Art's Colour Retrieval Dataset & Incremental Learning Colour Palette Generator

43593 - Teresa Dias do Vale e Amador Cruz

Mestrado em Ciência de Dados
Orientadora: Professora Doutora Catarina Oliveira
Co-orientadora: Professora Doutora Tânia Pereira

Data: 02-Jan-2024

UPT DCT DEPARTAMENTO CIÊNCIA E TECNOLOGIA

IMP.GE.212.0

Aos meus pais e familiares, cujo apoio incondicional foi essencial ao longo deste percurso. Aos amigos que me acompanharam, em particular ao meu melhor compincha, sempre incansável e paciente.

Aos professores e educadores, que com a sua dedicação e esforço estimularam o meu gosto pela exprimentação e pelo conhecimento. Em especial, às minhas orientadoras, por aceitarem este desafio e pelo acompanhamento constante e dedicado.

RESUMO

Este projeto apresenta uma ferramenta de combinação de cores baseada no framework GAN, com o objetivo de desmistificar o processo de selecionar cores. Este projecto foi impulsionado por novos desenvolvimentos e avanços na tecnologia de inteligência artificial. Apesar da existência de alguns modelos online, ainda são escassos os projetos sistematicamente documentados na literatura ciêntifica que abordem os desafios associados à combinação de cores.

Para treinar o modelo generativo, foi criado um conjunto de dados composto por paletes de cores derivadas de imagens de filmes de animação. As cores no conjunto de dados foram descritas usando o sistema perceptualmente uniforme de cores CIELUV. A ferramenta está publicada numa página web, o que possiblita o seu uso em vários tipos de dispositivos.

As interações do utilizador com a página web geram um novo conjunto de dados, que pode ser utilizado para o treino continuado do modelo generativo ou aplicado em outros projetos semelhantes. Foi realizada uma análise detalhada das paletes de cores provenientes de imagens, com foco na distribuição dos componentes de CIE Lch. Além disso, a estabilidade da GAN foi examinada ao longo das várias fases do treino.

Os algoritmos introduzidos, bem como os dois conjuntos de dados, servem como base para futuros projetos experimentais no campo da teoria das cores. A ferramenta web não só simplifica o processo de combinação de cores, mas também contribui com um conjunto de dados em constante expansão para exploração futura em projetos relacionados.

Palavras-chave: Teoria de Cor; Redes Adversativas Generativas; Quantização de Cor; Espaço Uniforme de Cor;

ABSTRACT

This project introduces a colour-combining tool based on the GAN framework, aiming to enhance the creative workflow when selecting colours. This was spurred by the promising avenues opened up by advancements in technology and artificial intelligence. Despite the existence of a few machine learning models online, there is still a notable scarcity of thorough documented projects in the published literature that effectively address the challenges associated with colour combination solutions.

To train the generative model, a dataset was curated, comprising colour palettes derived from frames of animated films. The colours in the dataset were described using the perceptually uniform CIELUV colour system. The tool is accessible via a web page, promoting cross-platform usability.

User interactions with the web page generate a new dataset, which can be utilised for ongoing training of the generative model or applied in similar projects. An in-depth analysis of colour palettes in the frames dataset was conducted, focusing on hue, chroma, and luminance distribution. Additionally, the stability of the GAN was examined by studying its losses across multiple training epochs.

The algorithms introduced, along with the two datasets serve as a foundation for future experimental research in the realm of colour theory. The web tool not only enhances colour combination solutions but also contributes with a constantly expanding dataset for further exploration in related projects.

Keywords: Colour Theory; Generative Adversarial Network; Creative Tool; Colour Quantisation; Uniform Colour Space;

Contents

ABBREVIATIONS

CIE	Commission Internationale de l'Éclairage
CNN	Convolutional Neural Network
CQ	Colour Quantisation
CMY(K)	Cyan, Magenta, Yellow (Key, standing for Black)
DNN	Deep Neural Network
GAN	Generative Adversarial Network
LUT	Lookup Table
ML	Machine Learning
RGB	Red, Green, Blue
UCS	Uniform Colour Spaces

List of Algorithms

List of Tables

List of Figures

1 Introduction

Over the centuries, there have been many models, rules, and mythologies created to combine colours harmoniously, but due to the subjectivity of the matter, it is difficult to determine which one produces better results. Currently, many creators rely on their own intuition and knowledge of colour theory to create harmonious colour schemes. However, this process can be time-consuming and subjective, as different individuals may have different interpretations of what constitutes a harmonious colour combination.

Advancements in technology and artificial intelligence have provided promising new pathways for this challenge. While a few machine learning models are available online, there are still not enough thorough and meticulously documented projects that achieve this goal in the published literature.

The goal of this project is to develop a tool to aid in the creation of colour schemes. This tool allows users to easily generate and explore various colour combinations for their projects. By generating multiple colour palettes, it reduces the effort of creating visually appealing designs and allows experimentation with new colour combinations. Whether it's for graphic design, web development, or interior decorating, this tool aims to streamline the process of selecting harmonious colours and enhance the overall creative workflow. This project aims to empower designers and creators to bring their visions to life effortlessly.

1.1 Motivation & Intent

Determining the features of this tool is the basis for its development. This involves identifying the essential functionalities and capabilities required for the tool to accomplish its purpose. Some key features that are beneficial include a wide range of colour palettes and combinations to choose from, the ability to customise and save colour schemes, and an intuitive interface that allows for simple manipulation by the end user. Additionally, it should provide real-time previews of how different colours will look together, as well as the ability to export colour codes. By incorporating these features, the tool will enable its users to quickly select harmonious colours that best suit their needs and preferences.

To allow users to generate colours to fit their work implies a model that can be conditioned, outputting palettes accordingly. For example, a graphic designer working on a website redesign project can use established brand colours as part of a palette, introducing them as input for the generator, which would then suggest a new colour combination.

The interaction of the user with the model entails an interface that allows them to specify their desired input colours and receive suggestions for a new palette. This interactive process ensures that the generated palette aligns with the user's requirements and preferences. Making the generator available on a web page allows it to be accessed on multiple platforms and also provides an opportunity for data collection.

Training a machine learning model requires an adequate dataset that accurately represents the desired outcome or task. To create such a dataset, for training the colour palette generator, a variety of existing colour palettes could be collected from various sources. These palettes would serve as the foundation for the model to learn the patterns and relationships between different colours. Additionally, user feedback could also be collected into a dataset to improve the generator in the future.

1.2 Approach Outline

Reviewing the published literature accommodates for a deeper understanding of the available algorithms to obtain and generate colour or other forms of visual data. Subsequently, guided by this frame of reference, the proposed methods were formalised according to the defined requirements.

This entails a procedure to acquire a dataset of colour palettes, the development of the colour model's structure and respective training, and the design and deployment of the web page where the generator can be accessed.

The first step is acquiring a dataset for training a colour palette generator. This dataset would be used by the machine learning model to learn the patterns and relationships between different colours in the same palette.

After acquiring the data, the next step is to design the model. This involves determining the architecture of the machine learning model and selecting appropriate algorithms and techniques. The model should be designed to effectively learn and generate new colour

palettes based on the patterns and relationships observed in the training dataset.

The model should be able to continue its training process using new data to further improve its ability to generate current colour palettes. This iterative process of training using new data ensures that the model stays updated.

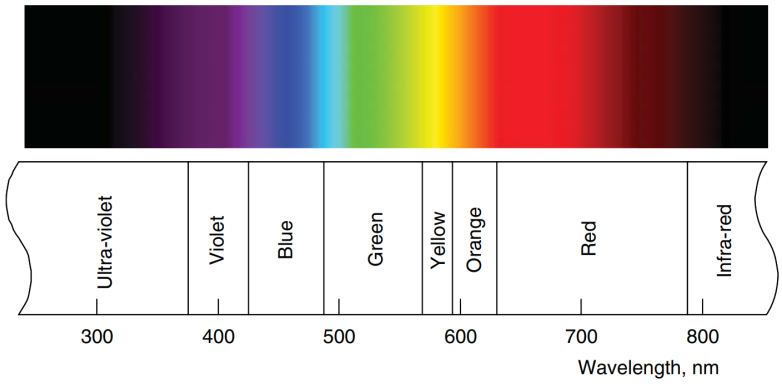
Once the model has been designed and trained using the training dataset, the next step is to create a web interface. This interface must allow users to interact with the model and generate new colour palettes based on their input. The interface should be user-oriented and intuitive, allowing easy navigation and access to the features of the model. In addition, it should include the option for users to export their generated colour palettes for future use. The option for the user to export the colour codes allows for data retrieval, as the colour combination was appealing enough for them to want to access its information.

The rest of the document is organised as follows. Chapter 2 delves into some of the concepts surrounding the history of colour, and describes some of the traditional methods used to combine colours. Afterwards, relevant published literature is reviewed in Chapter 3, followed by the methodology, in Chapter 4. Lastly, the results are analysed in Chapter 5, and Chapter 6 concludes the document.

2 Introduction to Colour

The groundwork for our modern understanding of colour was laid by Isaac Newton, when in 1666, in a dark room at Trinity College, Cambridge, he separated white light into a spectrum of colours using a glass prism. This experiment led to the conclusion that white light is a mixture of all colours in the spectrum, and these spectral colours can be described by their wavelength (Hunt and Pointer, 2011), as presented in Figure 2.1.

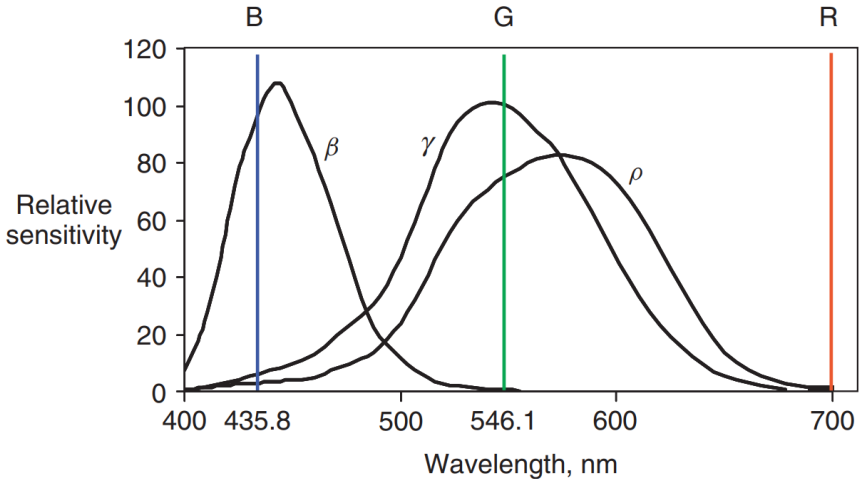
Figure 2.1: Spectrum of visible light, spanning wavelengths from approximately 380 to 750 nanometers. Source: (Hunt and Pointer, 2011)



2.1 Colour - Physical & Perceptual

Colour perception begins with the absorption of light in the retina by photosensitive receptors, cones, which provide colour vision at normal levels of illumination, and rods, which provide monochromatic vision under low light conditions. Rods and cones are not equally sensitive to all wavelengths, and the presence of three different cone types forms the basis for colour vision. Changes in the ratios of the cone responses allow for the discrimination of different wavelengths and intensities of light, as each type of cone is more sensitive to a part of the spectrum: reddish, greenish, or bluish (Hunt, 2004; Hunt and Pointer, 2011; Ohta and Robertson, 2005). Figure 2.2 illustrates the sensitivity to visible wavelengths of each cone.

Figure 2.2: Representative spectral sensitivity curves of the three different types of cone in the retina, together with the wavelengths. R, G, and B used for defining the CIE 1931 Standard Colorimetric Observer, 700 nm, 546.1 nm, and 435.8 nm, respectively. Source: (Hunt and Pointer, 2011)



Colour is often described using attributes such as hue (red, blue, or orange), value, how light or dark a colour is, saturation, and the intensity or purity of a colour (O'Connor, 2021).

> If one says Red (the name of the colour) and there are 50 people listening, it can be expected that there will be 50 reds in their minds. And one can be sure these reds will be very different.
>
> Josef Albers - Interaction of Color

The difficulty of describing colour effectively has led to the creation of many colour systems; some use a numerical coordinate system to define a colour space, such as Munsell's notation, while others describe each colour with a different name or code, such as the widely used system of Pantone (Setchell, 2012).

When using colour, one is often limited by the number of hues available, and mixing is one way to circumvent this problem. Additive colour mixing is the process of creating

colours using different mixtures of light, and the commonly used primaries are Red, Green and Blue. Subtractive colour mixing is used when combining pigments, such as Cyan, Magenta and Yellow inks found in printers (Ebner, 2007; Hunt, 2004). So, many colour systems define colour using the ratios needed to mix it with the system's primaries O'Connor (2021).

While mixing systems are a good way of describing colour to computers, to be displayed or printed, they are not suitable for describing human colour perception, as the perceived distance between equally distant colour pairs is not uniform throughout the colour space (Hunt and Pointer, 2011; Ohta and Robertson, 2005).

he CIE (Commission Internationale de l'Eclairage) proposes Uniform Colour Spaces, also known as CIE UCS, mathematical models that describe colour in a perceptually uniform way. This means that a certain numerical change in colour values corresponds to a consistent change in perceived colour. Two of the CIE UCS are L*u*v* (CIELUV) and L*a*b* (CIELAB), which are widely used in various industries (Ohta and Robertson, 2005). These two colour systems were proposed in 1976, but the CIE gave no recommendation for what cases would suit each one. Studies of CIELAB and CIELUV do not favor either model over the other (Pointer, 1981).

2.2 Colour Theory

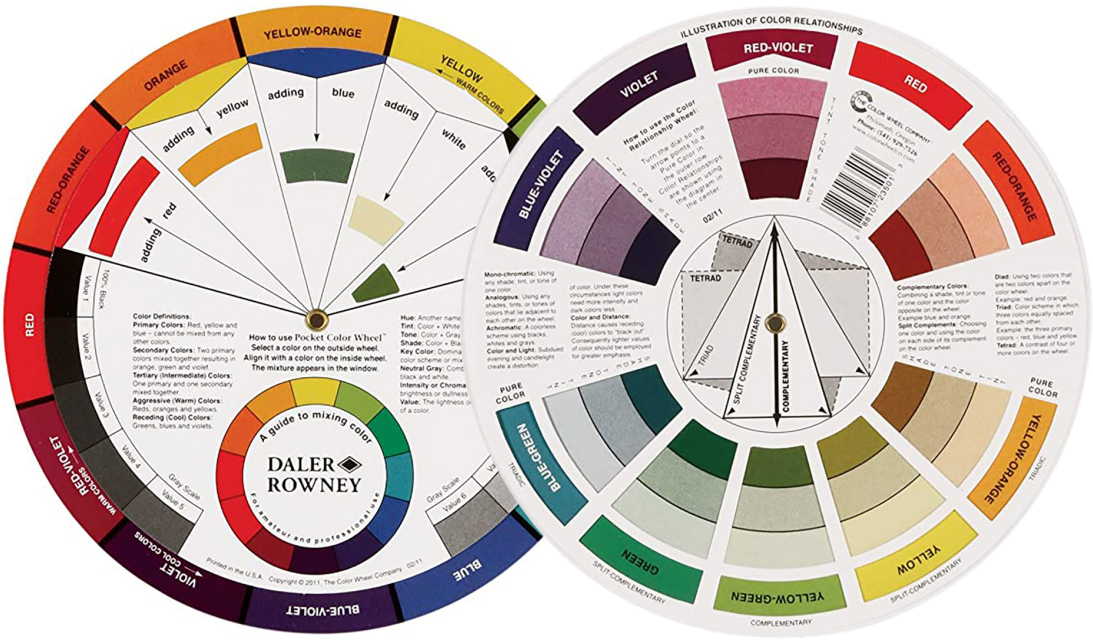
Traditionally, the study of colour theory revolves around mixing and combining colour, rooted in a more practical approach rather than a scientific one. It focuses on an experimental exploration of pigment colour, as the field emerged long before the creation of self-luminous colour. The primary colours used were red, yellow, and blue (close stand-ins for the CMY subtractive model), and black and white to create various shades and tints. The secondary colours can be obtained by mixing a pair of primaries, and tertiary colours by mixing a pair of adjacent primary and secondary colours. When arranged in a circle or polygon with the primaries forming an equilateral triangle and the secondaries and tertiaries placed between the colours used to create them, each opposing pair of hues is called complementary colours, and a set of adjacent hues is called analogous (O'Connor, 2010, 2021).

Colour educators such as Johannes Itten propose colour harmony guidelines, extracted

from similar diagrams to figure 2.2 by employing several geometrical techniques (Lara-Alvarez and Reyes, 2019). Although this construct can be used to aid in the choice of colour combinations, it is often limited to one or two dimensions, leaving out value or saturation information (O'Connor, 2023). Furthermore, these often inflexible rules are set in a vacuum, devoid of physical or circumstantial context (Albers, 2013).

Gadgets like the one provided by Daler Rowney (figure 2.2) provide a tangible representation of the geometric rules prescribed to create colour schemes. These images are the front and back of the same colour wheel tool.

Figure 2.3: Artist's Colour Wheel from Daler Rowney provides geometric rules for creating colour schemes. Source: Daler Rowney product web page.



An alternate approach to choosing colours is taking inspiration from works of art, design or in nature, sampling a limited number of distinct colours from the chosen subject. This method can also be used to provide an intuitive understanding of colour theory Pylypchuk et al. (2021); Weingerl and Javoršek (2018). Even if the extracted palette is not used directly, it can serve as inspiration or reference for creating a new palette, as analysing successful colour combinations in existing art can be a valuable learning experience.

3 Literature Review

Colour quantisation is a process in computer graphics and image processing that involves reducing the number of distinct colours in an image while preserving its visual quality as much as possible. The goal of colour quantisation is to represent the image with a smaller palette of colours, which can be useful in various applications such as reducing file size, compressing images, or simplifying visual representations.

CQ simplifies the colour palette of an image by grouping similar colours together. This simplification is beneficial for creating a harmonious and visually appealing colour scheme in art and design.

Therefore, a review of the published literature of CQ techniques is offered in Section 3.1.

Generative Adversarial Networks (GANs) are generative models designed to generate new, unseen data samples. By training the generator to create realistic samples, GANs can go beyond simply memorizing the training data and generate novel outputs that resemble the underlying data distribution. The adversarial training approach used by GANs, where the generator is trained to fool the discriminator, offers several advantages compared to models that learn directly from the data.

The generator's objective is to produce data that is indistinguishable from real data according to the discriminator. GANs are flexible in capturing complex and high-dimensional patterns in the data. The generator learns to produce realistic samples by iteratively adjusting its parameters based on the feedback from the discriminator. This adaptability is beneficial for tasks with intricate relationships among data features, such as colour palettes.

GANs can dynamically adapt to changes in the data distribution. As design preferences evolve or new trends emerge, GANs can be updated or retrained to generate colour palettes that align with the evolving requirements. By learning from diverse datasets, these models have the potential to reduce bias introduced by manual selection or rule-based algorithms.

Colour palettes are often associated with images, and tasks related to that type of data

are often tackled by GANs. For these reasons, Section 3.2 presents a review of the literature regarding this architecture.

When searching for relevant publications, the multiple regional English spellings of the search terms were introduced, so to not unintentional exclude pertinent articles.

3.1 Colour Quantisation

The term colour quantisation (CQ) is defined as "the process of selecting a set of colours to represent the colour gamut of an image, and computing the mapping from the colour space to representative colours", and was primarily used to reduce the number of bits per pixel necessary to display high quality reproductions of images (Heckbert, 1982).

To survey the methods currently applied, the peer reviewed journals and conference papers available in the b-on search engine were analysed, restricting the search to those written in English and published since 2020 that contained the term "colour quantisation" in the title or keywords. A total of **27** papers were found using these search criteria; **5** of these reviewed the field (reviewed in 3.1.1), and the remaining papers suggested new or improved methods (described in 3.1.2).

3.1.1 Overview Publications

Celebi (2023) proposed a thorough taxonomy, as well as an explanation of the various approaches and applications employed throughout the field's history, and provided a set of ideal characteristics that should be considered when conceiving a colour quantisation algorithm. The topic of colour science has its own section, which discusses the efforts undertaken to establish (nearly) perceptually uniform colour spaces, and the significance of this when computing colour differences. In an experimental comparison of a few algorithms (Kapur et al., 2022), the authors classified them into clustering or splitting the colour space and established that while median-cut scored better, fastoctree scored the fastest.

A more focused approach to CQ, centred on partition-based clustering techniques (Rout et al., 2022), highlights the benefits of reducing the number of colours and decreasing the file size in a generation increasingly reliant on digital devices with storage capabilities. After evaluating several clustering approaches, k-means++ has emerged as the higher

performer. The advantages of swarm-based approaches are spotlighted in this comparison study (Pérez-Delgado and Günen, 2023), like how they carry out more exploration of the solution space while considering many solutions at once while performing well for complex scenarios.

According to Ramella (2021), selecting the appropriate quality assessment metrics is frequently overlooked, as well as the limitations of the RGB colour space. Following an evaluation of popular measures, it is determined that the metrics with the highest correlation to the subjective human rating are VSNR (visual signal to noise ratio), MSSIM (mean structural similarity index), and WSNR (weighted signal to noise ratio).

3.1.2 Proposed Methods

Conforming to Celebi (2023) taxonomy, the remaining **22** publications can be categorised into two groups: **pre-clustering** and **post-clustering** algorithms. **15** studies (albeit not always clearly) are focused on pre-clustering approaches, whereas the remaining **7** papers are mostly concerned with post-clustering procedures.

Median cut, one of the first methods used in colour quantisation, is often compared to the proposed methods (Fernández-Rodríguez et al., 2023; Frackiewicz and Palus, 2022; Kılıçaslan and İncetaş, 2023; Mousavirad et al., 2020; Rong et al., 2020); however, multiple publications emphasise that tree algorithms are effective and appropriate when used for colour quantisation (Fernández-Rodríguez et al., 2023; Pérez-Delgado, 2020, 2021; Yamada et al., 2020; Yu et al., 2023).

Albeit computationally intensive, the k-means algorithm is frequently mentioned as a common CQ technique and was used by Hakanson et al. (2022) to quantify fluorescence. Some authors opt for different versions of the algorithm (Abernathy and Celebi, 2022; Huang, 2021; Thompson et al., 2020), whereas others reinterpret the data to better suit its usage (Bhargav et al., 2021; Frackiewicz and Palus, 2022; Mousavirad et al., 2020).

A straightforward way to implement CQ is to define thresholds to split the colour space into zones, with a colour representative assigned to each one. Several authors have presented this approach with varying degrees of sophistication Artemi and Liu (2020); Chan et al. (2020); Kartika et al. (2020); Kılıçaslan and İncetaş (2023); Lakhal et al. (2023); Lamsrichan et al. (2022); Pereira et al. (2021); Ren et al. (2020).

Most approaches assume that the number of colours in the palette is fixed in advance, and if that number changes, the process restarts, ignoring previously created palettes. However, Park et al. (2023) suggested employing a pre-trained image classifier (DNN) to create a master palette and then combining its colours based on their weight when a smaller palette is needed. Similarly, Fernández-Rodríguez et al. (2023) implemented neural networks to extract a hierarchical model for CQ, and Pereira et al. (2021) used a pre-trained classifier to subdivide the colour space into zones.

3.2 Generative Adversarial Networks

Generative Adversarial Networks offer several advantages, making them a popular and powerful tool in the field of machine learning. GANs are designed to generate new data samples that resemble a given training dataset. This generative capability is valuable in tasks such as image synthesis, style transfer, and data augmentation. GANs have been successfully applied to a wide range of applications, including image and video generation, super-resolution, style transfer, image-to-image translation, text-to-image synthesis, and more. The adversarial training framework of GANs encourages the generator to continuously improve by competing with the discriminator.

Goodfellow et al. (2014) proposed a new framework for generative models that involves training two networks simultaneously: a **generator** G that captures the data distribution and a **discriminator** D that estimates the probability that a sample came from the training data rather than the generative model. The useful parallel presented by the authors is as follows:

> The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles.
>
> **Generative Adversarial Nets** - Goodfellow et al. (2014)

The same paper addresses the potential development of a conditional GAN, in which both adversaries are fed a condition as an additional input. In a conditional GAN, the

condition provided to both the generative and discriminative models adds an extra layer of complexity to the game. The generative model now must produce counterfeit currency that not only looks genuine but also meets the specified condition. On the other hand, the discriminative model must not only detect counterfeit currency but also identify if it fails to fulfil the given condition.

Convolutions are a key operation in convolutional neural networks (CNNs), which are widely used in image and video recognition tasks. In CNNs, convolutions are used to extract features from input data by sliding a small window called a kernel over the input and performing element-wise multiplication and summation operations (Lecun et al., 1998). Convolutions can also be applied to other types of data, such as time series or text, by considering their respective dimensions.

A total of 29 papers were found using the same search engine and restrictions as previously, but with the search terms changed to include ”generative neural network”, ”conditional”, and ”convolutional” in the title or keywords.

The **dimensionality** of the data being produced or analysed by the GAN is a feature that can be used to differentiate publications. **One-dimensional** (1D) data (also called temporal data), **two-dimensional** (2D) data (sometimes called spatial data), and **three-dimensional** (3D) data (also called spatiotemporal data) were found in **11**, **17**, and **1** of the articles, respectively.

Another distinguishing factor is whether the output of the GAN is the answer to the problem studied or additional information to feed to a different model. A range of GAN-based models were trained to perform 2D data segmentation (Jeon et al., 2021; Li, 2022; Oluwasanmi et al., 2020; Wang et al., 2022b; Yu and Choi, 2023; Zhan et al., 2021), other models centre on producing or reconstructing images Li and Zhang (2021); Liu et al. (2022); Prabhat et al. (2020); Tan et al. (2022), whereas others aim to forecast natural phenomena (Dong et al., 2021; Wang et al., 2021, 2023b,a; Zhang and Zhao, 2022). Xi et al. (2020) developed a model to generate skeletal movements in time, using spatial coordinates of 49 joints as features, and Li and Sung (2021) propose a model that generates music of variable length, and Hendra and Kanazawa (2023) estimate image depth. Multiple articles centre around generating synthetic data to augment small or unbalanced datasets (He et al., 2023; Mohammadi et al., 2022; Qin et al., 2020; Ravikumar et al., 2023; Shreekumar et al., 2020; Wang et al., 2022a, 2023b; Xuan et al.,

2023; Zhang et al., 2022; Zhao and Guan, 2023), whilst some seek to correct or complete the original data (Lu and Su, 2021; Zhang et al., 2023).

Appearing in **6** of the articles, stochastic gradient descent (SGD) is an optimisation technique frequently employed in machine learning to train models. However, SGD has limitations in terms of convergence speed and finding the global minimum. To address these issues, various modifications and enhancements have been proposed, such as momentum, adaptive learning rates, and regularisation techniques. These advancements have improved the performance and convergence of SGD, making it a widely used optimisation algorithm in machine learning.

One such improvement to SGD is the ADAM optimiser. Notably, the majority of papers employ ADAM as the optimiser. The fact it is used in **14** of the publications demonstrates its effectiveness in optimising the training process. However, researchers may be relying too heavily on ADAM as an optimiser without proper justification or consideration for alternative optimisation techniques.

4 Methodology

This chapter describes the mechanisms necessary for creating the dataset (section 4.1), training the model (section 4.2), and divulging the colour palette generator on a web page (section 4.3). The steps of these procedures are

4.1 **Colour Extraction**

1. gather adequate images,
2. use colour quantisation techniques to harness palettes,

4.2 **Networks' Architecture**

3. develop generative model,
4. adapt data for training,
5. train model,

4.3 **Website Interface & Data Gathering**

6. design web page,
7. deploy model in web page,
8. retrieve data from web page.

The algorithms were coded in **Python**. Some of the libraries used were NumPy, Pandas, Matplotlib, scikit-learn, TensorFlow and Flask.

4.1 Colour Extraction

The first step in creating an ML model is to gather the training data, in this case, a collection of colour schemes. For this task, images with adequate colour schemes must be retrieved and processed with CQ techniques to gather said schemes. Compared to graphic design or paintings, photographs typically have more subdued colours, hence they are unsuited to provide varied data.

The demand for vast quantities of impartial data has steered the choice to animated films as a source of images. As with design and art, the colours used in animation are

purposefully chosen to convey a story and distinguish multiple elements within a single scene. Conveniently, Studio Ghibli, the Japanese animation studio, has made available frames from various films to the public, through their website https://www.ghibli.jp/. Figure 4.1 provides two examples from different animated films.

Figure 4.1: Example of frames from Studio Ghibli films

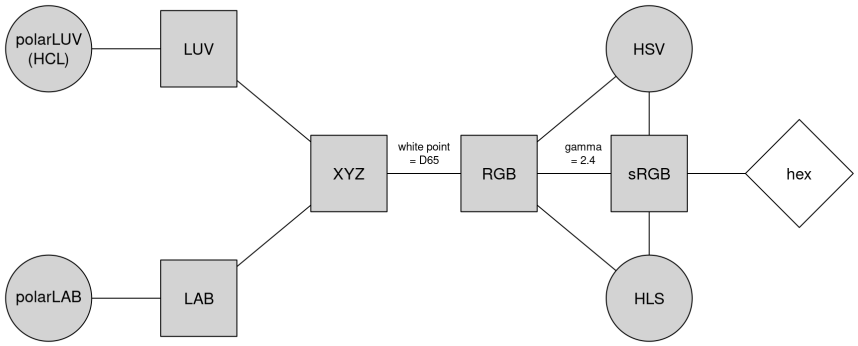
(a) Arrietty, 2010

(b) When Marnie Was There, 2014

To convert sRGB to CIELUV, the formulas described by the CIE are implemented. The sRGB values are first converted to linear RGB using a standard gamma correction, then, the linear RGB values are transformed into XYZ tristimulus values using a conversion matrix. Finally, the XYZ values are converted to CIELUV using another set of formulas.

A graph describing the paths passing through the different colour spaces is displayed in Figure 4.2. Translating HEX colour into CIE Lch requires crossing four different coordinate systems.

Figure 4.2: The pathways between each colour system. Source: *colorspace* r package documentation (Ihaka et al., 2023).



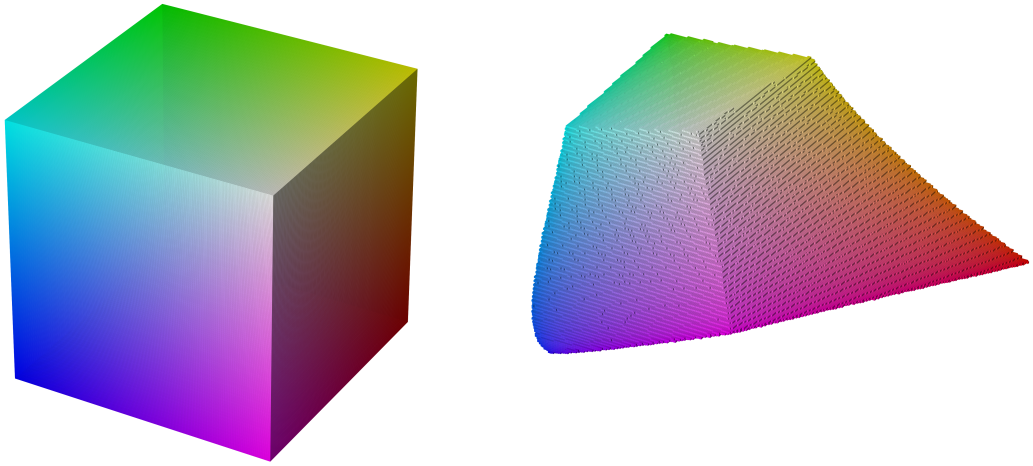
The transformations needed to achieve better colour uniformity deforms the RGB cube,

as can be observed in Figure 4.3. The colours produced by the addictive primaries fit neatly into a cube (three values between 0 and 255 in the case 24-bit colour) if represented in the RGB system, evident in Figure 4.3a. However, when translated to the CIELUV space, ”phantom” colours coexist with plausible colours within the ranges that bound the deformed cube, as shown in Figure 4.3b.

Figure 4.3: The distortion caused to the RGB cube to improve colour constancy

(a) cube in sRGB colour space

(b) cube in CIE LUV colour space



The animation frames are transformed from RGB to CIE LUV space to approximate the colour differences to human perception. The CIE LUV pixels are subsequently clustered using the k-means algorithm, retrieving five centroids as colour palettes. This clustering model proved suitable because of its spherical clusters with clear representatives, and the number of regions is defined beforehand.

The Algorithm 4.1 describes a simplified version of the palette extraction method. Assuming the colour space transformation functions and KMeans algorithm are defined, any image can be used to produce a colour palette of length $n_{clusters}$. This process returns a dictionary with HEX colours as keys and the respective cluster sizes as values.

A palette with five colours allows for a good balance between variety and simplicity. It provides enough diversity to create visual interest without the number of choices becoming overwhelming. Working with a smaller number of colours can help maintain consistency across various design elements.

Algorithm 4.1: Algorithm used to retrieve colour palettes from image.

Required Functions: `Count`, `Flatten`, `LUVtoHEX`, `RGBtoLUV`, `KMeans`

Input: $image_{array}$, $n_{clusters}$

1 $pixels_{RGB} \leftarrow$ `Flatten`$(image_{array})$
2 $pixels_{LUV} \leftarrow$ `RGBtoLUV`$(pixels_{RGB})$
3 $kmeans \leftarrow$ `KMeans`$(pixels_{LUV}, n_{clusters})$
4 $centres \leftarrow kmeans_{centroids}$
5 $labels \leftarrow kmeans_{labels}$
6 $palette \leftarrow$ `dict()`
7 **foreach** $k \in \{0, ..., n_{clusters} - 1\}$ **do**
8 $\quad colour_{HEX} \leftarrow$ `LUVtoHEX`$(centres[k])$
9 $\quad palette[colour_{HEX}] \leftarrow$ `Count`$(labels, k)$

Result: dictionary $palette$ of colours and corresponding cluster size

This is particularly important in branding, where a consistent colour scheme helps establish brand identity. Adding new colours while maintaining the brand's main colours can create a distinct product without losing the original branding. The colour palettes for each tea can in Figure 4.4 maintain the red and yellow of Lipton branding and the green of the tea leaves, adding a new colour for each flavour.

Figure 4.4: Lipton ice tea cans maintain brand's yellow and red, while distinguishing flavours by adding a different colour. Source: Lipton products web page.

By permutating the colours within each palette, the number of palettes is multiplied by

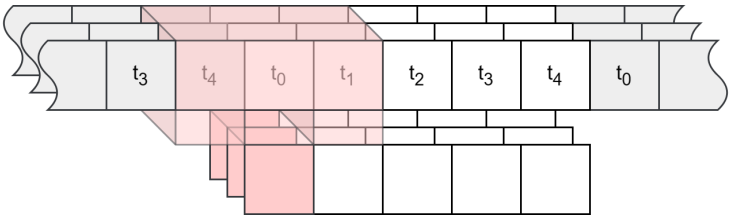
five factorial, given that its order is arbitrary. Furthermore, for each palette, one to four colours are selected as conditions for the model.

4.2 Networks' Architecture

The model architecture was expected to allow the input of an arbitrary number of colours as conditions and output a coherent palette that matched those criteria. The foundational blueprint for this work was **Pix2Pix**, a GAN that is conditioned on an input image and generates a corresponding output image, as proposed by Isola et al. (2017).

Considering that the data processed by the convolutional layers can be shifted and re-ordered, the extremities are padded with each other to encourage extraction of relational data. Compared to using constant values, the kernel can use valid data from previous layers across all the tensor's length. The proposed padding is illustrated in Figure 4.5, and the name suggested for this operation is cyclic convolution. This padding technique virtually joins the tensor's ends, forming a loop.

Figure 4.5: The ends of the tensor are padded with the values from the opposite side prior to the convolution.

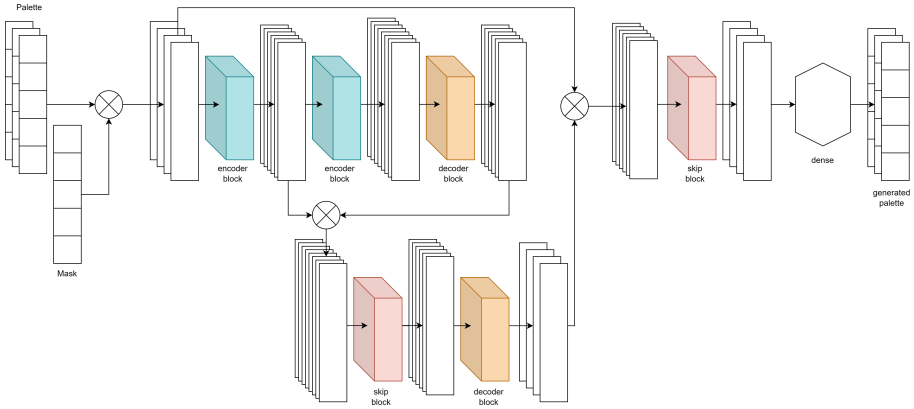


The generator links layers from the encoder path to the decoder path using a convolutional autoencoder called u-net, in keeping with Pix2Pix. Two tensors make up its inputs: a binary vector that indicates which colours are included in the condition and the palette, which consists of conditional and random colours.

After being concatenated into a tensor with four channels, the palette is fed to the encoder path. This path is made up of two blocks with a convolutional layer, a batch normalisation layer and an activation layer each. The convolutions have a size of three and quadruplicate the number of features of the tensor, while maintaining its length. The result of the convolutions is then normalised, subsequently being fed to the exponential linear unit (elu) activation function.

The decoder path comprises two decoder blocks and two skip blocks. The decoder blocks mirror the encoder blocks, instead of reducing the number of features by a factor of four with a deconvolution layer. After the first decoder block, the resulting tensor is concatenated with the output of the first encoder block, and a deconvolution of size one halves the number of features. This is followed by layers of batch normalisation and activation. Subsequently, it passes through the second decoder block and its output is concatenated to the input layers and deconvoluted to half the number of features. The last layers of the generator are a dense layer and a hyperbolic tangent function, resulting in a normalised output of length five with three channels, a synthetic palette.

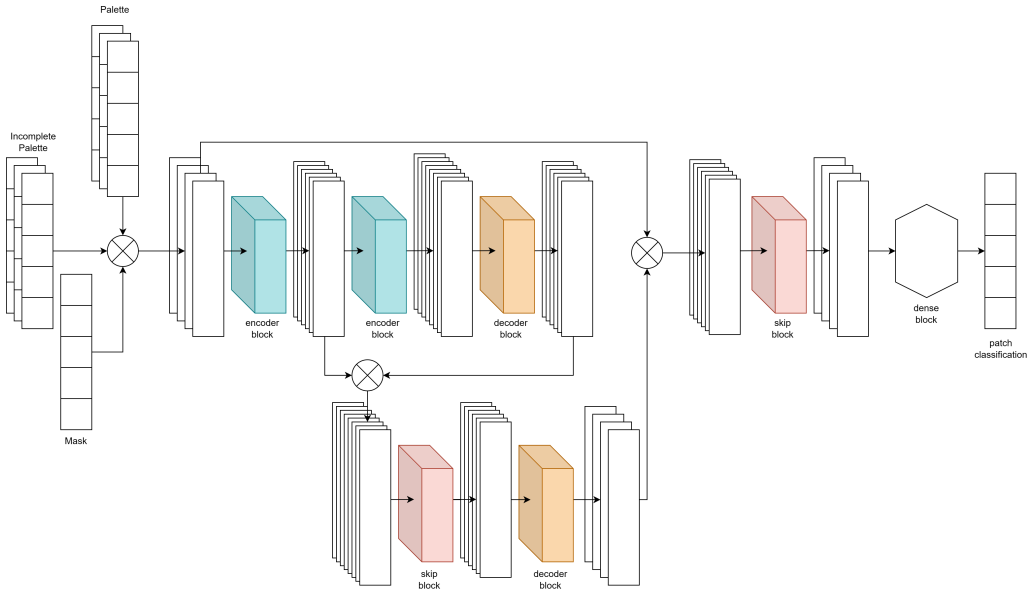
Figure 4.6: Generator Network Architecture.



In an effort to achieve a balance between the two, the discriminator's anatomy was kept as close to the generator network as possible. Together with the palette that needs to be categorised as real or synthetic, it receives the same inputs as the generator.

After being concatenated, the inputs pass through the first encoder block, going from seven to sixteen features. The second decoder block outputs a tensor of length five and depth five that is appended to the input to form a tensor with sixteen features. The dense block has no activation function and outputs a tensor of size five and depth one instead of the generator's three. Borrowing from Pix2Pix, the discriminator does not generate a single value, but instead outputs a tensor of size five and depth one, indicating the probability of each input being real or synthetic. This allows for more nuanced discrimination and helps the network better differentiate between real and synthetic samples. The remaining blocks are identical to their generator's equivalents.

Figure 4.7: Discriminator Network Architecture.



In a traditional GAN, the discriminator is evaluated on how accurately it classifies the inputs as real or synthetic, and the generator on how well it fools the discriminator into classifying its outputs as real. The training process continues iteratively, with the generator and discriminator simultaneously improving their respective abilities. The discriminator's task becomes more challenging as the generator improves its output quality, and vice versa. This adversarial relationship between the two networks drives them to converge towards a Nash equilibrium, where the generator produces realistic outputs, and the discriminator cannot distinguish between real and synthetic samples. This equilibrium marks the successful training of the GAN.

> **Nash equilibrium**, in game theory, an outcome in a **noncooperative game** for two or more players in which no player's expected outcome can be improved by changing one's own strategy. The Nash equilibrium is a key concept in game theory, in which it defines the solution of N-player noncooperative games. It is named for American mathematician **John Nash**, who was awarded the 1994 Nobel Prize for Economics for his contributions to game theory.
>
> **Encyclopedia Britannia** - Eldridge (2023)

As in Pix2Pix, binary cross-entropy (Equation 4.1) is used to calculate both terms in the discriminator loss (Equation 4.2a), and the generator's loss term in misleading the dis-

criminator. The loss function of the discriminator is kept the same as his task remains to discern between real and synthetic palettes, while the generator's loss function (Equation 4.2b) is updated to encourage the production of plausible colours matching the conditions provided. To achieve this, two terms are added to the generator's loss: the difference between the conditions and the corresponding generated colours, and a term reflecting the number of "phantom" colours. The difference between the conditions, $cond_x$, and the corresponding generated colours, $cond_{G(z)}$, ensures that the generator produces colours that align with the desired specifications. By minimising this difference, the generator learns to create palettes that closely resemble the given conditions. Additionally, the term reflecting the number of "phantom" colours, lut_ε, penalises the generator for creating colours outside the RGB range.

$$BCE(p, y) = -(y \ln(p) + (1 - y) \ln(1 - p)) \tag{4.1}$$

$$D_{loss} = 1/2 BCE(D(G(z)), 0) + 1/2 BCE(D(x), 1) \tag{4.2a}$$

$$G_{loss} = BCE(D(G(z)), 1) + \mid cond_{G(z)} - cond_x \mid + lut_\varepsilon(G(z)) \tag{4.2b}$$

To determine if a colour is inside the RGB cube requires translating its values from normalised CIE LUV to sRGB, but this transformation is too computationally expensive to perform every iteration. To circumvent this, a binary lookup table (LUT) of the CIE LUV colour space was defined such that each entry displayable in RGB has zero as value, otherwise, has value one. Afterwards, the binary values were softened using Gaussian blur.

The Algorithm 4.2 provides a outline of the process executed to produce the lookup table used to determine if the generator outputted values coinciding with plausible colours in the CIELUV system, or if it should be attributed a penalty. It presupposes the prior definition of CIE LUV to RGB transformation functions, as well as the average pooling method. The LUT starts as a zero filled 3D tensor $[0]_M$, then each element is attributed the adequate error value according to its index.

Optimisers are algorithms or methods used in machine learning to improve the performance of models by adjusting the parameters of the model during the training process. These algorithms determine how the model should update its parameters to minimise a

Algorithm 4.2: Algorithm used to define the error LUT, a 3D Tensor.

Required Functions: `LUVtoRGB`, `AvgPool`

$lut_\varepsilon \leftarrow [0]_M$

for $elem \in lut_\varepsilon$ **do**

  $RGB_{elem} \leftarrow \texttt{LUVtoRGB}(elem_{index})$

  if $\texttt{min}(RGB_{elem}) < 0 \vee \texttt{max}(RGB_{elem}) \geq 256$ **then**

    $elem \leftarrow 1$

$lut_\varepsilon \leftarrow \texttt{AvgPool}(lut_\varepsilon)$

Result: error lookup table lut_ε for CIELUV space

specific loss function. In the context of training a GAN, optimisers are used to update the parameters of both the generator and discriminator networks. The goal is to find an optimal set of parameters that allows the generator to produce realistic outputs and the discriminator to accurately classify them. By iteratively updating these parameters, optimisers drive the networks towards a Nash equilibrium, where both networks reach their best possible performance.

One commonly used optimiser in training deep neural networks is the ADAM optimiser. It combines the concepts of momentum and RMSprop to update the model's parameters. The ADAM optimiser calculates adaptive learning rates for each parameter by considering both the first and second moments of the gradients. This allows it to converge faster and achieve better results compared to traditional stochastic gradient descent algorithms (Kingma and Ba, 2014).

The AMSGrad version of ADAM is an improvement that addresses a limitation in the original ADAM optimiser. In standard ADAM, the learning rate can become too small over time, leading to slowed convergence or even oscillations in the loss function. AMSGrad solves this issue by modifying the update rule for the learning rate. Instead of using the current squared gradient to update the learning rate, AMSGrad uses the maximum of all previously seen squared gradients. This modification ensures that the learning rate does not decrease too much and allows for better convergence and performance in deep neural network training (Reddi et al., 2019).

In light of this, AMSGrad was selected as the optimiser to be applied, as it avoids the

issues of slowed convergence or oscillations in the loss function that can occur with a decreasing learning rate. AMSGrad ensures better convergence and performance, which ultimately provides a more reliable and efficient approach to addressing GAN training instability.

Following training, the models are saved so they can later be trained using new data. To produce colour palettes, the saved generator is loaded, the conditional colours are inserted in a palette of random values and given as input, and its output is passed to the colour space transformation functions.

4.3 Website Interface & Data Gathering

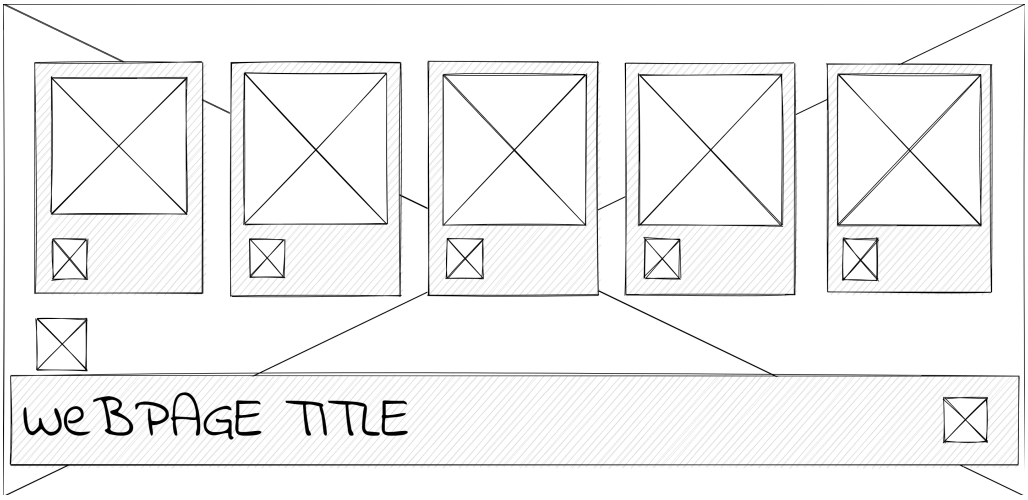
Since the goal of this project is to create a tool for artists or anybody else in need of a colour scheme, the model needs to be accessible to be consulted from somewhere other than the computer on which it has been trained. One way to achieve this accessibility is by deploying the model on a web server or cloud platform. This allows users to access and utilize the model remotely, regardless of their location or device. Additionally, providing an intuitive user interface can further enhance the accessibility of the model, making it easier for users to interact with and obtain colour schemes efficiently.

This interface must include an input field and a means of indicating which colours are required to be present in the palette, as they will be included in the condition. Furthermore, incorporating a feature that allows users to save and export the generated colour schemes can also contribute to the accessibility of the model. This would enable users to easily retrieve and use the colour schemes in their desired applications or projects without the need for repeated generation.

A blueprint of this interface is displayed in Figure 4.8. Each of the five horizontally spaced colour cards have a large square as colour input and a toggle button below them. The toggle is used to indicate if the that colour is part of the condition. The button in the bottom right corner is used to request a new palette. Once the user finds a colour scheme that they are satisfied with, they can access the hex codes by clicking the button on the left above the web page title.

Data retrieval can be achieved by using a button that lets users download palette information. This button signifies that the palette is adequate because the user expressed

Figure 4.8: Web interface wireframe, each of the five rectangular cards has a colour input field and a toggle button to communicate to the model if that colour belongs to the condition.



interest in its contents. A database containing the palettes acquired from multiple interactions is stored so that the model's training can proceed in the future. This database of users' preferred palettes can be used to improve the model's performance and generate even more accurate and diverse colour schemes. Furthermore, the model can adjust and change over time by adding to and updating the database, which keeps it current with design trends and preferences.

Moreover, preferences and trends can be analysed using the database. By analysing the database of user preferences and trends, designers and marketers may gain valuable insights into what colours are popular and how they can be effectively incorporated into their designs. Overall, the database of preferred palettes could serve as a valuable resource for both the model and its users, enabling the creation of more visually appealing and relevant applications of colour in design.

5 Results

In this chapter the results from the palette extraction and the training performance are analysed. The schemes are studied by hue, chroma and luminance (section 5.1), as this colour system is more closely related to the usual way to describe colour. The training losses of each model are studied (section 5.2) by their evolution throughout the epochs.

5.1 Colour Palettes

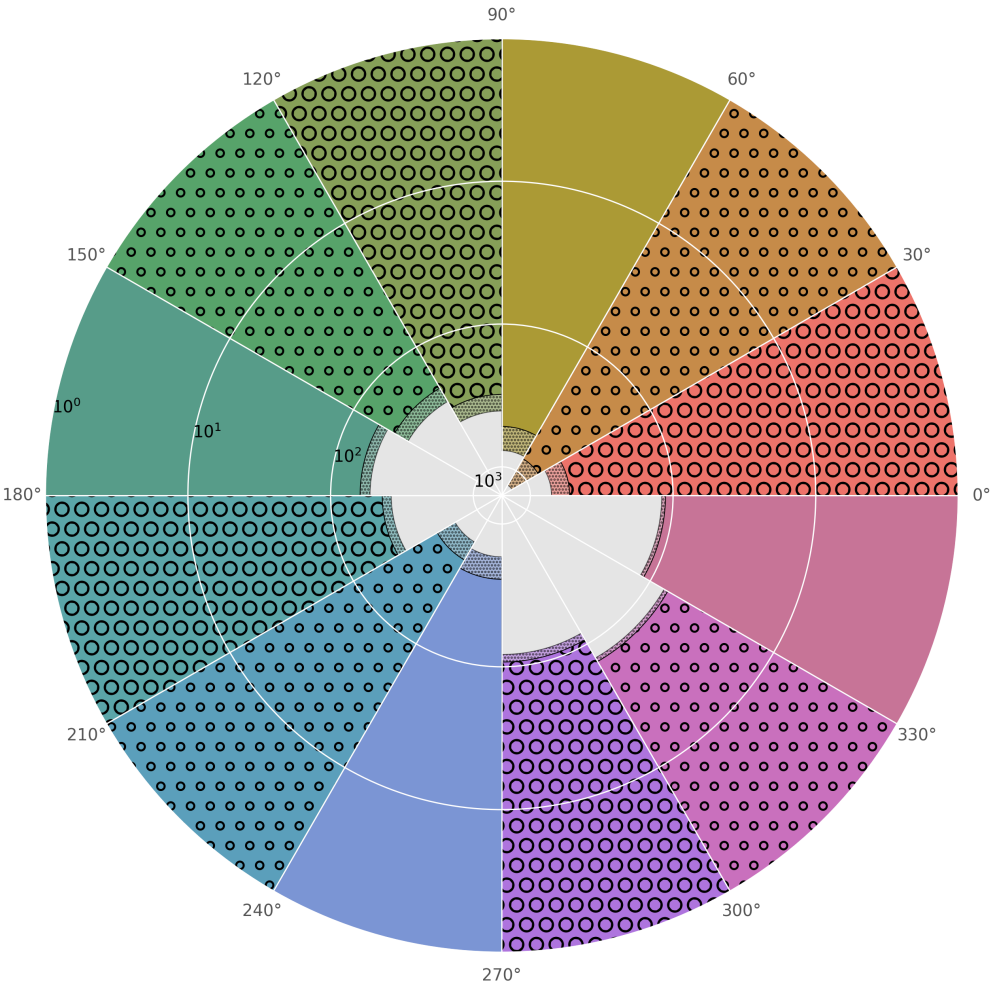
After collecting a total of 1100 frames from 22 studio Ghibli films, each image was converted to CIE LUV colour space and its pixels were separated into 5 clusters. The palette representations in Figure 5.1 are the result from the frames shown in Section 4.1 and illustrate the cluster size by scaling the width accordingly.

Figure 5.1: Colour palettes derived from the examples in Figure 4.1

To analyse the data, the colours were transformed to the polar version of CIE LUV, the CIE Lch (luminance, chroma and hue), and divided into 12 groups based on the hue angle. This colour system was chosen because it is better aligned with the conventional descriptors for colours. When characterising a colour in an everyday scenario, words like *pink*, *turquoise* or *orange* (hues), *light* or *dark* (luminance), and *dull*, *neutral* or *vibrant* (chromaticity) are often employed.

The colour of each “slice” of the circular histogram in Figure 5.2 is representative of the hue group. The radius axis logarithmic value increases towards the centre of the circle. Each section contains two overlapped bars, the dotted taller one, of which only the top can be seen, represents the group total frequency, the other more visible one represents the frequency of palettes containing that hue range. These frequencies are detailed in Table 5.1.

Figure 5.2: A histogram of the colours' hue, plotted in polar coordinates. The radial axis is inverted and in logarithmic units.



The choice of displaying the histogram's bars in a circular axis was the direct consequence of the colour system used for the data analysis, since hue is an angular coordinate. The radial axis is inverted to better showcase the segments colours and facilitate the size comparison of bars farther along the circumference. The radius is scaled logarithmically to boost the the bins with smaller frequencies, as otherwise some ranges would seemingly vanish.

Counting the amount of hue sections that are represented in each palette offers another method of interpreting the colour schemes. By using this counting technique, colours

Table 5.1: Hue frequencies of colour palettes

Hue		by colour		by palette		difference	
0	30	711	12.93 %	531	48.27 %	180	25.32 %
30	60	1311	23.84 %	801	72.82 %	510	38.90 %
60	90	769	13.98 %	521	47.36 %	248	32.25 %
90	120	406	7.38 %	311	28.27 %	95	23.40 %
120	150	276	5.02 %	215	19.55 %	61	22.10 %
150	180	189	3.44 %	161	14.64 %	28	14.81 %
180	210	266	4.84 %	230	20.91 %	36	13.53 %
210	240	664	12.07 %	476	43.27 %	188	28.31 %
240	270	589	10.71 %	410	37.27 %	179	30.39 %
270	300	122	2.22 %	108	9.82 %	14	11.47 %
300	330	77	1.40 %	72	6.55 %	5	6.49 %
330	360	120	2.18 %	113	10.27 %	7	5.83 %
	sum		5500		3949		1551

that appear more than once in the same palette are not overrepresented. Analysed in this manner, rather than counting each of the 5 colours separately, the average number of identifiable hues per palette is **3.59**.

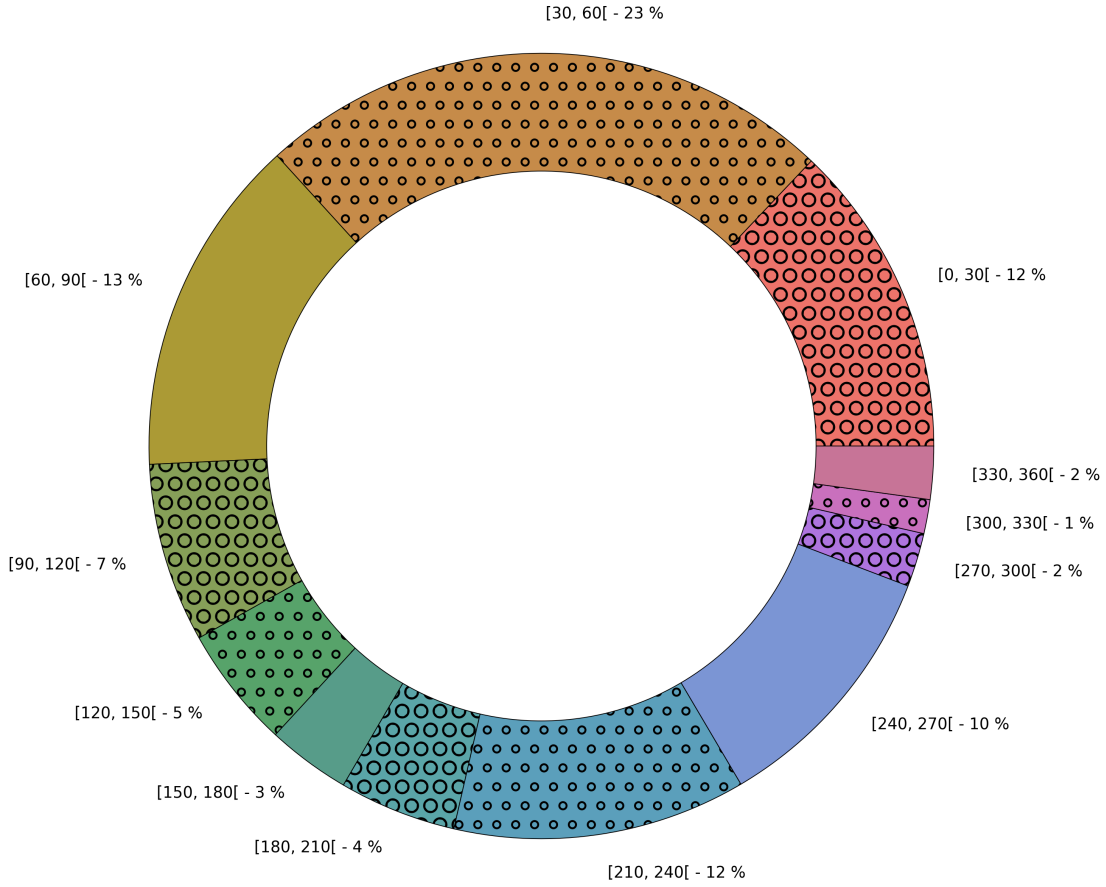
The result of these different counting approaches can be observed in Table 5.1, where the frequencies of each hue range are detailed. As an example, the range of hues between **210** and **240** degrees has **664** colours, but counting by palette representation diminishes this number to **476**. This means that, although 664 colours fall in this hue range, only 476 palettes have an element of that interval. The remaining **188** (28%) are in palettes that already have similar hues.

This suggests that many palettes tend to have overlapping hues within certain ranges, resulting in a smaller number of unique hues per palette. It also highlights the importance of considering both individual colours and their representation within palettes when analysing hue frequencies.

The Figure 5.3 presents the hue frequencies in a pie chart, with each section labeled with the corresponding percentage. This plot provides an opportunity to visualise ratios

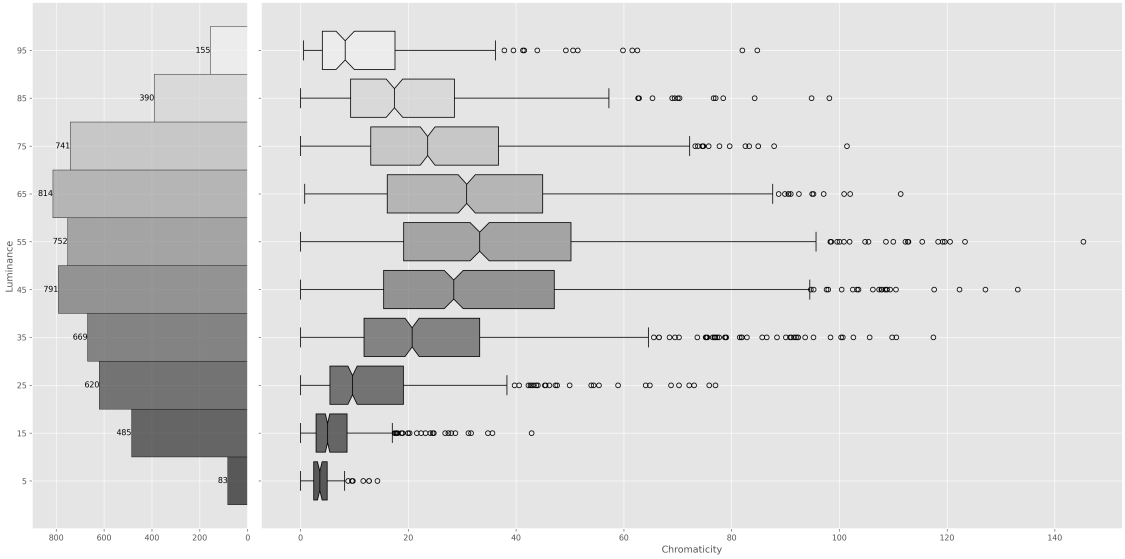
between the twelve hue ranges in a linear scale, as this proportional perspective was not available in Figure 5.2.

Figure 5.3: Hue ranges relative frequencies.



The left plot of Figure 5.4 is a histogram of the luminance of the colours. The plot in the right provides an alternative perspective of the data, by comparing the chroma distribution for each bin interval of the luminance. Chromaticity is limited in the lower range of the luminance spectrum, broadening as luminance increases, until it begins to narrow after the 50 to 60 range.

Figure 5.4: Plots of luminance and chromaticity distributions, in the left a histogram of the colour's luminance, in right a box plot of the chroma for each luminance bin of the left plot.



5.2 Training

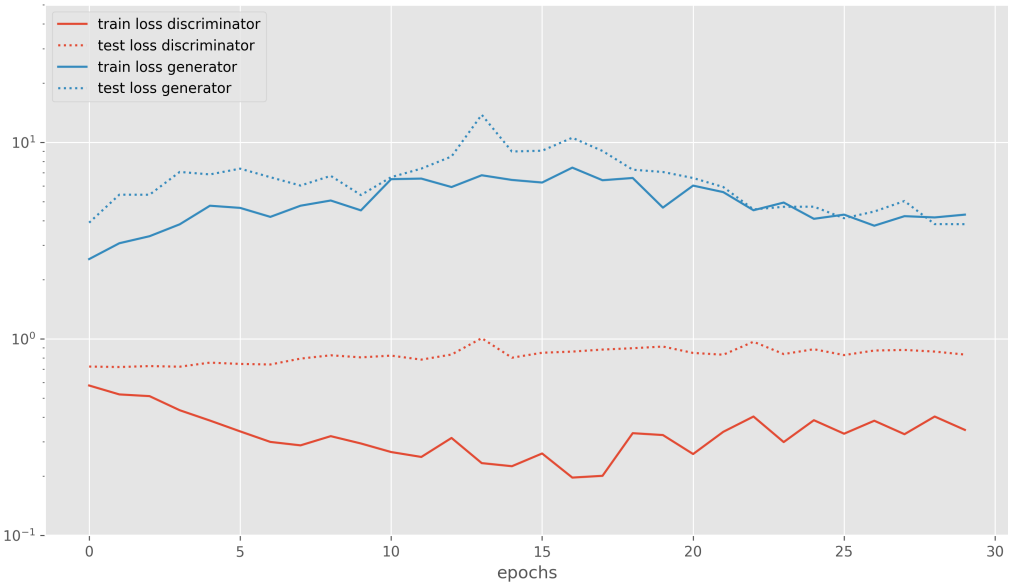
Before training the networks, the colour palette data was prepared by permuting elements from the same scheme and choosing which colours were inputted as conditions. As a result, the dataset's size grew from 1100 to $3960000 = 1100 \times 5! \times (2^5 - 2)$.

The figures in this section present the losses during the training of the generator and discriminator. The losses of generator are plotted in blue hues and the losses of the discriminator are graphed in red hues. This coherent coloring is to ease the interpretation and cross reference between figures.

The vertical axis is logarithmically scaled since the loss terms calculated using BCE are illustrated more accurately in this scope. This numerical representation also emphasises the smaller changes in loss values closer to the origin while diminishing the negligible changes in larger loss values.

The batch average losses of the generator and the discriminator for the first 30 epochs is plotted in Figure 5.5. Each epoch, a fifth of the data was separated and was used to test

Figure 5.5: Line graph of the two networks' losses of first 30 epochs.



if the models are overfitting to the training data. When compared to other ML models, where the objective is to minimise the loss and improve accuracy, GANs strive for Nash equilibrium between the two players.

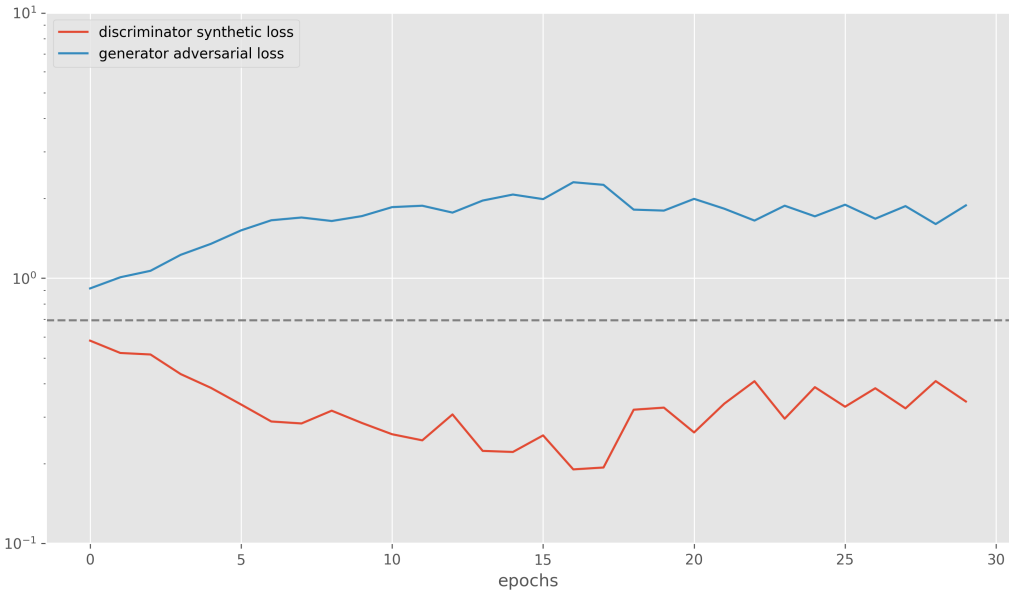
That can be difficult to achieve, as, sometimes, the discriminator can become more capable at detecting synthetic data faster that the generator can learn to synthesise the data. Other times, the generator finds a pattern that fools the discriminator, and stagnates, only producing that output, this failure is called mode collapse.

The two networks' counterbalancing conflict is illustrated in Figure 5.6, with the discriminator's loss term for classifying synthetic images and the generator's loss term corresponding to the fooling of the discriminator along the 30 epochs. Each valley for the generator's loss equates to a peak for the discriminator's loss and vice versa. The horizontal dashed line is marking the threshold loss value where the discriminator would be equivalent to random chance at classifying the data, $BCE(1/2, y)$ (Equation 5.1).

$$BCE(1/2, y) = -(y \ln(1/2) + (1 - y) \ln(1 - 1/2)) = -\ln(1/2) \approx 0.693 \tag{5.1}$$

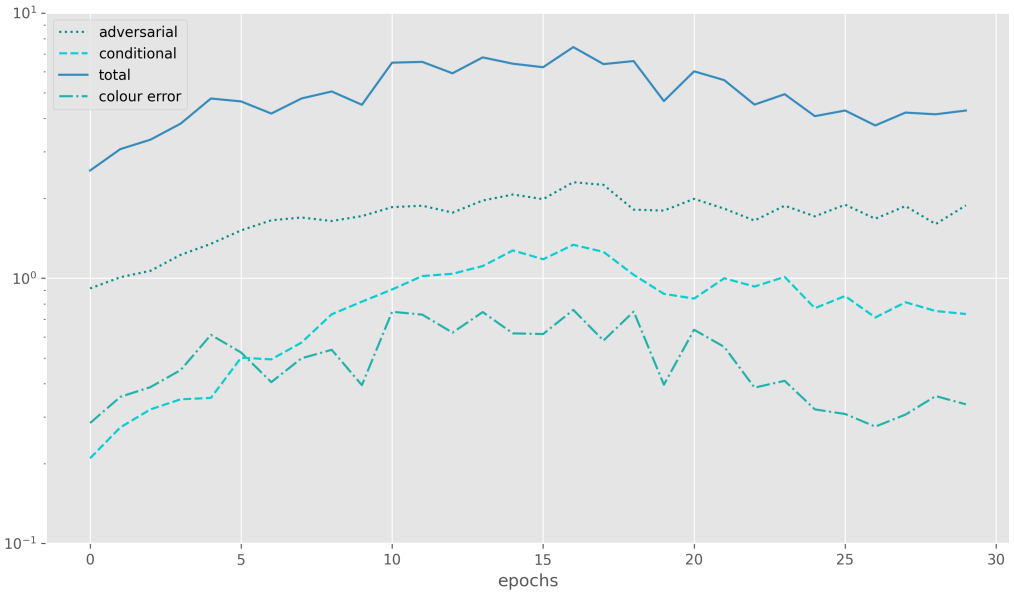
The different components of the generator's loss are displayed in Figure 5.7, along with its total loss. In addition to the adversarial loss present in Figure 5.6, the generator aims to minimise the conditional error, by matching the conditions given, and also reduce

Figure 5.6: Line graph of adversarial losses of first 30 epochs.



the colour error, generating colour inside the bounds of the sRGB cube. This helps in creating color schemes that are not only visually appealing but also plausible and relevant to the desired context.

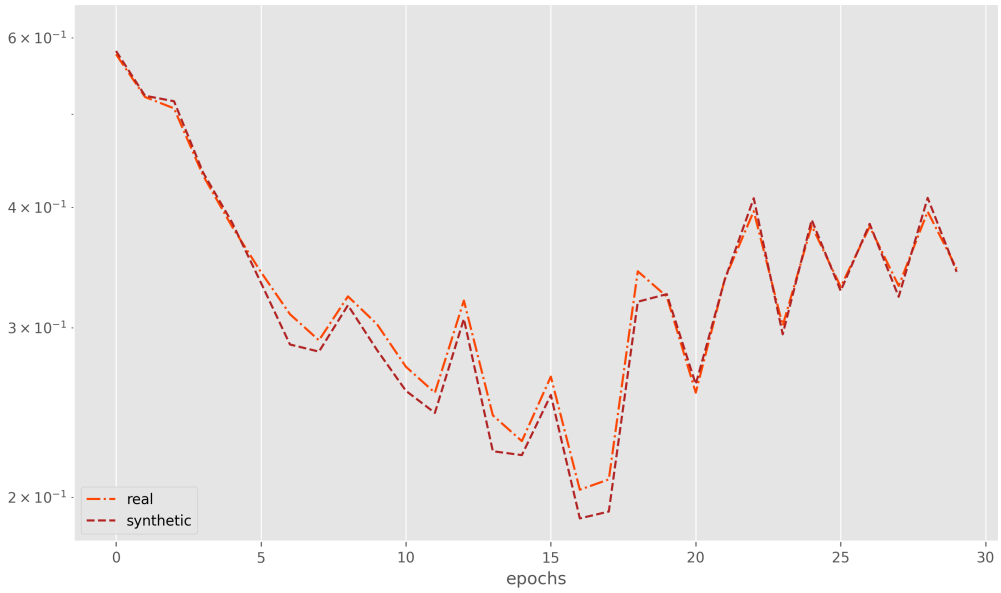
Figure 5.7: Line graph of generator's losses of first 30 epochs.



In Figure 5.8 both components of the discriminator's loss are graphed in the span of

30 epochs. However, the discriminator's actual loss was omitted because, being the average of two similar terms, it overlapped the existing lines and didn't contribute with any additional information.

Figure 5.8: Line graph of discriminator's losses of first 30 epochs.



The discriminator achieves minimum loss values between epochs 15 and 20, potentially compromising it's nash equilibrium with the generator. Afterwards, the value grows as the generator improves its output.

This behavior indicates that the discriminator is becoming less effective at distinguishing between real and synthetic samples as the generator's performance improves. The increase in the discriminator's loss after epoch 20 suggests that the generator is successfully fooling the discriminator, as the discriminator is now struggling to differentiate between real and generated samples.

This adversarial relationship between the generator and discriminator is what ultimately leads to the generation of palettes. Additionally, the conditional loss and the colour space error guide the generator to produce more accurate colour schemes.

5.3 Colour Palette Generator

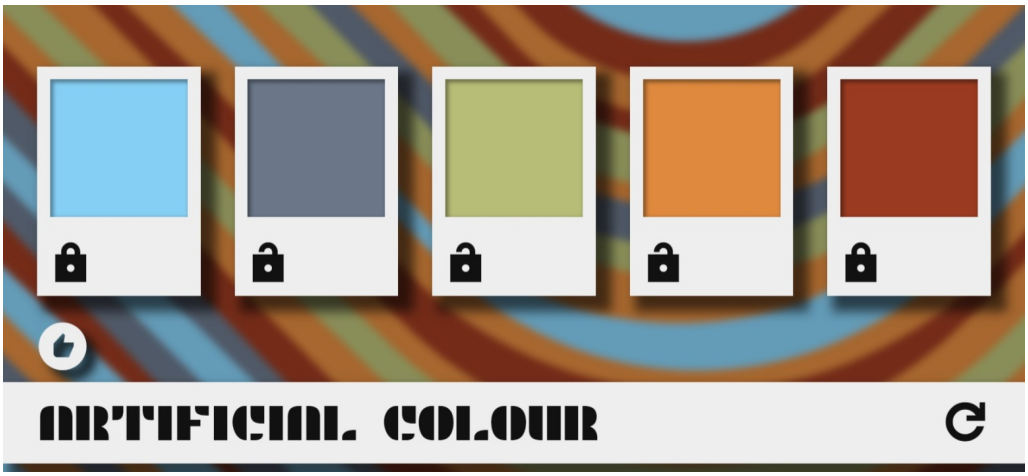
After the training was complete, the optimisers and models are saved so they can be used in future training, and the generator is uploaded to the website so it can be used to generator palettes.

To make the interaction between the user and the model possible, a script was created that takes the sRGB colours given as conditions and

- converts sRGB into CIE LUV colours,
- fills remaining palette with noise,
- creates corresponding binary mask of condition,
- inputs these into generator,
- converts generator's output back into sRGB.

The outputted colour palette is then displayed to the user, through the swatches and background colours. In Figure 5.9 the colours of this document were used as conditions. Additionally to generating a palette, the website also displays a pattern in the background with the suggested colours so the user can preview possible colour combinations. The tool **Artificial Colour** can be accessed in https://artificialcolour.pythonanywhere.com/.

Figure 5.9: Web interface, each colour square is an input field and the lock toggle communicates to the model if that colour belongs to the condition.



The pattern provides an example of how the colours look together. This allows the user to make informed decisions about the suggested colours before approving the palette.

6 Conclusion

The project develops a tool that aids in creating harmonious colour schemes, reducing the effort of creating visually appealing designs and allowing experimentation with new colour combinations. This tool is designed to streamline the process of combining colours and enhance the overall creative workflow.

The development of this tool involved identifying essential functionalities and capabilities required for its purpose, such as a wide range of colour palettes and combinations, an intuitive interface for simple manipulation, real-time previews of different colours, and the ability to export colour codes. These features enable users to quickly select harmonious colours that best suit their needs and preferences. Reviewing published literature helped to understand available algorithms for obtaining and generating colour or other forms of visual data.

The proposed methods are formalised according to defined requirements, including acquiring a dataset of colour palettes, developing the colour model's structure and respective training, and designing and deploying the web page where the generator can be accessed. The process of creating a colour palette generator required careful consideration of the available algorithms, the complexity of the process, and the potential for future improvements.

The process of creating a colour palette generator involved several steps. The first step was to acquire a dataset for training the model, which will be used by the machine learning model to learn the relationships between different colours in the same palette. The next step was to design the model, determining its architecture. The model was designed to effectively learn and generate new colour palettes based on the patterns observed in the training dataset. Once the model was designed and trained using the acquired dataset, a web interface was created to allow users to interact with the model and generate new colour palettes based on their input.

A model that can be conditioned to output palettes accordingly is necessary for users to generate colours to fit their work. The user's interaction with the model involves specifying

their desired input colours and receiving suggestions for a new palette, ensuring that the generated palette aligns with their requirements.

To train a machine learning model, an adequate dataset representing the desired outcome or task was needed. To create such a dataset, a variety of colour palettes from animation frames were collected, serving as the foundation for the model to learn patterns and relationships between different colours. Through user interaction with the web page, data can be collected into a dataset to improve the generator in a later date.

Future work could involve analysing the preferences of its users through the palettes they classified positively. The model should continue its training process using new data to further improve its ability to generate current colour palettes. This iterative process ensures that the model stays updated.

Although the palettes generated have 5 colours, simple modifications could result in models that output any natural number of colours. The model in its current form cannot generate palettes of indeterminate length. To do so would require a model for each palette length.

The colour palette dataset contains information relative to the colours' proportions. However, this knowledge is outside of this work's scope but could be useful for other explorations. As such, this information is preserved in the available dataset. The python scripts of the algorithms described are published in a Github repository, available at github.com/TeresaValeCruz/ArtificialColour/.

This project major contributions are the colour palette dataset and the model used to retrieve it, and the generative colour palette model. **Artificial Colour**, the web tool produced, is a valuable contribution, as in one hand it aids the creatives looking for a colour palette, while in the other hand it acquires new colour palettes that can be used to train other models.

References

Abernathy, A. and Celebi, M. E. (2022). The incremental online k-means clustering algorithm and its application to color quantization. *Expert Systems with Applications*, 207:117927.

Albers, J. (2013). *Interaction of Color: 50th Anniversary Edition*. Yale University Press.

Artemi, M. and Liu, H. (2020). Image Optimization using Improved Gray-Scale Quantization for Content-Based Image Retrieval. In *2020 IEEE 6th International Conference on Optimization and Applications (ICOA)*, pages 1–6. IEEE.

Bhargav, K. J. N. S., Palisetti, S., and Rao, M. (2021). A newton raphson method based approximate divider design for color quantization application. In *2021 18th International SoC Design Conference (ISOCC)*, pages 115–116. IEEE.

Celebi, M. E. (2023). Forty years of color quantization: a modern, algorithmic survey. *Artificial Intelligence Review*, 56(12):13953–14034.

Chan, Y.-H., Xu, Z.-X., and Lun, D. P.-K. (2020). A Framework of Reversible Color-to-Grayscale Conversion With Watermarking Feature. *IEEE Transactions on Image Processing*, 29:859–870.

Daler Rowney (2023). Daler rowney artist's colour wheel.

Dong, J., Chen, J., Wu, Q., Pan, B., and Liu, G. (2021). Day-ahead Prediction of Wind Power Based on Conditional Generative Adversarial Network. In *2021 IEEE Sustainable Power and Energy Conference (iSPEC)*, pages 73–79. IEEE.

Ebner, M. (2007). *Color Constancy*. John Wiley & Sons.

Eldridge, S. (2023). Nash equilibrium.

Fernández-Rodríguez, J. D., Palomo, E. J., Benito-Picazo, J., Domínguez, E., López-Rubio, E., and Ortega-Zamorano, F. (2023). A convolutional autoencoder and a neural gas model based on Bregman divergences for hierarchical color quantization. *Neurocomputing*, 544:126288.

Frackiewicz, M. and Palus, H. (2022). Efficient Color Quantization Using Superpixels. *Sensors*, 22(16):6043.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. *Communications of the ACM*, 63(11):139–144.

Hakanson, E. C., Hakanson, K. J., Anich, P. S., and Martin, J. G. (2022). Techniques for documenting and quantifying biofluorescence through digital photography and color quantization. *Journal of Photochemistry and Photobiology*, 12(100149-):100149.

He, J., Wang, X., Song, Y., Xiang, Q., and Chen, C. (2023). Network intrusion detection based on conditional wasserstein variational autoencoder with generative adversarial network and one-dimensional convolutional neural networks. *Applied Intelligence*, 53(10):12416–12436.

Heckbert, P. (1982). Color image quantization for frame buffer display. *ACM SIGGRAPH Computer Graphics*, 16(3):297–307.

Hendra, A. and Kanazawa, Y. (2023). TP-GAN: Simple Adversarial Network With Additional Player for Dense Depth Image Estimation. *IEEE Access*, 11:44176–44191.

Huang, W. (2021). Enhancing the Bionic Eye: A Real-time Image Optimization Framework to Encode Color and Spatial Information Into Retinal Prostheses. In *2021 IEEE MIT Undergraduate Research Technology Conference (URTC)*, pages 1–5. IEEE.

Hunt, R. (2004). *The Reproduction of Colour*. Wiley, 6 edition.

Hunt, R. W. G. and Pointer, M. R. (2011). *Measuring Color*. John Wiley & Sons.

Ihaka, R., Murrel, P., Hornik, K., Fisher, J. C., Stauffer, R., Wilke, C. O., McWhite, C. D., and Zeileis, A. (2023). Color spaces: S4 classes and utilities.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2017-Janua, pages 5967–5976. IEEE.

Itten, J. (1970). *The elements of color*, volume 4. John Wiley & Sons.

Jeon, U., Kim, H., Hong, H., and Wang, J. (2021). Automatic Meniscus Segmentation Using Adversarial Learning-Based Segmentation Network with Object-Aware Map in Knee MR Images. *Diagnostics*, 11(9):1612.

Kapur, A., Singh, A., Anand, A., Vaidh, D., Luthra, G., Gupta, M., and Boominathan, P. (2022). Comparison of Various Machine Learning Algorithms on Color Quantization Techniques. In *2022 IEEE International Conference on Current Development in Engineering and Technology (CCET)*, pages 1–10. IEEE.

Kartika, D. S. Y., Herumurti, D., Rahmat, B., Yuniarti, A., Maulana, H., and Anggraeny, F. T. (2020). Combining of Extraction Butterfly Image using Color, Texture and Form Features. In *2020 6th Information Technology International Seminar (ITIS)*, pages 98–102. IEEE.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15.

Kılıçaslan, M. and İncetaş, M. O. (2023). Adaptive Color Quantization Method with Multi-level Thresholding. *International Journal of Computational Intelligence Systems*, 16(1):7.

Lakhal, S., Darmon, A., and Benzaquen, M. (2023). A new spin on color quantization. *Journal of Statistical Mechanics: Theory and Experiment*, 2023(3):033401.

Lamsrichan, P., Manthamkarn, V., and Tuntoolavest, U. (2022). Performance Evaluation of the Block Truncation Image Coding with BCH Codes under Noisy Channels. In *2022 19th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 1–4. IEEE.

Lara-Alvarez, C. and Reyes, T. (2019). A geometric approach to harmonic color palette design. *Color Research and Application*, 44(1):106–114.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, S. and Sung, Y. (2021). INCO-GAN: Variable-Length Music Generation Method Based on Inception Model-Based Conditional GAN. *Mathematics*, 9(4):387.

Li, X. (2022). Analysis and positioning of geographic tourism resources based on image processing method with Ra-CGAN modeling. *AIMS Geosciences*, 8(4):658–668.

Li, X. and Zhang, Z. (2021). The comparison between Conditional Generative Adversarial Nets and Deep Convolutional Generative Adversarial Network, and its GUI-related application. In *2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*, pages 601–609. IEEE.

Lipton (2023). Lipton ready-to-drink cans.

Liu, S., Tremblais, B., Carre, P., Zhou, N., and Wu, J. (2022). Image Reconstruction with Multiscale Interest Points Based on a Conditional Generative Adversarial Network. *Mathematics*, 10(19):3591.

Lu, H.-P. and Su, C.-T. (2021). CNNs Combined With a Conditional GAN for Mura Defect Classification in TFT-LCDs. *IEEE Transactions on Semiconductor Manufacturing*, 34(1):25–33.

Mohammadi, A., Jannati, M., and Shams, M. (2022). A protection scheme based on conditional generative adversarial network and convolutional classifier for high impedance fault detection in distribution networks. *Electric Power Systems Research*, 212:108633.

Mousavirad, S. J., Schaefer, G., Celebi, M. E., Fang, H., and Liu, X. (2020). Colour Quantisation using Human Mental Search and Local Refinement. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3045–3050. IEEE.

O'Connor, Z. (2010). Colour harmony revisited. *Color Research & Application*, 35(4):267–273.

O'Connor, Z. (2021). Traditional colour theory: A review. *Color Research & Application*, 46(4):838–847.

O'Connor, Z. (2023). Traditional colour theory in design context: A focus on value. *Journal of the International Colour Association*.

Ohta, N. and Robertson, A. R. (2005). *Colorimetry*. Wiley.

Oluwasanmi, A., Aftab, M. U., Shokanbi, A., Jackson, J., Kumeda, B., and Qin, Z. (2020). Attentively Conditioned Generative Adversarial Network for Semantic Segmentation. *IEEE Access*, 8:31733–31741.

Park, J. H., Kim, S., Lee, J. C., and Ko, J. H. (2023). Scalable Color Quantization for Task-centric Image Compression. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 19(2s):1–18.

Pereira, E. M., Torres, R. d. S., and dos Santos, J. A. (2021). A genetic algorithm approach for image representation learning through color quantization. *Multimedia Tools and Applications*, 80(10):15315–15350.

Pérez-Delgado, M.-L. (2020). Color quantization with Particle swarm optimization and artificial ants. *Soft Computing - A Fusion of Foundations, Methodologies & Applications*, 24(6):4545–4573.

Pérez-Delgado, M.-L. (2021). Revisiting the Iterative Ant-tree for color quantization algorithm. *Journal of Visual Communication and Image Representation*, 78:103180.

Pérez-Delgado, M.-L. and Günen, M. A. (2023). A comparative study of evolutionary computation and swarm-based methods applied to color quantization. *Expert Systems with Applications*, 231:120666.

Pointer, M. R. (1981). A comparison of the CIE 1976 colour spaces. *Color Research & Application*, 6(2):108–118.

Prabhat, Nishant, Kumar Vishwakarma, D., Vishwakarma, D. K., and Kumar Vishwakarma, D. (2020). Comparative Analysis of Deep Convolutional Generative Adversarial Network and Conditional Generative Adversarial Network using Hand Written Digits. In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, number Iciccs, pages 1072–1075. IEEE.

Pylypchuk, O., Krivenko, O., Kolomiiets, Y., Bulhakova, T., and Shmeliova, O. (2021). Formation of Creative Abilities of Design Artists in the Design of Colour Schemes of Fine Art Objects in the Interior. In *Proceedings of the 7th International Conference on Social Science and Higher Education (ICSSHE 2021)*, volume 598, pages 10–15.

Qin, X.-H., Wang, Z.-Y., Yao, J.-P., Zhou, Q., Zhao, P.-F., Wang, Z.-Y., and Huang, L. (2020). Using a one-dimensional convolutional neural network with a conditional

generative adversarial network to classify plant electrical signals. *Computers and Electronics in Agriculture*, 174:105464.

Ramella, G. (2021). Evaluation of quality measures for color quantization. *Multimedia Tools and Applications*, 80(21-23):32975–33009.

Ravikumar, R., Sasipriyaa, N., Thilagaraj, T., Hareesh Raj, R., Abishek, A., and Gokula Kannan, G. (2023). Design and Implementation of Alzheimer's Disease Detection using cGAN and CNN. In *2023 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–7. IEEE.

Reddi, S. J., Kale, S., and Kumar, S. (2019). On the Convergence of Adam and Beyond. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–23.

Ren, S., Wang, M., Shahzad, K., Gao, Z., and Xu, J. (2020). Multi-Carrier Information Hiding Algorithm Based on Angle Structure Descriptor. *IEEE Access*, 8:122565–122578.

Rong, H., Gejia, W., and Jinghang, W. (2020). Color Feature Extraction of Chinese Style Illustration Based on MCCQ Algorithm and Perceptual Analysis. In *2020 International Conference on Innovation Design and Digital Technology (ICIDDT)*, pages 581–585. IEEE.

Rout, J., Das, S., and Mishra, M. (2022). An Empirical Analysis of Digital Image Color Quantization using Partition Based Clustering Techniques. In *2022 IEEE 2nd International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC)*, pages 1–6. IEEE.

Setchell, J. (2012). Colour description and communication. In *Colour Design*, pages 219–253. Elsevier.

Shreekumar, J., Shet, G. K., N, V. P., J, P. S., and Krupa, N. (2020). Improved Viseme Recognition using Generative Adversarial Networks. In *2020 IEEE REGION 10 CONFERENCE (TENCON)*, pages 1118–1123. IEEE.

Tan, Z., Gao, M., Li, X., and Jiang, L. (2022). A Flexible Reference-Insensitive Spatiotemporal Fusion Model for Remote Sensing Images Using Conditional Generative Adversarial Network. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–13.

Thompson, S., Celebi, M. E., and Buck, K. H. (2020). Fast color quantization using MacQueen's k-means algorithm. *Journal of Real-Time Image Processing*, 17(5):1609–1624.

Wang, C., Wang, P., Wang, P., Xue, B., and Wang, D. (2021). Using Conditional Generative Adversarial 3-D Convolutional Neural Network for Precise Radar Extrapolation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:5735–5749.

Wang, F., Al Hamadi, H., and Damiani, E. (2022a). A Visualized Malware Detection Framework with CNN and Conditional GAN. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 6540–6546. IEEE.

Wang, Z., Wang, S., Zhou, C., and Cheng, W. (2023a). AVO Inversion Based on Closed-Loop Multitask Conditional Wasserstein Generative Adversarial Network. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–13.

Wang, Z., Xia, H., Zhang, J., Yang, B., and Yin, W. (2023b). Imbalanced sample fault diagnosis method for rotating machinery in nuclear power plants based on deep convolutional conditional generative adversarial network. *Nuclear Engineering and Technology*, 55(6):2096–2106.

Wang, Z., Xu, N., Wang, B., Liu, Y., and Zhang, S. (2022b). Urban building extraction from high-resolution remote sensing imagery based on multi-scale recurrent conditional generative adversarial network. *GIScience & Remote Sensing*, 59(1):861–884.

Weingerl, P. and Javoršek, D. (2018). Theory of colour harmony and its application. *Tehnicki Vjesnik*, 25(4):1243–1248.

Xi, W., Devineau, G., Moutarde, F., and Yang, J. (2020). Generative Model for Skeletal Human Movements Based on Conditional DC-GAN Applied to Pseudo-Images. *Algorithms*, 13(12):319.

Xuan, B., Li, J., and Song, Y. (2023). SFCWGAN-BiTCN with Sequential Features for Malware Detection. *Applied Sciences*, 13(4):2079.

Yamada, Y., Masuyama, N., Amako, N., Nojima, Y., Loo, C. K., and Ishibuchi, H. (2020). Divisive Hierarchical Clustering Based on Adaptive Resonance Theory. In *2020 International Symposium on Community-centric Systems (CcS)*, pages 1–6. IEEE.

Yonebayashi, H. (Director). (2010). Arrietty. Studio Ghibli.

Yonebayashi, H. (Director). (2014). When Marnie Was There. Studio Ghibli.

Yu, X., Zhuang, H., Cui, Y., Deng, J., Ren, J., and Long, H. (2023). A dichotomy color quantization algorithm for the HSI color space. *Scientific Reports*, 13(1):8135.

Yu, Z. and Choi, K.-C. (2023). Augmenting RetinaFace Model with Conditional Generative Adversarial Networks for Hair Segmentation. In *2023 6th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 890–894. IEEE.

Zhan, Q., Liu, Y., Liu, Y., and Hu, W. (2021). Frontal Cortex Segmentation of Brain PET Imaging Using Deep Neural Networks. *Frontiers in Neuroscience*, 15:796172.

Zhang, J. and Zhao, X. (2022). Wind farm wake modeling based on deep convolutional conditional generative adversarial network. *Energy*, 238(Part B).

Zhang, J., Zhao, Z., Yan, J., and Cheng, P. (2023). Ultra-Short-Term Wind Power Forecasting Based on CGAN-CNN-LSTM Model Supported by Lidar. *Sensors*, 23(9):4369.

Zhang, Y., Jing, B., Wang, S., Pan, J., Du, S., Yang, K., Zhang, Q., Bao, J., Huang, S., and Zhang, X. (2022). Fault Diagnosis Based on C-DCGAN for Rolling Bearing. In *2022 Global Reliability and Prognostics and Health Management (PHM-Yantai)*, pages 1–5. IEEE.

Zhao, X. and Guan, S. (2023). CTCN: a novel credit card fraud detection method based on Conditional Tabular Generative Adversarial Networks and Temporal Convolutional Network. *PeerJ Computer Science*, 9:e1634.