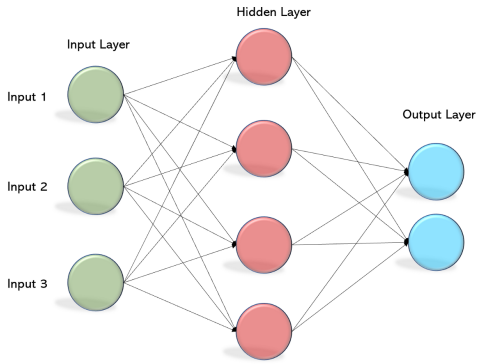

FIGURE 5.6: Example of a Artificial Neural Network [37]

The learning operation that is used to carry out the action is known as the learning algorithm. The learning algorithm is used during the training phase to modify the synaptic weights of the network. The modification of the synaptic weights will influence the behaviour of the network, which will determine the output of the network.

During the training phase, we have training sets that contain input data with corresponding targets. The input data is fed into the network, and the network performs a process called feedforward. Feedforward involves the summation of synaptic weights and corresponding inputs with the application of activation functions. This results in a predicted output. The predicted output is compared against the actual target output and an error is calculated. This error is used to determine how accurate the network was. If the predicted output is wrong, the network uses a learning algorithm known as the backpropagation algorithm. The backpropagation algorithm computes the gradient descent by calculating the derivative and updating the synaptic weights. This process repeats until the learning algorithm converges into what is known as a global minimum. At the

global minimum, learning will cease to proceed. This means the network can classify any input and map it to their corresponding predictions that match the targets.

Convolutional Neural Networks

Convolutional Neural Network (CNN) (also known as convnets) are a special class of ANN used in deep learning and are universally used for computer vision purposes [42]. CNN is similar to dense neural networks in ways that CNN consists of input and output layers, as well as several hidden layers, however, the difference lies in the layer: The layer of the CNN consists of a special type of layer called "convolutional layer" (hence the name CNN derives from) that are used to filter out features from the previous layer, called a feature map. The layers also consists of pooling layers which function to reduce the spacial representation size to reduce the number of parameters of the network. The effects of these layers enable CNN to extract complex features from data and learn the representations using fewer parameters, making CNN more efficient at the task of computer vision compared to traditional dense ANN.

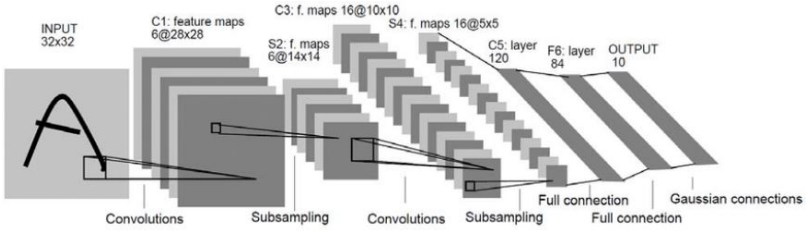

FIGURE 5.7: handwriting recognition using CNN [7]

For further information on how Neural Networks word, refer to Haykin's Neural Network textbook [22] amd deep learning with python [13].

Tuning results

The table shown below demonstrates the AutoEncoder tuning results but on a much smaller dataset (Landscapes by blackmamba [36]) which contains in total 8,000 images split 60:20:20 ratio. The results are taken from my interim report [40].

Experiments	Optimal epoch	Val loss	Val accuracy	Batch norm	Augmentation	Learning rate
Experiment 1	**27**	**0.0096**	**0.6548**	**No**	**No**	**Default (1e-3)**
Experiment 2	**14**	**0.0075**	**0.7155**	**No**	**No**	**1e-4**
Experiment 3	**21**	**0.0075**	**0.7065**	**No**	**No**	**1e-5**
Experiment 4	**85**	**0.0073**	**0.7106**	**Yes**	**Yes**	**1e-4**
Experiment 5	**62**	**0.0071**	**0.7177**	**No**	**Yes**	**1e-4**

TABLE 5.1: Experiments gathered throughout the training of the model

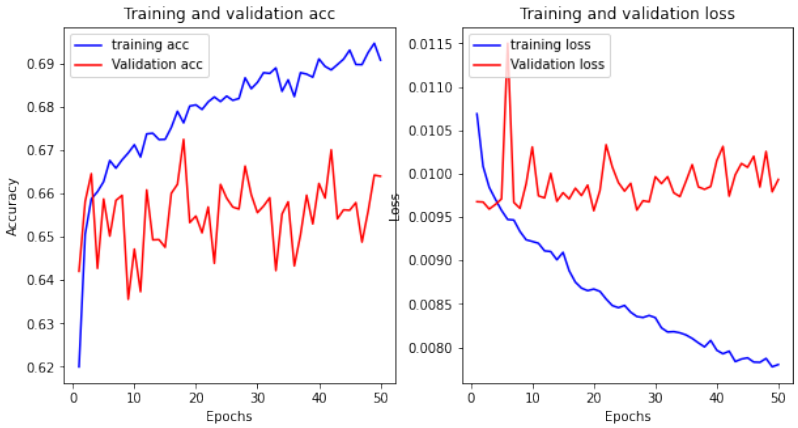

FIGURE 5.8: Experiment 1 plot

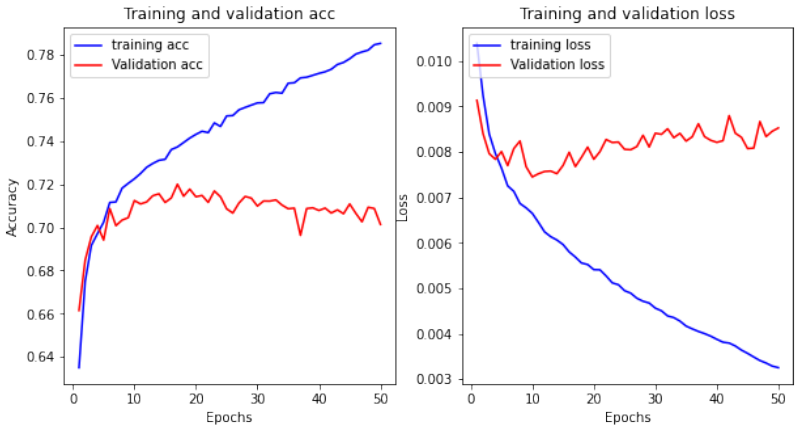

FIGURE 5.9: Experiment 2 plot

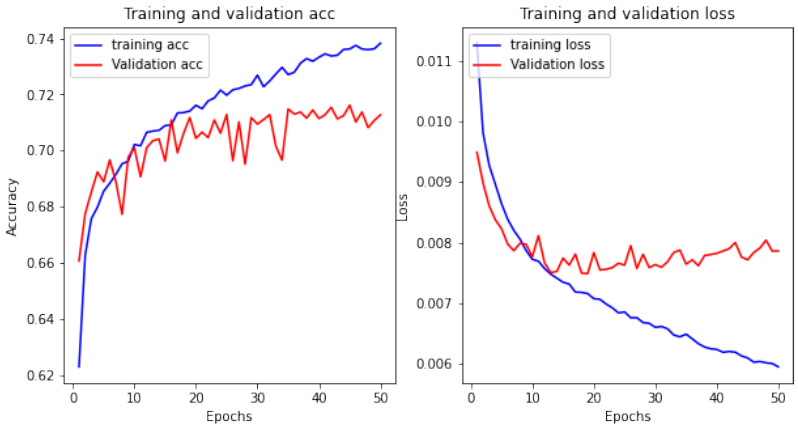

FIGURE 5.10: Experiment 3 plot

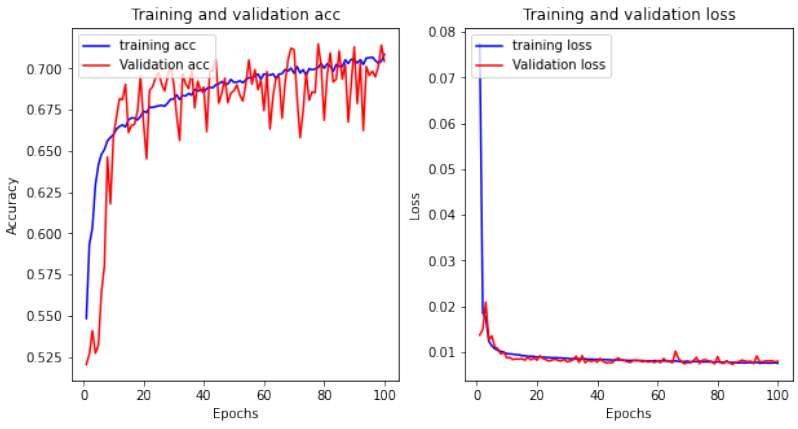

FIGURE 5.11: Experiment 4 plot

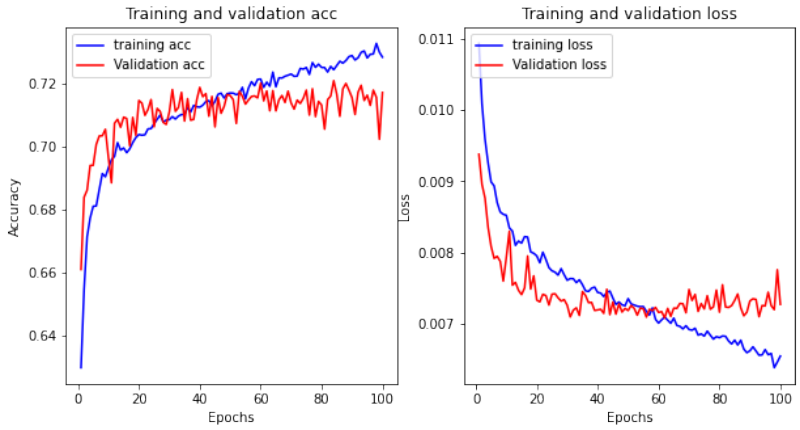

FIGURE 5.12: Experiment 5 plot

Optimal model analysis

FIGURE 5.13: Validation and accuracy plots of the optimal model

Training the Pix2Pix using a smaller dataset

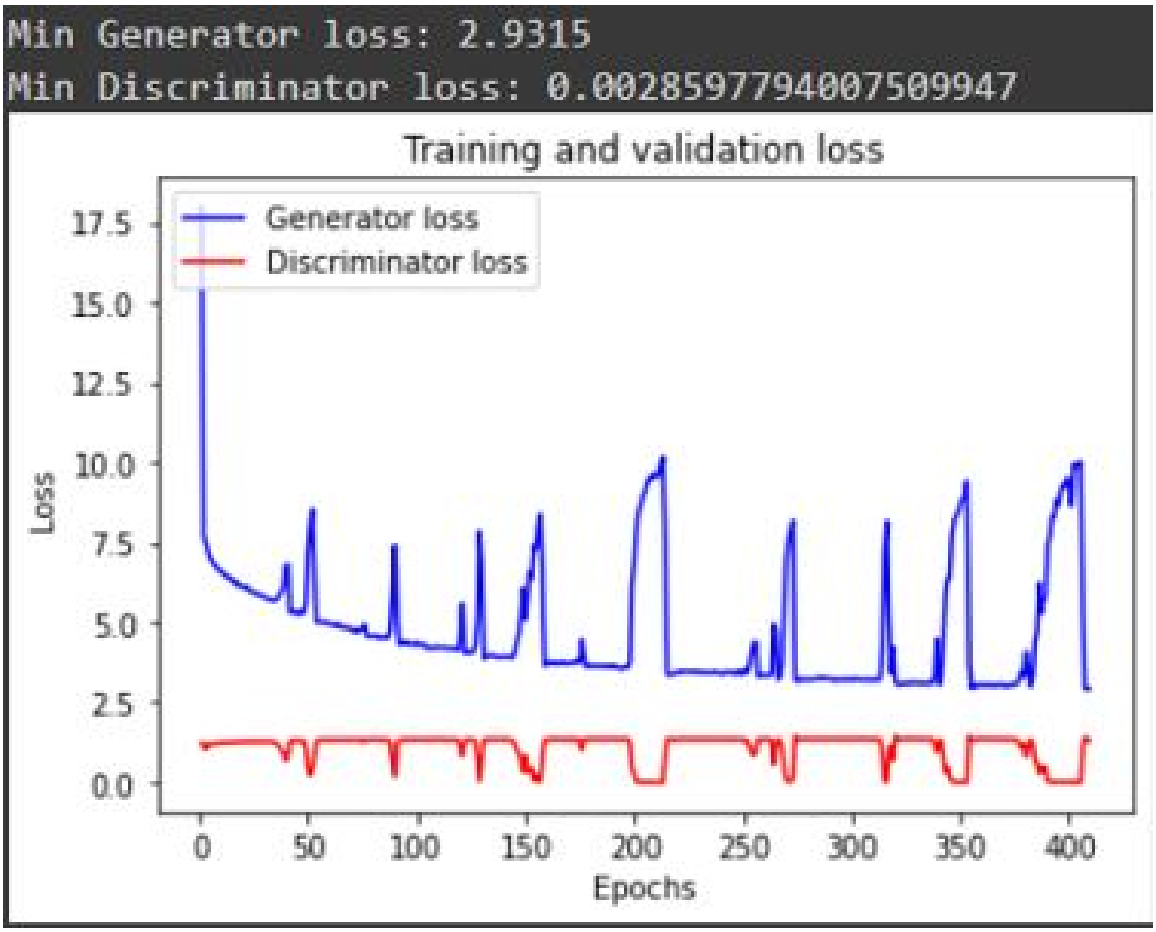

FIGURE 5.14: Using a smaller dataset (landscape by black mamba [36]), the Pix2Pix was trained over 400 epochs over the course of 24 hours.

User study results

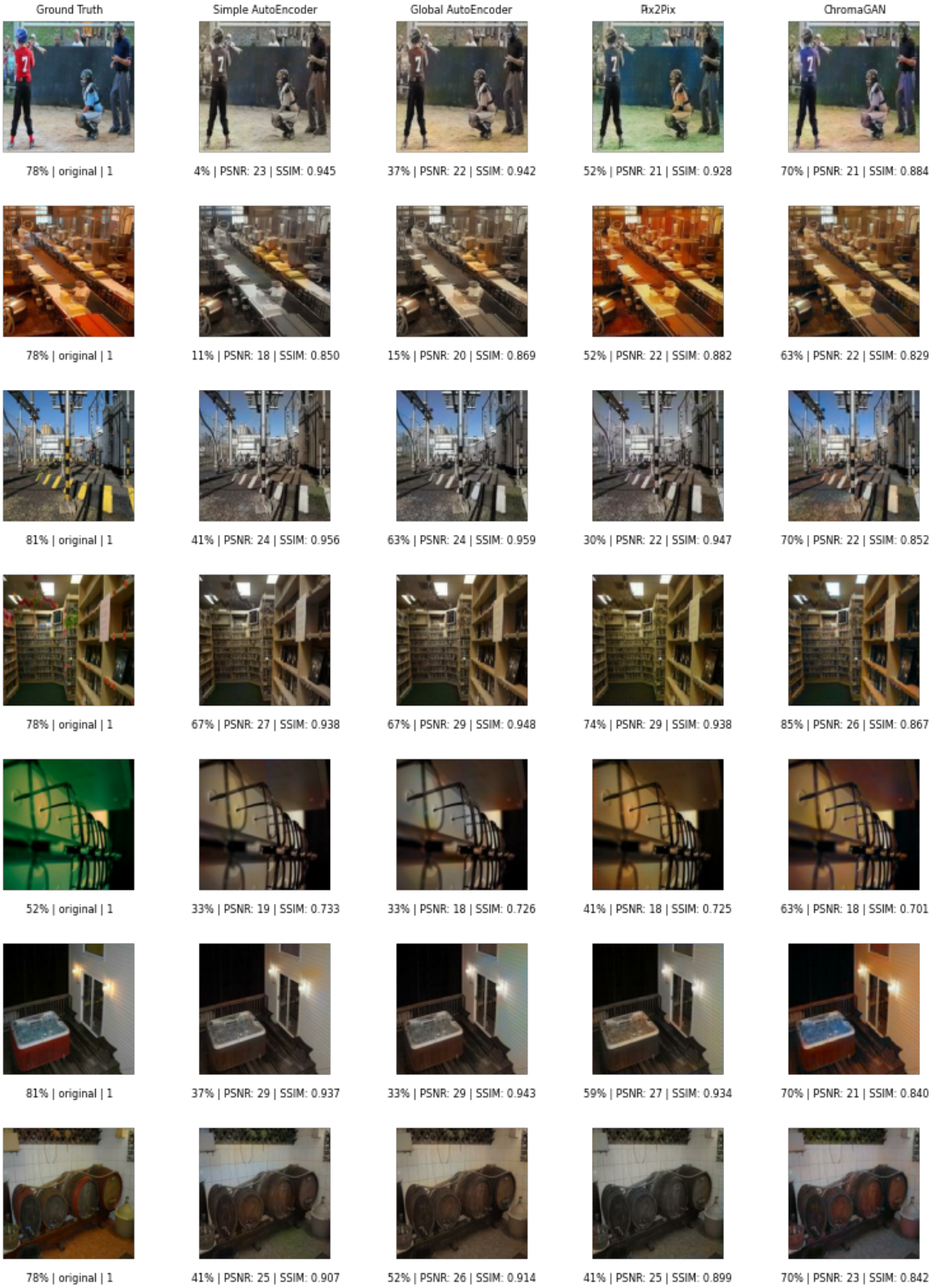

FIGURE 5.15: Full user naturalness study results

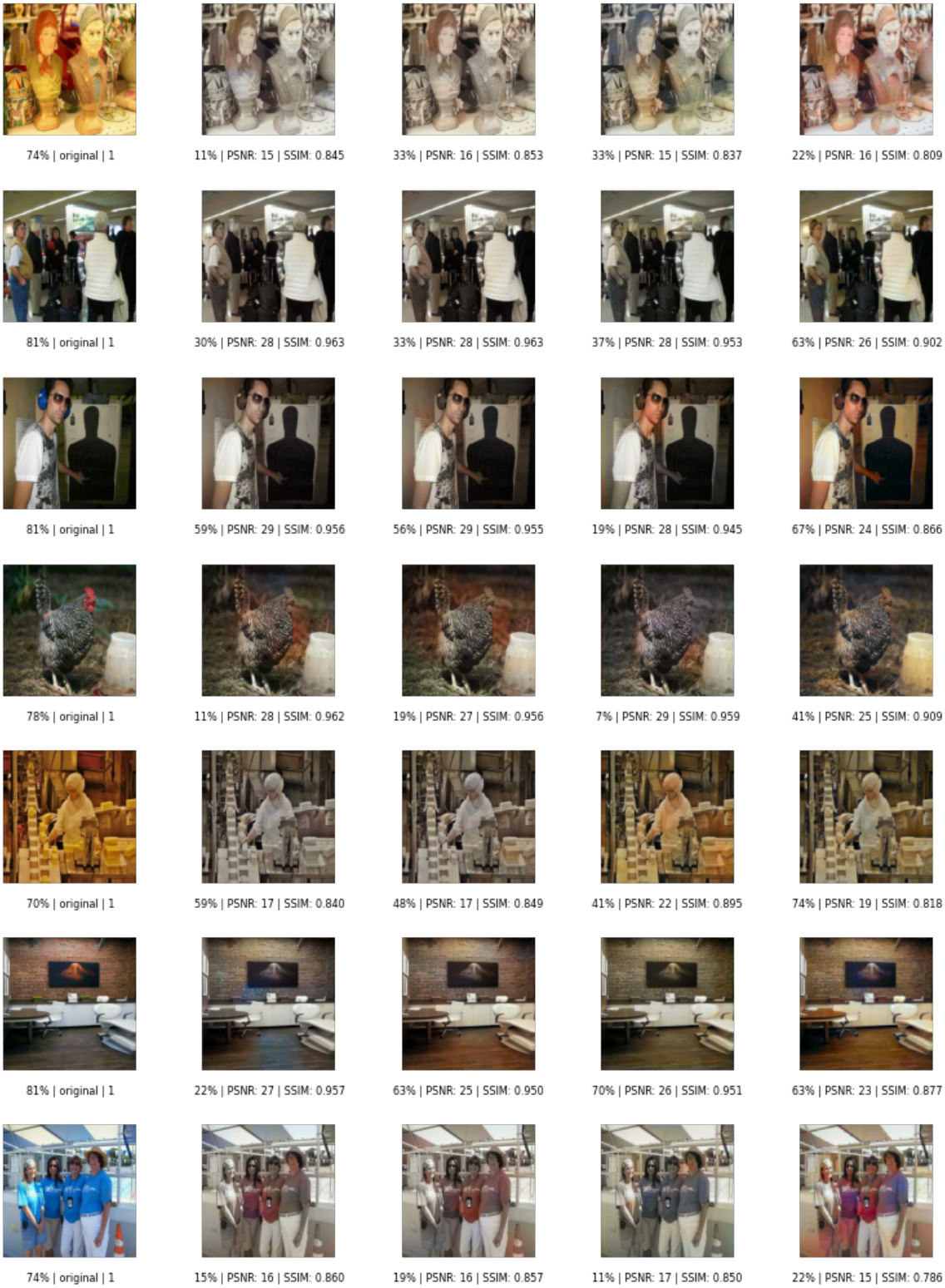

FIGURE 5.16: Full user naturalness study results

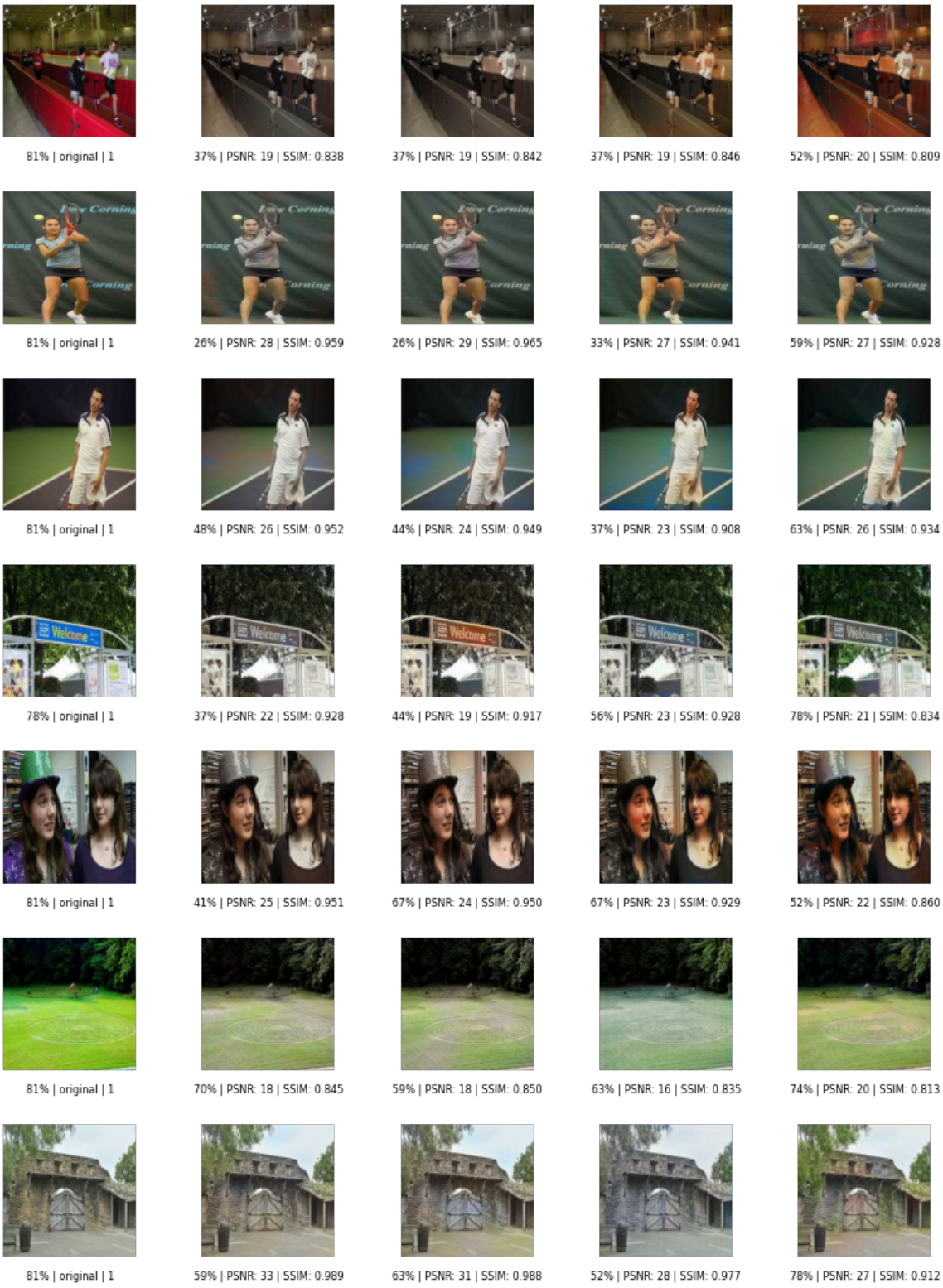

FIGURE 5.17: Full user naturalness study results

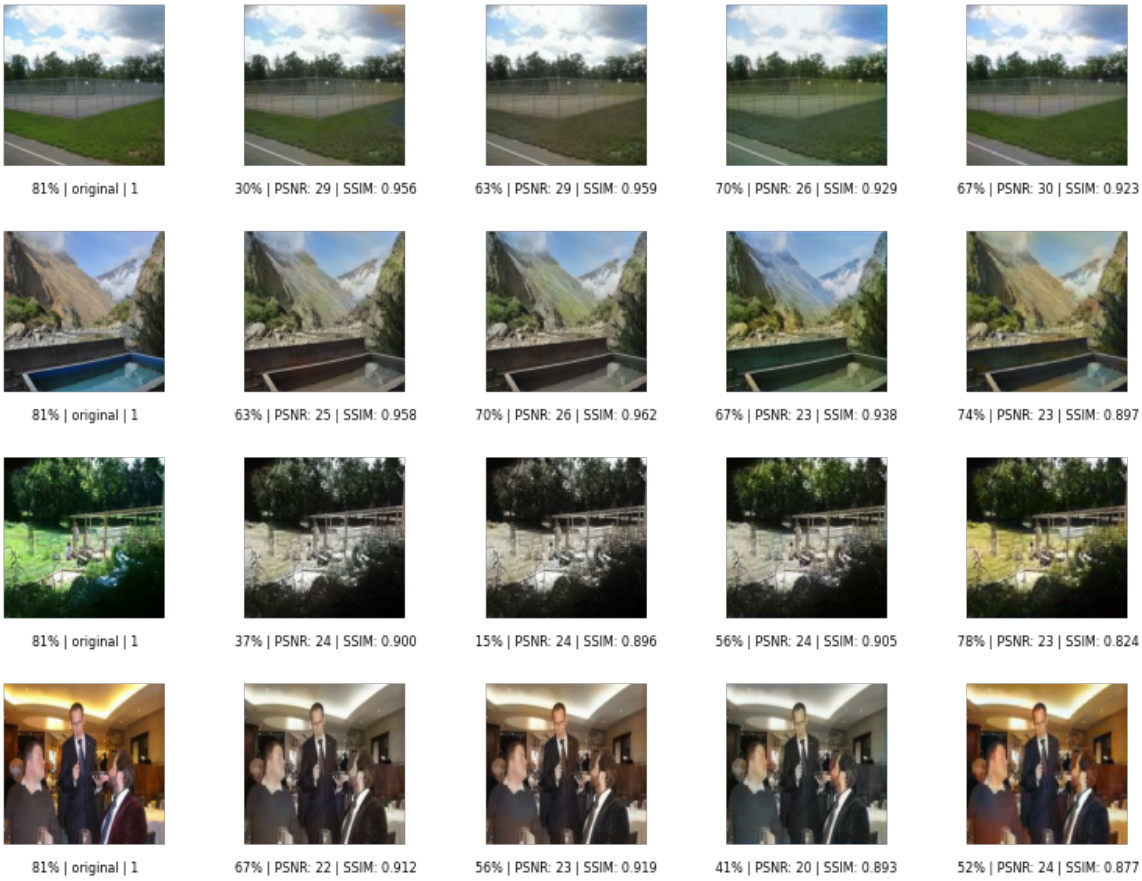

FIGURE 5.18: Full user naturalness study results

C Appendix

Video colourisation results

I have carried out colourisation of historical footages to see how they'd perform.

FIGURE 5.19: WW2 combat footage colourised using Simple AutoEncoder. Link: https://youtu.be/OW9Sdq4ejSo

FIGURE 5.20: WW2 combat footage colourised using Global AutoEncoder. Link: https://youtu.be/J4YZ3RDdiig

FIGURE 5.21: WW2 combat footage colourised using Pix2Pix. Link: https://youtu.be/fuUdiurJ$_qI$

FIGURE 5.22: WW2 combat footage colourised using ChromaGAN. Link: https://youtu.be/jiF394SHxI4

FIGURE 5.23: Black and white footage of new york ChromaGAN. Link: https://youtu.be/nqlh5KMAf3I | original footage link: https://www.youtube.com/watch?v=f-d88HkWFtQ

Bibliography

[1] jantic/deoldify: A deep learning based project for colorizing and restoring old images (and video!).

[2] anon anon. How to calculate psnr (peak signal to noise ratio) in matlab?, Feb 2014.

[3] Jason Antic, Jeremy Howard, and Uri Manor. Decrappification, deoldification, and super resolution · fast.ai, 2019.

[4] Saeed Anwar, Muhammad Tahir, Chongyi Li, Ajmal Mian, Fahad Shahbaz Khan, and Abdul Wahab Muzaffar. Image colorization: A survey and dataset. *arXiv preprint arXiv:2008.10774*, 2020.

[5] Dongyang Ao. Architecture of the patchgan discriminator network ..., Oct 2018.

[6] Mohammad Ali Bagheri. A tutorial on conditional generative adversarial nets + keras implementation, Jun 2019.

[7] Lucijano Berus, Dec 2018.

[8] Marc Blanch, Marta Mrak, Alan Smeaton, and Noel O'Connor. End-to-end conditional gan-based architectures for image colourisation. pages 1–6, 09 2019.

[9] Jason Brownlee. A gentle introduction to generative adversarial networks (gans).

[10] Jason Brownlee. A gentle introduction to the rectified linear unit (relu), Aug 2020.

[11] Jason Brownlee. Gentle introduction to the adam optimization algorithm for deep learning, Jan 2021.

[12] Kalpana Chauhan, Rajeev K. Chauhan, and Anju Saini. Enhancement and despeckling of echocardiographic images. *Soft Computing Based Medical Image Analysis*, pages 61–79, 1 2018.

[13] François Chollet. *Deep Learning with Python*. Manning, November 2017.

[14] Mahmoud El-Jiddawi. Auto colorization of black and white images using machine learning "auto-encoders" technique | by mahmoud el-jiddawi | becoming human: Artificial intelligence magazine, 2019.

[15] Isola et al. Papers with code - patchgan explained, Nov 2018.

[16] Lucas Rodes-Guirao Federico Baldassarre, Diego Gonzalez-Morin. Deep-koalarization: Image colorization using cnns and inception-resnet-v2. *ArXiv:1712.03400*, December 2017.

[17] Lucas Rodes-Guirao Federico Baldassarre, Diego Gonzalez-Morin. Deep-koalarization: Image colorization using cnns and inception-resnet-v2 github. *ArXiv:1712.03400*, December 2017.

[18] Steven Flores. Variational autoencoders are beautiful, Apr 2019.

[19] Daniel Frumkin, Melanie Manipula, Krisorn Kachai, Axel Fehr, and Dawson Sewell. Conditional generative adversarial network (cgan) - wiki.

[20] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[21] Aurlien Gron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition, 2017.

[22] Simon S. Haykin. *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, third edition, 2009.

[23] Mingming He, Dongdong Chen, Jing Liao, Pedro V. Sander, and Lu Yuan. Deep exemplar-based colorization. *CoRR*, abs/1807.06587, 2018.

[24] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

[25] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, 35(4):110:1–110:11, 2016.

[26] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[27] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks github. *CVPR*, 2017.

[28] Itseez. Open source computer vision library. `https://github.com/itseez/opencv`, 2015.

[29] Jeremy Jordan. Introduction to autoencoders., 2018.

[30] Frank Karstens. What is the rgb color space?

[31] Daniel Kronovet. Objective functions in machine learning, Mar 2017.

[32] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, and et al. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, Mar 2020.

[33] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23:689–694, 2004.

[34] Bo Li, Yu-Kun Lai, and Paul L Rosin. Chapter 1 a review of image colourisation. 2019.

[35] Ming Ronnier Luo. *CIELAB*, pages 1–7. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[36] Black Mamba. Landscape color and grayscale images | kaggle, 2021.

[37] Awhan Mohanty. Multi layer perceptron (mlp) models on real world banking data, May 2019.

[38] Seán Mullery and Paul F. Whelan. Human vs objective evaluation of colourisation performance, 2022.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] Muneeb Rehman. Muneeb rehman - image colourisation using convolution neural network (interim report). 2022.

[41] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

[42] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way | by sumit saha | towards data science, 2018.

[43] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color, 2016.

[44] Patricia Vitoria, Lara Raad, and Coloma Ballester. Chromagan: An adversarial approach for picture colorization. *CoRR*, abs/1907.09837, 2019.

[45] Patricia Vitoria, Lara Raad, and Coloma Ballester. Chromagan: Adversarial picture colorization with semantic class distribution github. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2445–2454, 2020.

[46] Emil Wallner. How to colorize black amp; white photos with just 100 lines of neural network code, Feb 2021.

[47] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[48] M. Ware. *MECHANISM OF IMAGE DETERIORATION IN EARLY PHOTOGRAPHS*. SCIENCE MUSEUM, 1994.

[49] Jianxiong Xiao, James Hays, Krista A Ehinger Aude, and Oliva Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo.

[50] Kevin Yu Xinyang Geng, Angela Lin. Deep image colorization with user guidance. 2019.

[51] Andy B. Yoo, Morris A. Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In Dror Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, pages 44–60, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[52] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Antonio Torralba, and Aude Oliva. Places: An image database for deep scene understanding. *CoRR*, abs/1610.02055, 2016.

[53] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Ieee transactions on pattern analysis and machine intelligence places: A 10 million image database for scene recognition. 1109.

[54] Jianlong Zhou, Amir H. Gandomi, Fang Chen, and Andreas Holzinger. Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, 10(5), 2021.