Imperial College London

BEng Individual Project

Imperial College London

Department of Computing

A Machine Learning Pipeline for Automated T-Shirt Graphics Generation

Author:
Samir Hasan Chowdhury

Supervisor:
Dr. Thomas Lancaster

Second Marker:
Dr. Tom Crossland

Abstract

A machine learning pipeline is explored for the automatic generation of novel T-shirt print designs. A continuous data pipeline is developed to ingest and clean marketplace listings, followed by feature extraction for high-level design characteristics. This includes image segmentation to isolate print graphics, colour palette extraction using CIELAB space K-means clustering, multi-label classification for colour-related tags, and large language model inference to extract semantic content from listing titles. These high-level features drive a chain of generative AI models for the dynamic generation of prompts and design elements. The outputs are compiled into SVG via an AST internal representation to enable design manipulation. The proposed system demonstrates the feasibility of generating commercially viable T-shirt graphics at scale with minimal human intervention, and provides a foundation for an evolutionary algorithm based framework for aesthetic evaluation and optimisation in the future.

Acknowledgements

This project would not have been possible without the insight, guidance and judgement of Dr. Thomas Lancaster, to whom I am immensely grateful for the opportunity to have grown under his mentorship. I would also like to thank Dr. Tom Crossland for his invaluable expertise at the halfway mark.

Furthermore, I would like to express my deep appreciation for my loyal, loving friends and family, whose encouragement and belief in me have sustained me throughout this challenging, yet rewarding journey.

Finally, I wish to explicitly acknowledge my late mother, whose great sacrifices laid the foundation for my education. I hope I have made you proud.

Contents

Chapter 1

Introduction

1.1 Motivation

The fashion e-commerce industry has seen rapid growth, generating $133 billion in revenue in 2021 in the US alone and is projected to grow to over $210 billion by 2029 [1]. A significant factor driving this growth is the rise of e-commerce platforms and online marketplaces, which have lowered barriers to entry for clothing merchants, enabling them to market and sell clothing at scale. This is coupled with online shoppers having convenient access to a diverse array of clothing, resulting in a paradigm shift from physical shopping to online shopping.

A popular business model in the apparel e-commerce space is the design and sale of printed clothing. This is in particular due to the outsourcing of the bulk of the operational overhead by print-on-demand services. The ease and scalability of production and fulfilment, as well as the reduced risk and upfront costs with just-in-time production, are significant benefits. However, this accentuates a new bottleneck - the generation of clothing designs has traditionally been resource-intensive, requiring time, active market research and artistic talent.

This motivates the exploration of automated methods for each stage of the design pipeline. Indeed, recent advancements in machine learning, coupled with the ubiquity of data on the internet, have sparked interest in the automation and acceleration of procedures across many business sectors. For market research, computer vision models have been developed to clean large marketplace datasets and identify seasonally popular clothing features [2]. Furthermore, techniques involving state-of-the-art generative artificial intelligence (AI) models have been developed to automate design generation, enabled by their unprecedented ability for new content generation, such as images and text, that resemble the quality of human output [3]. For instance, Stable Diffusion and ChatGPT have been used with a human-in-the-loop to iteratively produce customised T-shirt print designs [4]. This reiterates the ample commercial opportunities for increased automation of the T-shirt graphic generation pipeline.

1.2 Contributions

This project combines ideas seen in prior research to produce a pipeline for automated T-shirt graphic generation. The main contributions are outlined as follows:

- Data is continuously ingested from online marketplaces into a relational database for the market research of printed T-shirts, through web scraping and application programming interfaces (APIs). Other labelled datasets for miscellaneous artistic components are collected, such as colour palettes and fonts.
- Bounding box detection algorithms are experimented with to isolate the print design graphic from the T-shirt images as a cleaning step, to reduce noise for higher quality feature extraction.

- Key artistic features are extracted from the print design. Colour palettes are obtained by clustering in CIELAB space. A multi-label classifier of colour-related tags is developed using a K-Nearest Neighbours (KNN) approach with a distance metric based on the Earth Mover's Distance (EMD) metric. Large Language Models (LLMs) are employed to extract semantic information from marketplace listing titles.
- An AST node architecture is developed for the internal representation of the design, capable of outputting SVG markup.
- The extracted features are used to dynamically generate prompts for generative AI models, for the production of print design components similar to those seen on the market.

Chapter 2

Background

This chapter outlines the relevant techniques, literature, and existing software pertaining to each stage of the T-shirt print design generation pipeline proposed in this study. Where necessary, technical definitions are introduced to equip the reader with the foundational knowledge required to understand the methods and concepts discussed.

2.1 Market Research

A prerequisite for the production of commercially appealing products is to have an understanding of the market demand. This section explores the range of data that can be gathered to inform print designs, as well as the technical considerations for its acquisition.

2.1.1 Existing Products

One approach to market research is to analyse competing products currently or historically sold on the market. Competing businesses may have conducted their own market research, or have more expertise on product design, resulting in their products performing well in the market. Online marketplaces like eBay, Etsy, and Amazon function as real-time repositories of market-tested designs. Listings are equipped with attention-capturing images and titles, and associated metadata such as pricing, sales volume, and user ratings offer quantifiable proxy markers for consumer demand. Therefore, analysis of these designs may provide valuable insight into consumer appeal.

Prior studies have used marketplace data to extract semantic and visual attributes that correlate with product popularity. For instance, Chen et al. analysed a large Alibaba Taobao dataset using multi-label classification and statistical methods to identify seasonally popular clothing features from image data, such as colours and patterns [2]. Similarly, Skenderi et al. forecasted fashion trends by coupling historical clothing sales of a company with Google Trends time series data associated with the colour, category and fabric attributes [5]. The success of these approaches highlights the potential of marketplace-specific, data-driven insights in guiding design decisions for T-shirt print designs.

However, a notable drawback of analysing existing products is the competition within a saturated market. If generated designs are too heavily influenced by pre-existing ones, the lack of originality may result in the product being perceived as uninteresting, potentially limiting the commercial success of this approach. Moreover, a design could be associated with intellectual property - while parody of intellectual property is permissible according to UK law, direct mimicry is not [6].

2.1.2 Trends

An up-to-date awareness of trends can play a role in producing more relevant and novel designs. The relevancy of trends may be cyclical, such as cultural holiday events, or one-off, such as an internet meme. Capitalising on one-off trends in particular can be an effective strategy, as designs can be swiftly listed to market on detection to capture short-term consumer attention. This motivates the exploration of automated trend detection methods.

Trends can be identified through social media platforms, online forums and trend analytics tools. Miao et al. consistently detected and predicted viral trends on micro-blogging websites like Twitter and Weibo, through user tracking [7]. Ravi et al. used image feature clustering from the Instagram feeds of fashion influencers to detect and monitor trends [8].

While these appear to be promising directions for identifying the semantic content of print graphics, they are not without limitations. The tracking of users in particular in these studies raises privacy concerns. Secondly, the topic of a trend may be inappropriate, distasteful, or aesthetically unsuitable. Therefore to make use of trend information, steps to filter such content based on ethics and aesthetics are required.

2.1.3 Data Collection

This section discusses the practical methods to cultivate a high quality dataset for analysis, comparing common methods for collecting data pulled from various sources into a central database.

Pre-compiled Datasets

Pre-compiled datasets are sources of conveniently structured data ready for use. Datasets are often distributed by e-commerce platforms, academic institutions, or companies specifically for research purposes. Chen et al. use a dataset provided by the Alibaba Taobao online marketplace containing half a million clothing products and is populated with item metadata (e.g. images, item names and clothing categories) and user transaction histories [2]. Similarly, Skeneri et al. use the Visuelle 2.0 dataset containing 5355 clothing products' images, textual tags and time-series data, provided by retail fast-fashion company Nuna Lie [5].

While this is an effective option for collecting general data, there are a few notable drawbacks. Firstly, a dataset consisting specifically of printed T-shirts may be difficult to source, due to the niche nature of the research topic. While some datasets may contain a sufficient number of printed T-shirts, the filtering process may pose a significant challenge computationally and technically. Additionally, a pre-compiled clothing dataset alone is static, potentially being outdated. In contrast, the following methods can be used to collect new data continually through automation.

Web Scraping

Web scraping is the automated process of extracting desirable data from the HTML content of web pages [9]. While it is considered a crude data collection method, it is very effective for developing niche datasets at scale. For instance, Andriyanov et al. used Python web scraping scripts to obtain film data from the IMDB website [10].

To prevent the potential strain on server resources and a desire to protect their data and ecosystem, many websites employ anti-scraping measures. These include access barriers, such as logins requirements and CAPTCHAs, to prevent access by automated scripts or "bots". Although these defences can be circumvented, doing so raises ethical concerns, particularly when alternative methods exist to access the data [11].

APIs

Application Programming Interfaces (APIs), if available, are the officially supported methods to access website data. The data requested is often returned in a structured format, such as JSON

or XML.

Online marketplaces in particular may provide API access to their product listings and historical sales. Real-time listing data from eBay is made accessible through its Browse API [12]. The eBay Marketplace Insights API provides up to 90 days of historical sales data [13]. Similarly, Amazon provides access to current listing data through its Product Advertising API [14]. However, there are notable limitations to APIs that make web scraping appear to be preferable:

- APIs are not always open for public use. The eBay Marketplace Insights API, for instance, is limited to those part of the eBay Partner Network and requires a detailed application and approval process [13].
- APIs may not expose all the data desired that is available from the website's frontend. This could otherwise be acquired by web scraping.
- API access may be rate limited and fees may be charged.

2.2 Feature Extraction

Feature extraction distils raw data into condensed, valuable insights. It is used in data analysis for business intelligence purposes and for improving machine learning models that rely on high quality data for training. As discussed, T-shirt graphics and other metadata contain rich information that can be used to generate insights, motivating further discussion in this section.

2.2.1 Image Processing

Image processing algorithms and image features are used to gain a higher-level understanding of images. This could be used to analyse T-shirt print design images, or to filter content. To this end, this section introduces some important computer vision machine learning architectures and begins by discussing image features that can be used to conduct higher-level analysis of images.

Image Features

- **Edges:** Algorithms such as Canny edge detection are used to highlight object boundaries, characterised by sharp discontinuities in pixel intensity [15].
- **Colour Histograms:** Colour histograms encode the distribution of colours in an image.
- **Contours:** This defines an enclosed, connected set of points with the same colour or intensity and defines a boundary [16]. Contours are a useful feature for locating bounding boxes for object detection tasks.

For data cleaning, procedural methods can be developed using these features. An image with an irregular amount or distribution of detected edges may be deemed irrelevant. For design isolation, the print design is likely to manifest as a central cluster of edges within a T-shirt-shaped edge boundary. By comparing colour distributions, images with significantly different histograms from a reference set can be flagged as outliers. This may be a particularly useful feature for filtering out plain T-shirts, as the variance in colours is significantly reduced without a design.

While procedural methods for image processing using these features are interpretable and computationally inexpensive, their performance can vary widely. Procedural methods typically struggle to generalise. In contrast, deep learning architectures can perform significantly better.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep learning models especially well-suited for image analysis tasks. CNNs for computer vision tasks are usually composed of 2D convolutional layers that learn to detect spatial hierarchies of features (e.g., edges, textures, shapes) through training on large image datasets. CNNs can be applied for a wide range of tasks:

- **Image Classification:** This is a task of identifying an image with a label [17]. Chen et al. develop a binary classifier to automatically remove brand logos and advertisements from a clothing dataset consisting of millions of images with an accuracy of 75.5% [2].
- **Object Detection:** Object detection model architectures like Faster R-CNN [18] or YOLO [19] can localise objects within images to identify bounding boxes.
- **Image Segmentation:** Image segmentation model architectures like UNet can be used to produce fine-grained print design image masks through pixel-level classification [20].

CNNs offer a highly effective approach for image understanding tasks, but require more computational resources for inference and training. A key challenge in building a CNN model from scratch is the process of procuring a large, accurately-labelled training dataset. An effective workaround is to employ transfer learning for fine-tuning, a technique where a model trained for one task can be reused to train for a different, but related task [21]. This significantly reduces the quantity of labelled data required.

2.2.2 Colours

A fundamental artistic component in graphic designs is the colour palette, a representative sample of the colours constituting a design. These are often used by artists as a basis for the creation of designs, due to their visual consistency and aesthetic appeal.

To effectively analyse colour palettes, it is important to understand how colours are represented numerically on a computer. Colours are represented with respect to a colour space, a coordinate system mapping that enables quantitative analysis, comparison and clustering [22]. Commonly used colour spaces include RGB, HSV and CIELAB.

RGB

RGB represents colours by their Red, Green, and Blue components. Typically, 8-bits are used for each channel, so that 256 colours can be represented in each channel. Colours are commonly represented as a tuple of integers (0-255), floating point values (0.0-1.0), or a hex code (`#000000`-`#FFFFFF`).

RGB colour space excels for efficient representation of digital images and display purposes as it matches how hardware emits light. However, RGB is unintuitive to work with and performs poorly for perceptual comparison. The Euclidean distance in RGB space does not align well with how humans biologically perceive colour differences, making it unsuitable for clustering or similarity metrics in design contexts [22].

HSV

HSV represents colours by their hue, saturation and value (intensity) components. Hue represents the pure colour and is measured as an angle in a circular representation. Saturation represents how vivid the colour is, while value represents the brightness, and both measured between 0 and 1.

HSV space is more intuitive than RGB for tasks involving colour selection and adjustment, making it popular among digital artists and in design tools. By separating hue from the brightness (value), HSV allows designers to manipulate aspects of a colour independently. However, as with RGB, HSV is not perceptually uniform, as equal changes in hue, saturation, or value do not necessarily result in equal changes in perceived colour [22].

CIELAB

To better model human perception, CIELAB is a colour space introduced by the International Commission on Illumination, and is considered to be approximately perceptually uniform. Colours are represented by three components: **L*** (luminance, 0 to 100), **a*** (green-red axis, -128 to 127) and **b*** (blue-yellow axis, -128 to 127). [22].

Improvements to colour comparison in CIELAB space is achieved by modifying the colour difference metric used. The Euclidean distance metric in CIELAB space (ΔE^*_{76}) is superseded by the CIEDE2000 (ΔE^*_{2000}) formula, which compensates for various perceptual non-uniformities, and is now the standard for accurate colour difference measurement [23].

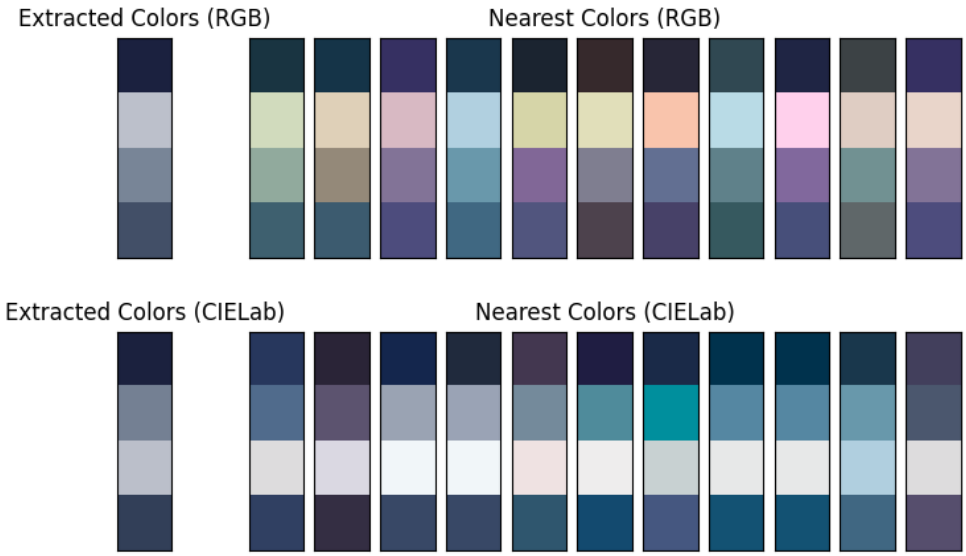

Figure 2.1: Preliminary results of a palette matching algorithm demonstrating the efficacy of CIELAB compared to RGB colour spaces for perceptual comparison.

2.2.3 Semantic Content

Other than visual, artistic characteristics, the content of the T-shirt is also important to analyse. T-shirts are often characterisable by their thematic or stylistic content, such as humour or minimalism. They may also contain specific objects, such as a cat. These can be represented as textual tags that are assigned by a machine learning classification approach.

Andriyanov et al. demonstrate the efficacy of a multi-modal learning approach for the task of classifying film genres from film posters and text data [10]. A neural network architecture is used, combining a VGG-16 and BERT model to infer from film posters and descriptions respectively. A similar approach can be used in the context of T-shirt theme classification, as T-shirt listings from online marketplaces contain both image data and text data, such as titles or descriptions.

2.3 Design Generation

This section discusses existing methods to create print designs, ranging from manual tools like image manipulation software to automatic ones like generative AI models. To begin, this section introduces some important image formats that require consideration in the rest of this section.

2.3.1 Image Format

The choice of image format plays a critical role in the manipulation and quality of T-shirt print designs. Image formats can be broadly divided into two main categories: raster and vector graphics.

Raster Images

Raster images are represented by a grid of coloured pixels. Common raster formats include JPEG, PNG, and BMP. While raster images are well-suited for detailed visual content and photographic data, a problem due to their pixel representation is that they lose information and quality on distortion, such as rotations and scaling to larger resolutions. These transformations introduce undesirable visual artifacts, such as blurring or pixelation, which result in a loss in visual clarity and image quality. To avoid this, a high-resolution is essential when using raster images for print applications. For print designs to be of sufficient quality, most sources recommend print images are at least 300 DPI (dots-per-inch), which translates to a digital image size of 3000×3000 pixels for a 10×10-inch print area [24].

Low resolution images can be made higher resolution by upscaling, a process of pixel value interpolation. As demonstrated in the paper by Dong et al., AI upscaling through the use of a CNN effective at increasing image resolution while minimising quality loss [25].

Vector Images

Vector images are defined by a collection of mathematically defined objects, such as Bézier curves and polygons. The most common vector format is SVG (Scalable Vector Graphics). This format encodes graphical components in XML format, a markup language that stores data in a tree-like structure. The precise nature of SVG component representation allows designs to be transformed while preserving quality, justifying vector graphics as the industry standard for digital print design.

Vectorisation techniques, such as tracing or using differentiable rasterisers like DiffVG, can be applied to convert raster content into SVG [26]. However, as noted by Xing et al., automatic raster-to-vector conversion remains a challenging task. The output SVG markup is often of little utility to professional designers because they are impractical to edit [27].

2.3.2 Existing Graphic Design Software

Digital artists often use comprehensive tools to create commercial-grade T-shirt print designs. The purpose of discussing existing tools is to evaluate the features worth implementing in an automatic print design generation framework.

GIMP

GIMP (GNU Image Manipulation Program) is a free, open-source image manipulation program for raster graphics editing. It supports non-destructive editing through layers and transformable components - a common feature among image manipulation software. A key benefit of this to designers is that it enables iterative design workflows, through the ease with which design elements can be repositioned, scaled, or otherwise modified independently.

In addition to its GUI, GIMP supports automated image manipulation with scripts through an object-oriented paradigm for the representation and manipulation of images. [28].

Inkscape

Inkscape is free, open-source image manipulation software for vector graphics editing, hence using SVG format as its native file type for resolution-independent design output. Similarly to GIMP, Inkscape provides a Python-based scripting interface. This enables the automated parsing and manipulation of SVG elements [29].

Canva

Canva is an example of a cloud-based graphic design platform. In addition to image manipulation capabilities, the platform provides designers with a vast array of high-quality design templates and elements, ranging from stylistic fonts to AI-generated graphics. The orderly structure and visual appeal of pre-selected graphical components provides a firm foundation for automatic print design generation. This is reminiscent of the approach taken by Liu et al., where novel design generation is initiated by selecting a base pattern from a predetermined collection, curated by expert designers [4].

2.3.3 Generative AI

Generative AI refers to a class of machine learning models that can produce new content, such as images and text, based on data seen during training [3]. In the context of apparel design, generative AI is a promising emerging technology for increasing automation in the creation process. There are many generative AI model architectures that have been developed for both image and text generation. This section explores the architectures and techniques currently available and how they can be leveraged for print design generation.

Diffusion Models

Diffusion models learn to generate content through a forward diffusion process and a reverse denoising process [30]. In the training step, the forward process involves adding noise to training images over many steps until the data becomes indistinguishable from random noise. Then, the model learns to reverse this process in the denoising stage from random noise back into a coherent image. At inference time, a trained model can start from pure noise and generate a new image through the iterative denoising process.

As explored by Liu et al., diffusion models are useful for generating prints with artistic or thematic elements for customised apparel design [4]. Stable Diffusion in particular is capable of synthesising images via text prompts using a cross-attention mechanism [31]. Alternatively, an inpainting process can be used to modify existing designs in desired regions, by replacing a masked area of an image with new content [31].

Generative Adversarial Networks

The Generative Adversarial Network (GAN) architecture consists of two competing models in a minimax game: a generator and a discriminator model [32]. The generator is responsible for creating synthetic images similar to those contained in the training set, while the discriminator seeks to distinguish the generated output. The generator learns to produced better images when the discriminator successfully detects generated output. Meanwhile, the discriminator learns to better detect generations when it incorrectly classifies an image. Over many iterations, the adversarial nature of the game enables each model to learn to get better at their respective

tasks. Consequently, GANs are capable of producing high-quality images, providing another architecture for image generation.

Evolutionary Algorithms

Evolutionary algorithms (EAs) are a type of reinforcement machine learning algorithm for optimising black-box functions, inspired by natural selection and Darwin's biological theory of evolution [33]. Potential solutions to a problem are treated as individuals in a population, where a fitness function is used to measure the quality of a solution based on its characteristics. One defining characteristic of EAs is their production of new solutions by mutating or combining the existing ones. T-shirt designs have an array of properties that can be modified or transferred from other images to produce novel combinations. For example, the semantic content of one design can be combined with a colour palette of another. This motivates the exploration of EAs, for which there appears to be little literature on their use in automated fashion design.

Grabe et al. use EAs in an attempt to guide a GAN to generate clothes of a target style [34]. Style is determined by clustering on the embeddings of the penultimate layer of a ResNet50 model. The fitness is measured as the probability of belonging to the target style cluster, but the study concludes that the designs do not always visually correspond to the target style as expected. This highlights the difficulty in producing a robust fitness measure for visual tasks.

T-shirt designs have a diverse range of characteristics that can independently maximise aesthetic appeal. For example, a text-based design looks significantly different to a purely image-based one, yet both can be commercially viable. The Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) framework is a specific approach within EAs to optimise for both quality and diversity, by maintaining fit solutions among a descriptor space [35]. The descriptor space for diverse designs could be the text-to-image ratio, or related to colour palette complexity, allowing the algorithm to discover a broad range of high-quality designs.

Style Transfer

There are many art styles that T-shirt designs can be characterised as, such as watercolour, graffiti and silhouette. An interesting approach for generation, perhaps as a mutation or crossover operation in EAs, would be to transform existing designs from one style to another.

Gatys et al. introduce neural style transfer, a method using image-to-image models to re-render an image, separating and re-combining its structural content and artistic style with another image [36] [37]. This results in a hybrid image that maintains the layout of the original while adopting the visual characteristics, like colour and texture, of the reference style.

A recent online trend that has gained traction uses OpenAI's `GPT-4o` model to transform images into a target style, like that of Studio Ghibli, with remarkable visual consistency. This is depicted in Figure 2.2, highlighting the feasibility and appeal for modifying print design style [38].

Large Language Models

Large Language Models (LLMs) are generative AI models trained to predict a coherent sequence of words (or tokens, to be precise) through statistical representations of language. LLMs are typically based on the transformer architecture by Vaswani et al., where the self-attention mechanism enables the model to learn contextual relationships between words, regardless of their position in the sequence [39].

An aforementioned challenge in using generating AI for image generation is that the output may contain illegible or incoherent text. In contrast, LLMs are highly capable of producing coherent strings, motivating the experimentation of LLMs in a procedural approach for text-based print design generation. While a well-known drawback of LLMs is its tendency to produce hallucinations, this is inconsequential in the development of T-shirt designs.

Figure 2.2: A `DALL-E3` generated design (left) converted to Studio Ghibli style via `GPT-4o` (right) [38]

.

Another use case of LLMs is for prompt generation for the input of text-to-image models. Iteratively refining these prompts may be necessary to achieve designs suitable for printing, a task that can be automated using LLMs. This is explored effectively by Liu et al. to iterate on a Stable Diffusion prompt using ChatGPT [4].

LLMs are not limited to producing human language - the LLM4SVG framework is capable of producing well-formed SVG markup for vector image generation [27]. However, such models struggle to generate intricate designs suitable for T-shirts and remains an active area of research.

Considerations

Despite the appeal of generative image models for the scalability of content generation, they may still lack the consistency needed for commercial-grade prints. This varies significantly by model. Even with state-of-the-art models however, while images produced are impressive at a first glance, there are often issues with consistency of finer details. At times, images can appear uncanny and evidently AI-generated. This has traditionally been a challenge for text content, which can appear illegible or nonsensical.

Furthermore, state-of-the art image generation models typically require significant GPU memory to store the weights and biases. This limits the ability to conduct inference and training locally on limited hardware. However, LoRAs (Low Rank Adaptations) can be used to fine-tune large models on limited hardware by introducing trainable, low-rank matrices into existing model weights [40]. Alternatively, cloud compute or API services are widely available.

2.4 Design Evaluation

The remainder of this chapter discusses methods for design evaluation. Design evaluation is considered for two possible use cases: market research on popular features to better inform design contents and for the development of a robust fitness function for iterative improvement in an EA framework. This section explores methods seen in literature for digital art evaluation.

2.4.1 Human Evaluation

The complexity of image understanding and the subjectivity of aesthetics pose significant challenges for automated evaluation. Therefore, the gold standard for evaluating artistic content is human judgement.

User-Driven Generation

Liu et al. evaluate T-shirt design generation iteratively by requiring the user to manually approve or disapprove, until the user is satisfied with the output [4]. Ding et al. propose an AI agent approach for design conceptualisation to simulate the design pipeline and achieve improved aesthetic outcomes [41]. While these methods are effective and appropriate for tailoring designs for a specific individual user, they require constant human attention and intervention for design generation and improvement.

Sales Metrics

A more scalable approach to evaluation is by directly listing to the market. Online marketplaces generally monitor metrics of listed products for sellers to make data-driven business decisions. These metrics are correlated with sales performance and can be used to effectively measure the quality and relevance of a given product without necessarily achieving sales:

- **Impressions:** When a shopper makes a search query, the impression count increases if the listing appears on the page, measuring relevance.
- **Clicks:** The number of times a shopper visits the main sales page of a listing.
- **Click-through-rate:** The proportion of clicks relative to impressions. CTR is associated to commercial appeal relative to other similar products from initial inspection of the title and thumbnail image.

A/B testing is a method to identify which design variant customers prefer, and can be used to make data-driven decisions for iteratively optimising designs. A/B tests with respect to the metrics described can be effective measures of commercial fitness of a T-shirt. However, there are other possible confounding variables requiring consideration, such as price, shipping details and seller reputation, that may colour the interpretation of these metrics.

Sentiment Analysis

Product reviews, forum discussions and focus group interview transcripts are examples of subjective, qualitative data used by businesses to analyse the appeal of commercial products. Traditionally, this information would require humans to interpret, which can be expensive and time consuming. However, with natural language processing techniques and recent advancements with LLMs, the synthesis of such information can be increasingly automated with remarkable accuracy. This motivates the exploration of sentiment analysis in a data-driven approach to design iteration.

Prior studies aggregate text polarity to classify consumer sentiment. Abdulsalam et al. measure net polarity, where common words have a recorded ± 1 polarity value from a dataset (or Bag of Concepts) [42]. However, a deep learning approach, using Recurrent Neural Networks for instance, offers significantly higher accuracy with the increase in context awareness [43]. Polarity by itself is limited in its specificity of insight, however. This is addressed in a study by Park et al., where sentiment polarity of specific features - identified using LLMs and clustering algorithms - are measured using a fine-tuned BERT model [44].

A drawback of a sentiment analysis evaluation approach is that acquiring organic design reviews can be difficult. Moreover, reviews may not go into detail about the particularities of the features. Therefore, this direction is unlikely to be fruitful in the context of this study.

2.4.2 Machine Learning Evaluation

While machine learning methods are still not as effective as humans, the scalability of machine learning approaches are appealing for evaluating the generated print designs.

Aesthetic Scoring

Aesthetic scoring models are a branch of computer vision models for quantifying perceived image quality. In the context of T-shirt graphic generation, these models can help filter or rank designs by their visual appeal and likelihood of commercial success.

Metrics such as the Structural Similarity Index Measure (SSID), Inception Score (IS) and Fréchet Inception Distance (FID) exist to evaluate image generation algorithms on a holistic level, making them useful benchmarking metrics for model comparison. For instance-level artistic evaluation however, Chen et al. introduce the ArtScore metric, which quantitatively evaluates how closely AI-generated art resembles fine art as the output from a neural network [45]. In addition, Kao et al. note the existence of correlations between aesthetic quality and semantic content in images to achieve semantically-aware aesthetic scoring [46]. This is a crucial consideration for T-shirt designs where semantic content is constrained.

Alignment

A semantic alignment score measures the similarity of the semantic content of data. Models such as CLIP (Contrastive Language Image Pretraining) enable direct image and text comparison, as the model trains both an image and text encoder together with a contrastive loss function to align their representations [47]. This enables the use of cosine distance to measure image and text similarity of their respective embeddings. In the context of T-shirt graphic generation, alignment scoring ensures the graphic generated reflects the content specified in prompts or other associated text data.

Chapter 3

System Overview

With the necessary prerequisite information discussed, this chapter gives a high-level overview of the implementation of the project. In summary, the project is divided into two key stages. First, data is curated to inform the semantic and stylistic content of designs. This is followed by design generation with commercial models, based on insights collected. The possible approaches discussed in the background are evaluated to justify the proposed approach.

3.1 Pipeline Overview

3.1.1 ETL Architecture

ETL (Extract-Transform-Load) is a data pipeline framework for collecting data from a variety of sources, cleaning and transforming the data, before finally loading it into a centralised location in a structured format. Data pipelines allow for the continuous and scalable collection of large datasets, hence being a suitable framework for the data collection component of this project.

3.1.2 Datasets

This section outlines the data collected for the purposes of analysing existing print design content and ultimately generating T-shirt designs informed by this data. Other artistic component data are also collected, such as curated colour palettes and fonts.

Marketplace Clothing Data

Online marketplace data is collected instead of general topical trend data. This solves the problem of determining whether a design is semantically suitable for printed T-shirts because other T-shirts already incorporate it. Online marketplaces are rich repositories for printed clothing data, as they are consistently sold in large quantities on these platforms and often packaged with useful metadata. T-shirt data is collected from the following sources:

- **eBay Seller Hub:** A research platform for sellers and provides up to 3 years of historical market data on the eBay platform. Since the data is behind a login, limited amounts of this data is web scraped.
- **eBay Browse API:** Provides access to real-time market listing data. Because the API is highly accessible and provides real-time market information, this is the preferred method for collecting data and consists of the majority of the designs collected.
- **Etsy Listings:** An online marketplace for more unique items in comparison to eBay, which often lists second-hand goods as well. The data collected from Etsy is web scraped from live listings. However, as Etsy employs anti-scraping measures, this is also kept to a minimum.

Table 3.1: Marketplace T-Shirt Datasets

Table 3.2: eBay Seller Hub

Column
item_id
title
item_url
img_url
avg_sold_price
avg_shipping_cost
total_sold_count
total_sales_value
date_last_sold

Table 3.3: eBay Browse API

Column
item_id
title
img_url

Table 3.4: Etsy Listings

Column
item_id
title
item_url
img_url
seller_star_rating
seller_review_count
currency_symbol
price

Table 3.1 outlines the specific attributes initially ingested from each data source. The `item_id` is a unique identifier for the listing. The `img_url` attribute contains the URL string to the thumbnail image of the T-shirt sold - storing the URL as opposed to the image data itself saves memory and can be easily downloaded at runtime using a HTTP GET request. The `title` column provides a succinct description of the item being sold. As online merchants aim to attract buyers to their products, the title contains a rich set of keywords relating to the semantic and thematic content of the design, hence providing valuable insight for design analysis. These attributes are shared between each source and allow the datasets to be merged into one.

The other quantitative fields such as `price`, `total_sold_count` and `date_last_sold` are potentially useful proxy measures for the popularity, quality and trendiness of a given design or theme. This data can be used to filter the ingested clothing data, as apparel with higher prices and sellers with better star ratings are more likely to have better designs. This data could potentially be retained by imputing the missing data from the other sources, or a method to aggregate these values into a unified score could be developed. However, this is not implemented at this time.

ColorHunt Colour Palettes

ColorHunt is an expertly-curated website repository of colour palettes, created by Gal Shir [48]. The data collected from ColorHunt is outlined in Table 3.5.

Table 3.5: ColorHunt Colour Palette Data

Column	Description
`palette_id`	A unique identifier for the colour palette
`colors`	A list of four hex string colour values
`likes`	Number of likes
`color_tags`	A list of colour-related tags
`other_tags`	A list of theme-related tags
`date`	Time since palette submission to ColorHunt
`url`	The URL of the colour palette

Each colour palette is identified with a `palette_id`, which is used to form its URL path. A colour palette consists of exactly 4 RGB colours as hex strings. Each palette is labelled textual tags of the colour names, styles and themes of the palette, such as `red`, `pastel` and `winter`. There are a total of 49 colour/other tags. Popularity of a palette is measured by `likes`. The `date` entries are string values, such as `"3 months"`, to indicate how long ago the palette was submitted. This format is not particularly suitable for analysis, compared to a numerical time stamp. Since the palettes are scraped in reverse-chronological order, rather than using this date

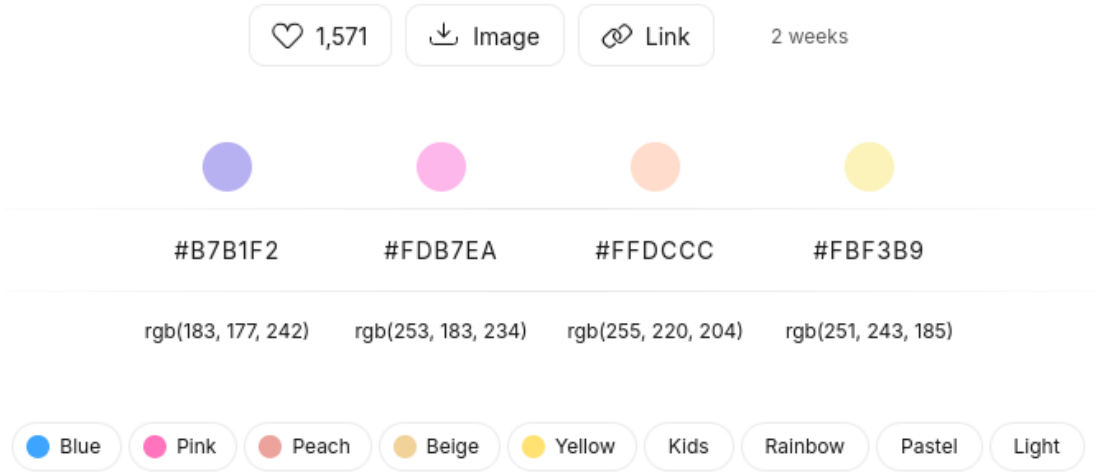

Figure 3.1: A colour palette with associated labels and metadata from ColorHunt.co [48].

value, the time of submission can be imputed by assuming the palettes are uploaded at a uniform rate within the time interval. However, a simpler approach to use the submission number is used instead.

Precompiled Font Database

A free dataset of 18575 DaFonts font files compiled by Virkus is collected for text-based design generation [49]. The dataset is packaged with a CSV file of the metadata, listing the `filename`, `base_font_name`, `file_format`, `creator`, `category` and `theme`. This is loaded into an SQL database.

The `file_format` is one of two categories: `truetype` and `opentype`. OpenType fonts are an extension of TrueType fonts, supporting advanced features for enhanced font styles.

The category and theme columns are text tags that give insight into the artistic style of the font. There are a total of 61 themes, including `Army`, `Trash` and `Western`. There 8 categories: `Basic`, `Dingbats`, `Fancy`, `Foreign_look`, `Gothic`, `Holiday`, `Script` and `Techno`. `Dingbats` fonts use symbols and shapes as opposed to characters, making them unsuitable for text-based designs. It should also be noted that not all fonts support every character. For instance, some fonts do not include lowercase characters and therefore are not rendered in the SVG output.

3.1.3 Feature Extraction Pipeline

Following the collection of the desired raw data, a separate pipeline is developed to enrich the dataset with high-quality, high-level features. These features are used to build a richer understanding of the designs and use these insights as input for the design generation stage. While a wide range of features could potentially be extracted, the pipeline focuses on colour-related information. Extraction of semantic information is performed at a later stage.

Design Isolation

Because the clothing image data from online marketplaces contain noise (e.g., backgrounds and human models), a print design segmentation algorithm is required to facilitate targeted analysis of print design graphics. This involves finding a bounding box to crop the marketplace image to a pure print design region. This is a challenging task - print designs are diverse in their appearance and marketplace images can vary widely in arrangement. This motivates the experimentation of data-driven, procedural and machine learning based approaches to maximise the accuracy of

this task - which is especially crucial as the accuracy of the remaining feature extraction tasks depend on this.

Colour Analysis

The colour feature extraction process makes heavy use of the ColorHunt colour palette dataset. The textual tags that label the palettes describe the colours present, as well as associated themes and styles of their combination. Moreover, the colour palettes are curated by experts and popularity is recorded with the like count, indicating the perceived aesthetic quality of colour palettes.

This motivates the use of machine learning methods, such as K-means clustering to extract print design palettes and K-Nearest Neighbours for associating the labels of ColorHunt colour palettes in a multi-label classification algorithm. Moreover, by associating the like data, this can be used to partially evaluate aesthetic quality, which is important for producing a fitness function in an EA framework.

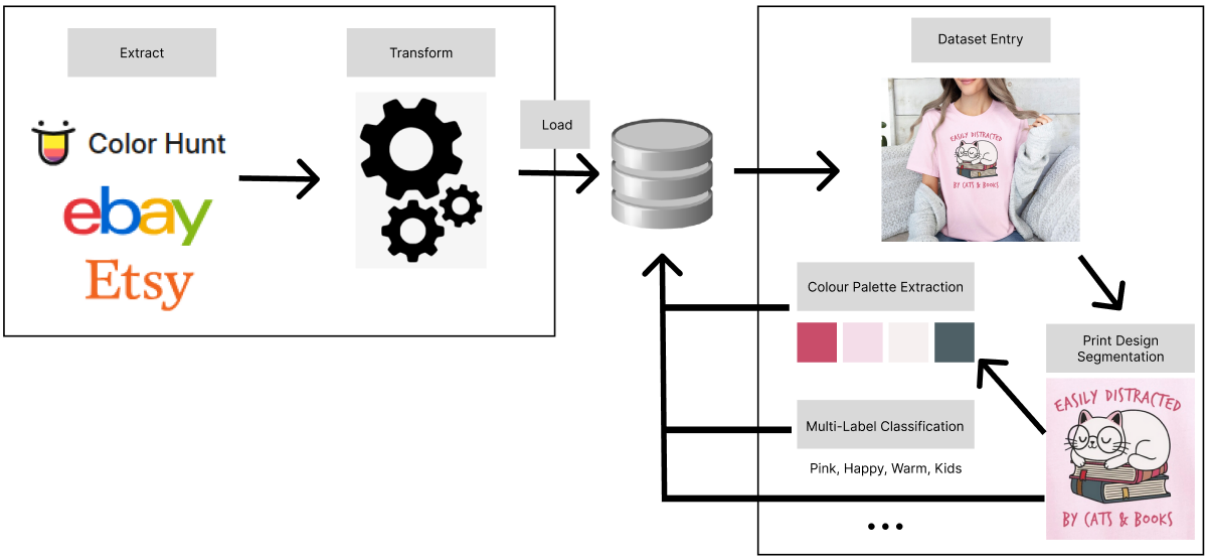

Figure 3.2: Asynchronous ETL and feature extraction pipeline data flow

3.2 Design Generation

In order to build print designs, a framework for the generation and composition of artistic components is developed.

3.2.1 Design Format

When designers use software like GIMP or Adobe Illustrator and save progress with the intent of continuing editing later, rather than exporting the file as the final output type (e.g. a `.PNG` or `.SVG`), an intermediate format is used. These are `.XCF` and `.AI` files for GIMP and Adobe Illustrator respectively. These formats preserve all the image editing information, such as layer groups and transparency, enabling designers to be able to return to editing later. Clearly, this provides convenience for editing, compared to if the graphical components were rasterised and pasted directly onto the canvas on saving.

Although this project is concerned with automated design generation, a format that enables the rearrangement of distinct visual components is preferable. The generated output may require further editing to enhance or make viable. Text in a design may need to be aligned better, which would simply be a matter of modifying the numerical position parameters of the text component

while leaving other content untouched. Generated images may need patchwork edits to clean up AI artifacts.

These considerations point to SVG as the appropriate target output format, as each component can be manipulated independently with the tree-node structure. To clarify, SVG supports the embedding of raster images produced by generative AI models for a hybrid approach, so it is not necessary for all the internal components within the SVG designs to be vector format. While vectorisation methods have been explored, the choice not to vectorise embedded raster images is made due to its lack of utility in helping designers modify the graphics - the output SVG is typically too complex. Moreover, the existence of up-scaling models for increasing the resolution of raster components is sufficient to produce high-quality prints, if necessary.

3.2.2 AST Architecture

Producing SVG markup is a code generation task. Rather than directly building the SVG, an internal representation is used to abstract away the syntax specific to SVG. This is inspired by the frameworks provided by GIMP and Inkscape that support automated image editing through scripts, as well as concepts seen in compilers for the decoupling of representations.

AST node data structures are developed to represent the components of print design content at every level. SVG elements contain other elements, such as images and text, in a tree-like structure motivating the mirroring of this structure in the internal representation. The hierarchical structure of trees naturally supports grouping components together, which mimics the layer feature seen in existing image manipulation software. Moreover, each node can have distinct properties which represent position, size and colour to name a few. A visual output is produced from a given AST root node when compiled into SVG markup.

Template Structures

Motivated by Canva's range of design templates to provide structure and inspiration for designers in the creative process, template structures are developed using the AST nodes. Factory classes take in content as parameters (e.g. images, text, font, colours) to output an SVG design with a fixed structure, as demonstrated in Figure 3.3. The design output is not expected to be well-aligned and market ready. Rather, it is used to enable experimentation with arrangements, or to pass the output to style transfer models for better control of the visual output structure.

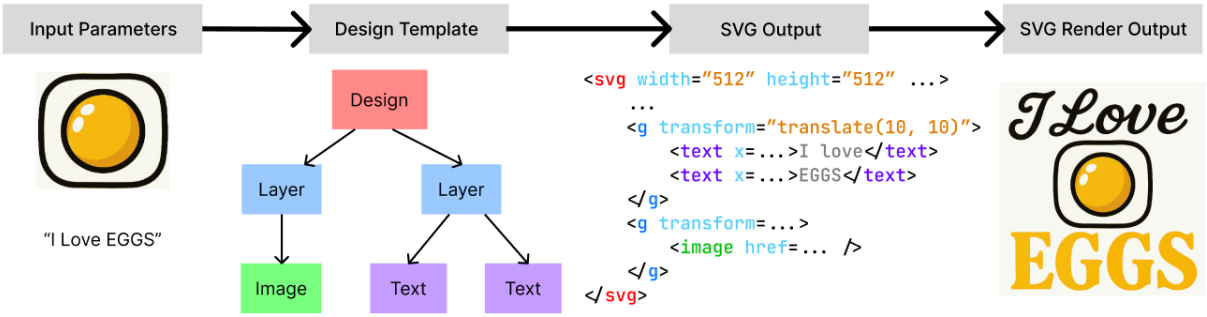

Figure 3.3: Data flow from image and text input to an SVG format print design, using a template.

3.2.3 Generative AI

To automate the creation of graphical elements, generative AI models are used as these models have recently become performant enough for their use to be considered in practical applications. Rather than limit the software developed in this project to a specific model, the modular architecture enables models to be interchanged under a unified interface. Due to the significant

training time, data and hardware requirements, paired with the practical availability of high-quality, off-the-shelf models, a conscious decision is made not to proceed with developing in-house models. The main challenge addressed is in deciding the content of the designs, emphasising the importance of a robust data pipeline for high-level feature extraction and design evaluation for iteration.

The generation of image components relies on generative AI models that use textual prompts as input. At the time of implementation, supported models include DALL-E3 (via OpenAI API) [38] and the Stable Diffusion v1.5 model for local inference (via Hugging Face). Moreover, text generation is necessary for the creation of text-based designs and refining prompts. To achieve this, LLMs such as the DeepSeek-R1:free (via OpenRouter) and OpenAI's GPT-4.1 models are employed [50].

Image Component Generation Pipeline

The utility of the multi-label classification task in the colour analysis section is made clear by the requirement of text prompts as input for generative AI models. Depending on the model, natural language tags are more reliable than hex strings for producing the colours desired. Figure 3.4 shows how the proposed pipeline uses these tags and title data to generate prompts, which in turn generate graphics inspired by an existing T-shirt design. This acts as a mutation operation in an EA framework.

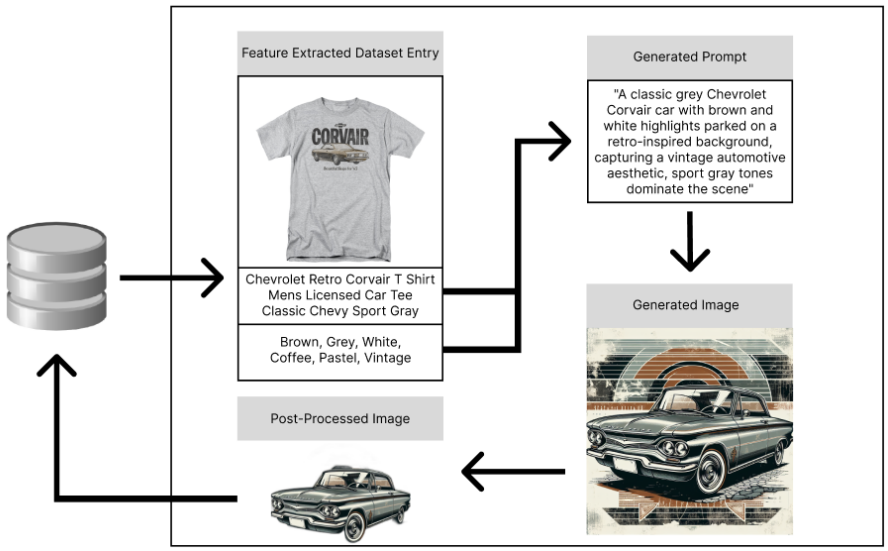

Figure 3.4: Design generation pipeline.

3.3 Environment and Dependencies

Python 3.12 is the language of choice for development. This is appropriate due to the vast ecosystem of powerful, well-documented libraries available for each task outlined.

Key Libraries

- Data processing: `numpy`, `pandas`, `scipy`, `lxml`, `sqlite3`, `requests`, `selenium`
- Image processing: `PIL`, `opencv`, `rembg`, `cairosvg`, `pyciede2000`
- Machine learning: `torch`, `sklearn`, `hugging_face_transformers`, `openai`, `diffusers`
- Visualisation: `matplotlib`

Development Environment

A virtual environment is used for the self-containment of the dependencies needed for the project as listed in `requirements.txt`. Dependencies are managed using `pip`, the package installer for Python. A `.env` file containing environment variables is used to store sensitive API keys and various configuration parameters without hardcoding them into the source code; these are made available at runtime using the `dotenv` module. Git is used for version control. VSCode is the IDE used for development. An NVidia GeForce RTX 3060Ti GPU with 8GB VRAM is used for model inference and training.

Software Testing

The system is subject to some automated unit and integration tests to ensure correctness of individual components and their interactions. This ranges from type checking to output verification. Testing is conducted through the `pytest` library.

Due to the visual nature of many of the tasks, it is difficult to automate the testing of the correctness of some behaviour. Instead, scripts are developed to manually verify visual output. In particular for SVG design rendering, visual output is manually validated by rendering content in a Firefox web browser. Firefox in particular is chosen, as other SVG renderers experimented with appear to be limited in their ability to render embedded content.

Chapter 4

Data Pipeline

The data pipeline serves as the foundation for producing aesthetic, data-driven designs. The goal of this chapter is to outline the technical implementation details for each stage of the data pipeline. First, the individual feature extraction tasks are detailed. This is followed by the details of their integration in the pipeline.

4.1 Print Design Extraction

Since the T-shirt images from online marketplaces typically include human models, backgrounds, or other irrelevant apparel, extracting just the print design enables more targeted analysis downstream. This section outlines the approaches explored to perform this data cleaning task.

To summarise, 4 strategies to produce bounding boxes of the print design region are explored. These are a transformer-based segmentation using a pre-trained model, a procedural approach inspired by decision trees and entropy minimisation, a procedural approach based on contours, and a deep learning segmentation approach using a UNet model architecture trained on a synthetic dataset.

4.1.1 Transformer-based Segmentation: SegformerB3

The first approach uses the SegformerB3 Clothes model, a pre-trained model available on the Hugging Face Model Hub [51]. It is a transformer-based semantic segmentation model fine-tuned for garment segmentation [52]. The model classifies each pixel of an input into one of 18 clothing categories, but for the purposes of this project, only the "Upper-clothes" label is of interest, which encompasses T-shirts as well as other tops such as jumpers and blouses. The model achieves an accuracy of 0.87 and an Intersection-over-Union (IoU) score of 0.78 for this class, which indicates that the model can reliably segment T-shirts in most images.

However, the mask output alone is insufficient for print design isolation. Initial experiments demonstrate that the model segments T-shirts while simultaneously unmasking the print design region at times, underlining the complexity of print design segmentation. Moreover, the storage of binary masks of a large dataset of images does not scale as well as bounding boxes, which motivates the proposed algorithm.

Algorithm

1. Define the hyper-parameters `width_cut`, `top_cut`, `bottom_cut`.
2. Run the Segformer B3 Clothes model to produce a segmentation mask corresponding to the "Upper-clothes" label.
3. Find the minimal axis-aligned bounding box that contains this segmented region. This is a task of finding the minimal and maximal x and y coordinates of the mask pixels.

4. The bounding box is cropped according to the hyper-parameters provided. This is necessary to prune excess segmentation such as the sleeves and collar. In the case no segmentation occurs, the bounding box covers the entire image.

4.1.2 Entropy Bounding Box Detection

The following proposed approach is an experimental, procedural method inspired by ideas from information theory. Before describing the algorithm, the motivation behind this approach is discussed.

Motivation

Decision trees are a machine learning algorithm for classification tasks. The principle behind them is to split datasets on their attributes such that the resulting subsets are maximally homogeneous, as quantified using Shannon Entropy [53]. More formally, the Shannon Entropy $H(X)$ calculates the impurity of a set X, where I indexes the set of classes that X may contain, and p_i denotes the proportion of the population comprised by class i:

$$H(X) = -\sum_{i \in I} p_i \log_2 p_i$$

An optimal split maximises the purity of the resulting subsets. If a dataset is split into n subsets X_i, $i = 1 \dots n$ on some feature Y, the conditional entropy $H(X|Y)$ measures this by taking the weighted average entropies of the subsets:

$$H(X|Y) = \sum_{i=1}^{n} \frac{|X_i|}{|X|} H(X_i)$$

Assuming print designs are sufficiently visually complex, the print design region likely has significantly different visual features compared to the rest of the image. The segmentation algorithm is developed with the hypothesis that the desired segmentation region overlaps well with an optimal entropy-based partition.

Algorithm

1. Hyper-parameters for the algorithm are defined:

   - `grid_size`: The dimensions of the grid that the image is partitioned into, determining the granularity of the bounding box search. The rectangles in the grid are the features Y to split on.
   - `num_clusters`: K-means clustering hyper-parameter for grouping pixel colours. This value determines the number of classes entropy is minimised for.

2. K-means clustering of the image is conducted to minimise noise and group related pixels.

3. The image pixels are divided into a (`grid_size`, `grid_size`) grid of pixels. The counts of each K-means clustering label are tabulated in a (`grid_size`, `grid_size`, `num_clusters`) shape `numpy.array`.

4. A subset of the rectangles contained in this grid define the candidate bounding box regions. The bounding box partition that yields the minimal conditional entropy is chosen.

4.1.3 Contour Bounding Box Detection

Print designs are generally placed on plain coloured T-shirts. Observing the edge data of online marketplace images, the outline of the T-shirt is often away from the image boundary and the print design edges appear as a complex cluster within the T-shirt edge boundary. This motivates the use of a procedural algorithm using contours of edge data likely to correspond to a print design, making use of the OpenCV library [54].

Algorithm

1. Hyper-parameters for the algorithm are defined as follows:
    - `size`: The dimensions the image is scaled down to, measured in pixels.
    - `centrality`: The proportion of the central region that the image the bounding box must be contained in. A float value in $(0, 1]$.
    - `min_w`/`min_h`: The minimum width/height of the bounding box as a proportion of `size`. A float value in $(0, 1]$.
2. As a preprocessing step, the image is scaled down to (`size`,`size`), converted to greyscale and Canny edge detection is applied [15]. The lower and upper thresholds for Hysteresis Thresholding are fixed at 50 and 150 respectively.
3. Contours are detected using the `cv2.findContours` function and sorted in reverse order of area, as calculated using the `cv2.contourArea` function.
4. The contours are mapped to a rectangular bounding box (x, y, w, h) using the `cv2.boundingRect` function.
5. The candidate bounding boxes are filtered to satisfy conditions which relate to the likelihood the image corresponds to a print design region. These are based on the provided hyper-parameters:
    - If the bounding box width and height are too small, it is rejected.
    - If the bounding box is not contained in a central region defined using the `centrality` hyper-parameter, it is rejected.
6. The largest remaining bounding box is selected and rescaled to the original image dimensions. If none exist, the full image dimensions define the bounding box.

The size reduction is useful for two reasons: first, a smaller image is faster to process, which is important for the scalability of the feature extraction process on a large database; second, only large scale edge features such as the T-shirt and print design boundaries are important. Initial experimentation with other approaches, such as using Difference of Gaussian (DoG) filtering for large-scale edge detection, appear to yield poorer results. However, scaling down the image causes the loss of precision of the contour boundary when rescaled back up to the original size. The rescaled contour boundary could be used for region proposal and refined further, but this optimisation is not employed.

4.1.4 UNet Segmentation

A more sophisticated alternative explored is a UNet-based segmentation model. UNet is a convolutional neural network architecture widely used for segmentation tasks due to its ability to preserve feature information at all scales [20]. This motivates its use for T-shirt design segmentation in particular, as we wish to segment a design area from within a T-shirt area of a uniform colour.

Synthetic Dataset

Training a UNet model to detect print designs requires a dataset consisting of the T-shirts with ground truth segmented regions labelled. Because manually segmenting print designs is a labour-intensive task, a scalable alternative approach is experimented with by creating a synthetic dataset:

1. First, a dataset consisting of 70 plain T-shirt images with manually annotated bounding boxes is created. These boxes define the region where a print design is typically located, and are defined by their top-left coordinate, width and height, a list of four integer values.

2. Print design images are sourced from online, generated by rendering graphic fonts with the PIL library, and Stable Diffusion v1.5 model generated images. These designs are made with a transparent background for conversion into a binary mask.

3. Synthetic mock-up images are created by transforming and pasting the designs inside the labelled bounding box region of the plain T-shirts. To increase the diversity and number of images in the dataset and encourage generalisation, data augmentations like reflections are included.

4. Each synthetic mock-up image is paired with the transformed binary mask associated with the print design placement, concluding the creation of the synthetic dataset.

An approach using a lazily generated dataset is attempted, producing new images and corresponding masks for training as they are requested during training. However, the generation of images is extremely slow, so this approach is not deployed.

Training

The UNet model is trained using the PyTorch library [55]. A total of 200 images are reserved for testing and the remainder are used for training. Some hyper-parameters are listed in Table 4.1.

Hyper-parameter	**Value**
Iterations	10000
Batch Size	16
Loss Function	Binary Cross-Entropy
Optimiser	Adam
Learning Rate	1×10^{-4}
Kernel Size	3
Activation	ReLU

Table 4.1: UNet model hyper-parameters

Preliminary results show high training accuracy, with the model successfully capable of segmenting designs seen in training. However, the model over-fits, failing to segment unseen prints, and is therefore not suitable for deployment at this stage. This could be due to a number of reasons: the training set contains a small number of print designs; the synthetic nature of the dataset does not generalise well to non-synthetic designs; or, hyper-parameters such as iterations are non-optimally selected and early stopping is required. In any case, since further optimisation of the print design isolation task is of low priority, the significant overhead required for procuring a dataset, training and hyper-parameter tuning is not considered further.

4.2 Colour Analysis

This section delineates the methods used to analyse print design colours. First, a palette is extracted from the print design using clustering. This palette is then compared against palettes in the ColorHunt corpus and the nearest palette metadata is associated to the print design.

4.2.1 Palette Extraction

The first step consists of extracting a colour palette of the four most predominant colours of the print design. This task is not as simple as getting the four most frequent colours from the raw image; gradients, lighting conditions and noise make that method unlikely to capture a sufficiently accurate representation of the range of colours in the design. A random sampling method could be effective, however it is susceptible to producing duplicate colours in the case colours are unevenly distributed. Instead, clustering algorithms provide a more robust method for identifying the colour palettes, as the colours obtained are prominent, yet distinct.

K-Means Clustering

K-means clustering is an unsupervised machine learning algorithm that can automatically partition the colour space of the designs into K discrete clusters. It achieves this using an iterative method, as outlined in Algorithm 1.

Algorithm 1 K-Means Clustering Algorithm

Initialise K centroid colours
repeat
  Assign each pixel to the nearest centroid.
  Update each centroid as the mean colour of all assigned pixels.
until All centroids are sufficiently stationary.
return Centroid colours

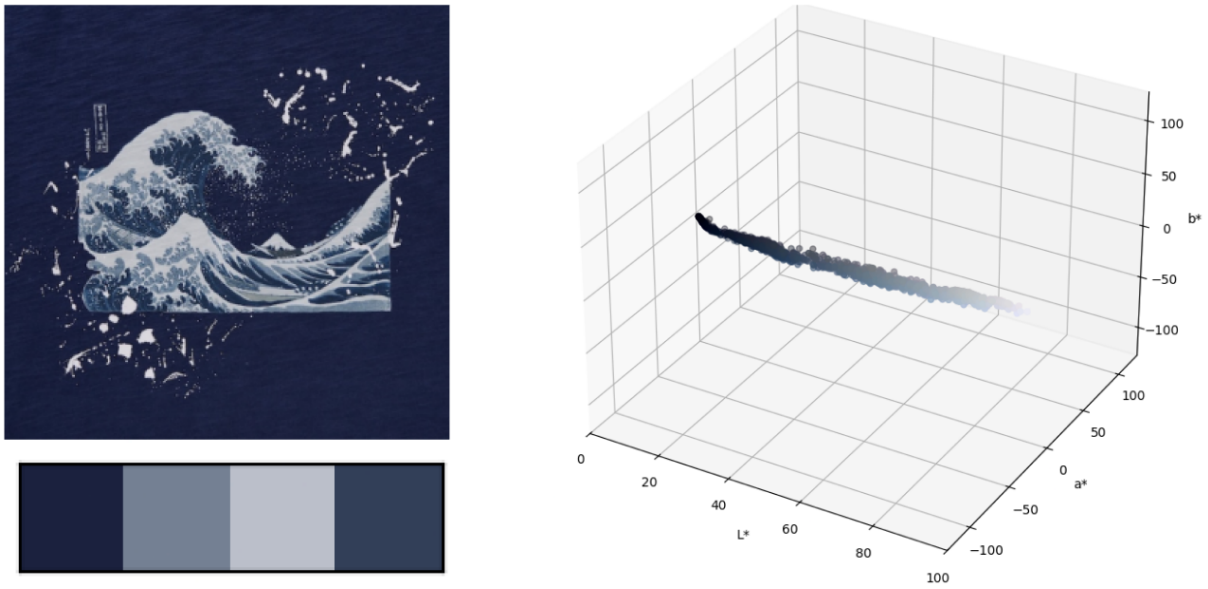

Figure 4.1: K-means clustering results with $K = 4$ of an ocean wave print design.

Clustering is implemented using the Scikit-Learn library `sklearn.cluster.KMeans` function [56]. To be precise, it uses a variant called called K-means++, which chooses the initial centroid

positions randomly from the colours attained by the print design, which minimises the likelihood of a degenerate solution.

Usually for clustering tasks, K is a hyper-parameter that is chosen using a heuristic such as the elbow method. However, taking $K = 4$, the resultant cluster centroids sufficiently represent a colour palette of the four most dominant colours of the print design and is ready for the palette matching procedure that follows.

4.2.2 Multi-Label Classification

Next, the multi-label classification task is responsible for selecting the tags to associate to the extracted palette. Naturally, palettes that appear to be similar should result in similar tags being assigned. This motivates the approach to multi-label classification with K-Nearest Neighbours.

Distance Metric

To identify the nearest palettes, a concrete notion of similarity requires defining. Similarity is quantified using a distance metric, where a lower distance score translates to higher similarity between palettes. To take into account the arbitrary ordering of colours in a palette, the palette distance $d_{Palette}$ is proposed as the solution to the combinatorial optimisation problem given by Equation 4.1, where:

- P and Q denote the two palettes of four colours being compared. p_i and q_j correspond to CIELAB colour space vector representations of a colour in the palette.
- $F = \{f\}_{ij} \in S_4$ is the decision variable permutation matrix that defines the optimal pairings between the palettes, also known as the flow matrix. Multiplying the flow matrix F with the palette matrix Q rearranges the colours in Q to align with its the pair in P, which is useful for visualisation purposes, as demonstrated in Figure 2.1.
- d_{Colour} is a colour difference formula. The `pyciede2000` library's implementation of the CIEDE2000 formula is used in the CIELAB colour space implementation [23].

$$
\begin{aligned}
d_{Palette}(P,Q) = \quad & \text{minimise}_F \quad \textstyle\sum_{i=1}^{4}\sum_{j=1}^{4} f_{ij}\cdot d_{Colour}(p_i,q_j) & \\
& \text{subject to} \quad \textstyle\sum_j f_{ij} = 1 \quad && \text{for } i \in \{1,2,3,4\} \\
& \textstyle\sum_i f_{ij} = 1 && \text{for } j \in \{1,2,3,4\} \\
& \textstyle\sum_{i,j} f_{ij} = 4 & & \\
& f_{ij} \in \{0,1\} && \text{for } i,j \in \{1,2,3,4\}
\end{aligned}
\tag{4.1}
$$

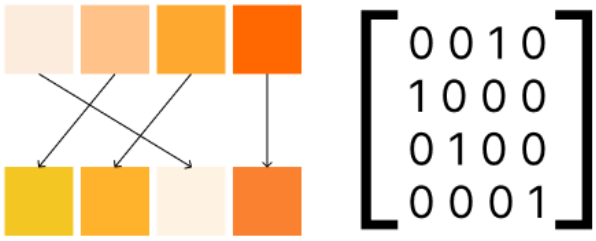

Figure 4.2: Depiction of optimal colour pairings between the palettes. The resultant adjacency matrix is F, where f_{ij} is activated if and only if there is a pairing between the first palette's i^{th} colour and the second palette's j^{th} colour.

The formulation of this problem is based on the Earth Mover's Distance metric proposed by Rubner et al [57]. To compute this distance, the `scipy.optimise.linear_sum_assignment` function is used, which solves (4.1) using the Hungarian Algorithm [58] [59].

K-Nearest Neighbours (KNN)

KNN is a non-parametric, supervised machine learning algorithm that performs classification based on the the K most similar entries in the labelled training set. With the distance metric between colour palettes established, KNN classification is employed to find the dataset palettes with greatest similarity to the design palette. The most common textual tags associated with the K nearest palettes are then associated to the print design palette. For more accurate results, hyper-parameter tuning would be conducted to define K for each label independently. Since there are 49 labels for this task, $K = 1$ is chosen for simplicity.

4.3 Palette Popularity

The ColorHunt dataset encodes aesthetic quality of a palette by recording like count data. This section analyses this data for the purposes of devising a fitness function for print design aesthetic quality. This is motivated by the fact like count is a human-centric metric of visual appeal and therefore may correlate with the aesthetic appeal of print designs.

Recency Bias Correction

Analysing the plot Figure 4.3 of log-like count against submission number, it is clear that the two quantities are associated. The initial linear trend in the log-linear plot suggests exponential growth in the like count over time. This may be attributed to the growth in website traffic. However, at the tail of the graph, there is a sharp drop in likes observed in recent submissions, likely due to insufficient time for users to interact with the content, as opposed to a drop in quality or website traffic.

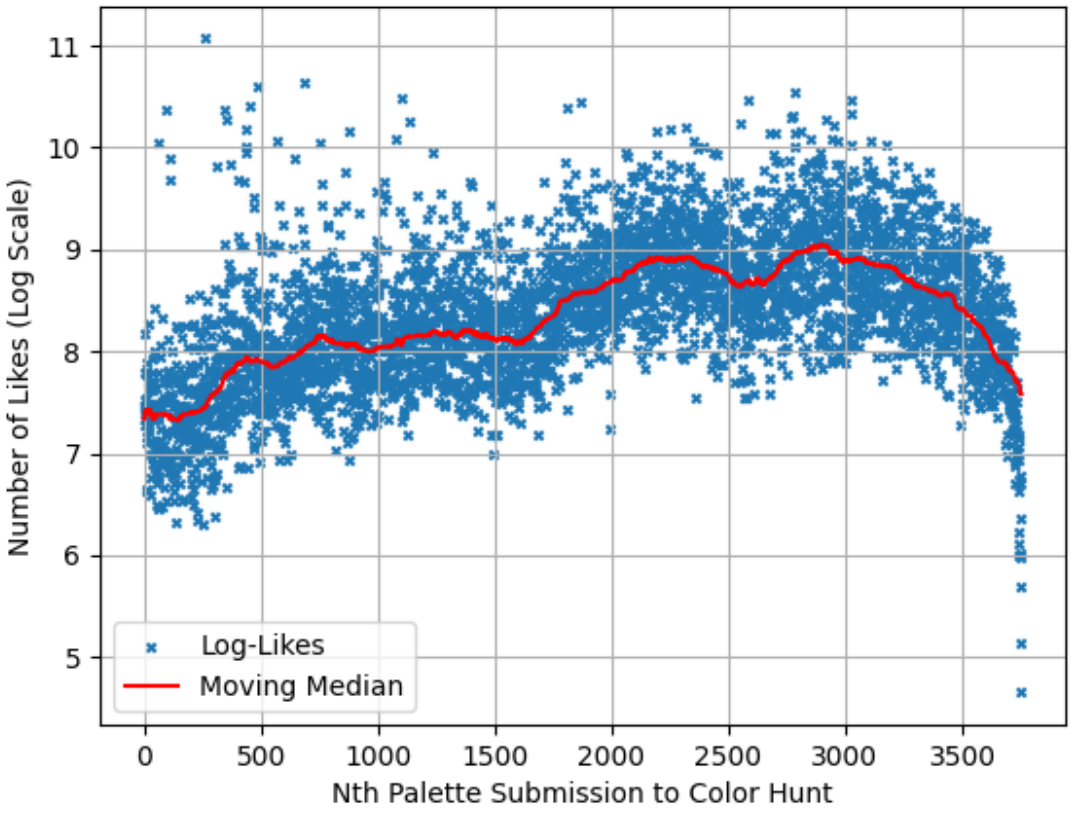

Figure 4.3: Log-linear scatter plot of palette like count against submission number to ColorHunt.

To more accurately quantify colour palette popularity using like count, the data is corrected to take into account these covariates. To achieve this, first a curve is fit to the graph. Attempts to do this include:

- **Moving Median:** A moving median in a window of size 200. The median is chosen instead of the mean as it is more robust to outliers.
- **Custom Model:** An attempt to build a model is made using the interpretations of exponential growth and saturation with $\hat{y} = \theta_0 e^{\theta_1 x}(1 - e^{\theta_2(x-\theta_3)})$. The global optima for the non-linear least squares problem can be approximated using the Gauss-Newton method.
- **Polynomial Regression:** This is a smooth curve fitting approach that minimises residual between points and the curve by solving a convex optimisation problem, though the parameters are not as interpretable.

Figure 4.3 demonstrates that the moving median plot fits the graph well. By comparing the like count to the median in the interval, this model makes the assumption that factors such as website traffic do not have a significant influence on the like count relative to others within a window of 200 submissions. This allows the likes of palettes within a similar time interval to be compared, producing a better measure of relative popularity that is independent of time.

To compute how popular a palette is, the like count is compared relative to the median in the 200 submission interval:

$$\text{Adjusted Popularity Score} = \frac{y_i}{\hat{y}_i}$$

y_i denotes the like count of palette submission i, and $\hat{y}_i$ denotes the median centred at submission i with window size 200. This results in a score in the range $[0, \infty)$. This is to be incorporated into a fitness function that takes into account colour palette popularity for aesthetic scoring.

4.4 Pipeline Implementation

With each stage of the pipeline addressed in detail, this section delineates the technical details involved in obtaining and processing the data collected and the integration of the pipeline. As discussed in the system overview, there are two components to the data pipeline. The first stage is an ETL pipeline focused on sourcing the raw data required for market research. This is followed by a feature extraction pipeline to enrich the data further.

4.4.1 ETL Pipeline

Extract

The extract phase is responsible for collecting data from different sources, through web scraping or APIs. The implementation allows for selective activation of different extractors based on the provision of boolean flags. Four data sources can be extracted from: ColorHunt, eBay Browse API, eBay Seller Hub, and Etsy listings.

- **ColorHunt:** The `ColorHuntPageScraper` class downloads HTML from the ColorHunt website using a Selenium Chrome web driver, due to the need for scrolling to retrieve the dynamically loaded older palettes. Once the initial page is downloaded with links to each palette present in the HTML, the scraper accesses each individual palette page to obtain additional metadata.
- **eBay Seller Hub:** The `EbayPageScraper` extracts product data from locally saved Seller Hub HTML pages. The `lxml` library is used to parse the HTML using XPath expressions.

- **Etsy Listings:** Similarly to the eBay Seller Hub extractor, the `EtsyPageScraper` scrapes locally stored Etsy HTML pages.
- **eBay Browse API:** Requests are made to the eBay Browse API `v1/item_summary/search` endpoint to retrieve live listings of printed T-shirts. To narrow search results to those of greater relevance and quality, the search results are filtered by price range ($10-50), buying options (fixed price), account type (business) and condition (new).

Each scraper inherits from the `PageScraper` abstract base class, which defines an interface for scraping data from a directory of static HTML files to a `pandas.DataFrame`. Reusability of scraping logic is ensured through the use of the `scrape_directory_to_dataframe` method across all subclasses to compile datasets from saved web pages.

Transform

Following extraction, the data is passed to the `transform_data` function for organising the data into a standardised schema.

- **ColorHunt:** Colour palette values, colour tags and other metadata are separated into normalised tables. Duplicate tags are removed, such as those detected in Figure 4.4. This highlights the improvements made to data integrity in the transformation stage.
- **Clothing Data:** Clothing data from eBay and Etsy are merged. To do this, a `source` column is added to trace the item origin, and columns that are not shared are dropped. Data imputation methods are not considered at this time. The remaining column names from each source are standardised which permits the union of the clothing data.

Figure 4.4: The duplicate “Orange" tag in palette ffedc6ffc478fff87d724330.

Load

The lightweight SQL database engine `sqlite3` is used, as it is suitable for relatively small-scale datasets and requires minimal setup overhead.

The `load_data` function stores the transformed `DataFrame` objects into a structured relational database. This is achieved using a series of SQL `INSERT` statements. While most insertions are configured to ignore conflicts, an exception is made for the ColorHunt palette data to update the like count of existing records. This is particularly important for newer palettes, which initially have significantly lower like counts due to recency bias.

The database schema is normalised in 3rd Normal Form (3NF), resulting in improved query speed, reductions in redundancy and improved data integrity. The relational structure is illustrated in the Entity-Relationship (ER) schema shown in Figure 4.5.

4.4.2 Feature Extraction Pipeline

A separate feature extraction pipeline is responsible for enriching the dataset for the downstream design generation task. Feature extraction is a significantly more intensive task compared to the ETL process, due to the overhead in IO-bound image retrieval from URLs and subsequent processing. To prevent redundancy of computation, the feature extraction pipeline begins by

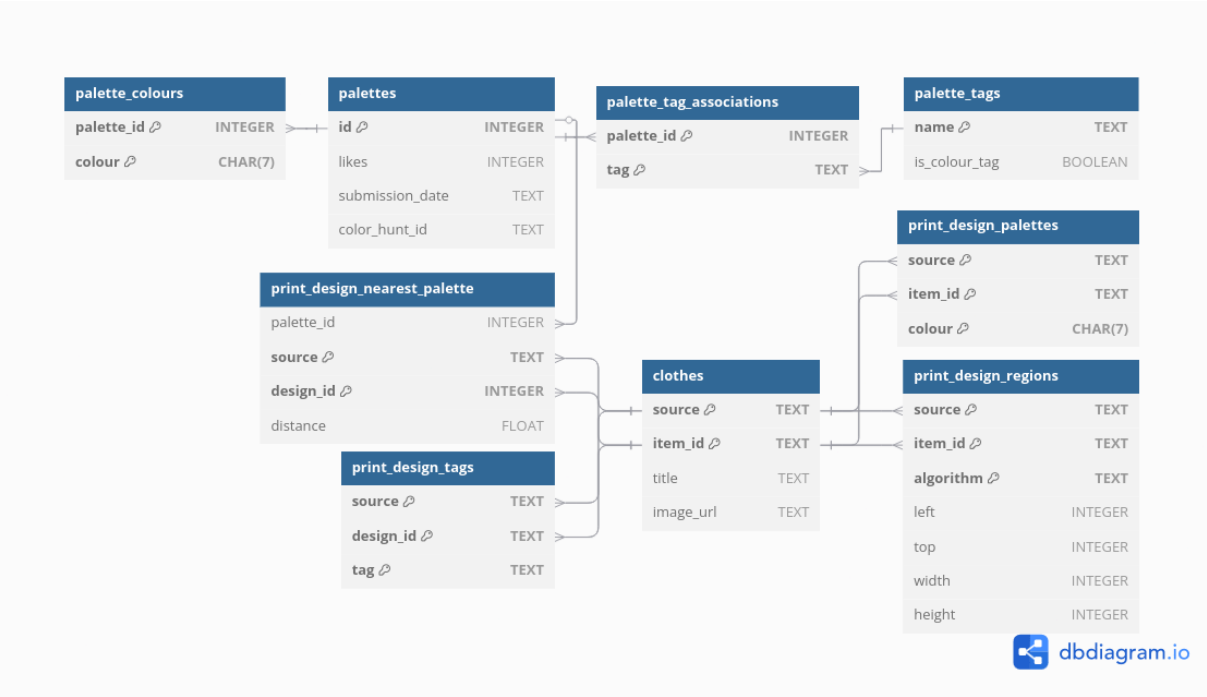

Figure 4.5: The SQL database schema.

querying the database for entries lacking extracted features. It then extracts and writes these features to the database sequentially.

The first feature extracted is the bounding box of the print design. The predicted bounding box is stored in the `print_design_regions` table with the name of the algorithm that generated it. This is so that other segmentation algorithm results - and even ground truth labels for evaluation - can be recorded in the database. The image is then cropped according to the bounding box and used for feature extraction in the remainder of the pipeline.

The CIELAB colour space multi-label classifier is run to conduct clustering and nearest neighbour search of colour-related information:

- The colour palette extracted from the print design is converted from integer tuples into hex strings and stored in the `print_design_palettes` table.
- The predicted tags are added to the `print_design_tags` table.
- The `print_design_nearest_palette` table keeps a reference to the nearest palette and the $d_{Palette}$ distance for evaluation purposes.

4.4.3 Scheduling

Motivated by the need to keep up-to-date with recent market trends, both the ETL and feature extraction pipelines are autonomously run on regular intervals to continuously populate the dataset. This is achieved using cron, a job-scheduler for Unix-like operating systems. While more sophisticated orchestration tools such as Apache Airflow are available, cron's lightweight and simple setup makes it suitable for the pipeline outlined.

A total of three bash scripts are developed. They are made executable by modifying permissions using the Linux `chmod` command and can be run directly. The purpose of the scripts are as follows:

Figure 4.6: Visualisation of the features extracted for each clothing entry.

- `run_etl.py`: Runs the ETL pipeline. The `argparse` module is used to allow the script user to specify precisely which components of the ETL pipeline to run, with the `--colorhunt`, `--ebay_browse`, `--ebay_seller_hub`, `--etsy` commands options available.
- `run_feature_extract.py`: Runs the feature extraction pipeline. Two command options are supported:
  - `--load` command option conducts feature extraction and loads the data into the database. By default, this does feature extraction on all clothes items that have not yet had features extracted. An optional integer value can be passed to limit the number processed with the `--n [amount]` command option.
  - `--clear` evicts all clothes feature data from the database. This is used for development purposes when feature extraction methods are optimised.
- `setup_cron_jobs.py`: This script schedules the previous two bash scripts to run on a regular basis, in particular on the 0th minute of every hour. Only the `--ebay_browse` command option is active, as API-based scraping is preferable for reasons discussed earlier.

Chapter 5

Design Generation

This chapter begins by presenting an object-oriented framework for representing and creating print designs in SVG format. This is followed by the technical implementation details of generating print graphics using a chain of generative AI models in an evolutionary algorithm framework.

5.1 Internal Representation of Designs

A natural architecture for image editing is to have visual components placed using hierarchical layers on a canvas. To achieve this, design components such as images and text, are represented as nodes within a tree-like data structure. This architecture is ideal for conversion into SVG - a standard image format in graphic design. This has the benefit of maintaining the locations, sizes and contents of visual elements of the design. Moreover, the tree-like data structure improves the ease of modification as elements are grouped appropriately.

5.1.1 AST Node Architecture

Node

The `Node` class is an abstract base class - a blueprint superclass for all other design elements. It defines the following interface, for which subclasses may need to override:

- `to_lxml` returns an `lxml.etree._Element` object with appropriate tags and attributes. This is an intermediate representation of the subclass, used to produce the SVG format XML element. This is an abstract method, requiring implementation in subclasses.
- `to_svg` returns the node's SVG markup representation as a pretty printed string. This is implemented fully in the `Node` class and not overridden by any subclasses, It uses the template method design pattern, calling the abstract `to_lxml` method. When a subclass implements the `to_lxml` method, this immediately allows `to_svg` to produce SVG markup, minimising duplicated logic.
- `get_dependencies` is a helper method used in `to_svg` that specifies the external dependencies required by the node for the SVG to be rendered correctly, such as fonts.
- `to_dict` is an abstract method that serialises the node into a dictionary format, in particular for a JSON representation. It is currently deprecated as SVG is the target output format.

ImageComponent

The `ImageComponent` class represents a raster image in the design, and is a leaf node in the internal representation. It takes the following parameters:

- `image`: A PIL Image object representation of the raster image to be rendered.
- `position`: A position tuple to specify its top-left (x, y) coordinate's offset relative to its parent node.

This class corresponds to an `<image>` SVG element by storing the position as `x` and `y` attributes, and encodes the image into a base64 string to be stored directly in the SVG.

TextComponent

The `TextComponent` class represents the text rendered in the design, another leaf node in the internal representation. The parameters supported are as follows:

- `position`: Defines the coordinates of the text placement similarly to the ImageComponent
- `text`: The actual string to be rendered.
- `font_family`: The name of the font available from the font database. The specific font file is loaded in during the serialisation of the design at a later stage.
- `font_size`: The size the text is rendered in.
- `fill`: The RGB tuple representation of the colour the text is filled in.
- `tag`: Specifies which mode the text is rendered in, either "image" or "text" mode.
  - **Image mode:** Renders the text as a raster graphic element using the `PIL` library. The image is encoded into a base64 and the SVG output follows the same format as the `ImageComponent` node.
  - **Text mode:** The text is embedded in a `<text>` SVG element with the provided attributes. In addition, the "text-anchor" and "dominant-baseline" SVG attributes are hardcoded with the "middle" value for centre alignment. In this mode, a reference to the font family is included in the node's dependencies.

Layer

The `Layer` node is a container to group related elements together. The parameters are as follows:

- `position`: A tuple used to translate all children nodes relative to the parent.
- `components`: A list of `Node` objects, such as `ImageComponent` or even nested `Layer` objects. This structure enables hierarchical grouping, allowing children components to be transformed together. The ordering of components in the list specifies the draw order.

SVG serialisation of this class uses the `<g>` top-level element, with the assigned attribute "transform" attribute set to `"translate(position)"`. Moreover, each child is individually serialised by calling its `to_lxml` function and added as children to the `<g>` element. The dependencies of this node is calculated as the union of all dependencies of its immediate children nodes.

Design

The `Design` node is the root node of the tree. It represents the top-level `<svg>` SVG element, and organises multiple layers into a complete design. The parameters are as follows:

- `canvas_size`: A `(width, height)` tuple defining the design dimensions.
- `layers`: A list of `Layer` objects. The ordering indicates the draw order onto the canvas.

When serialising the `Design` node, dependencies are collected similarly to the `Layer` node, which minimises duplication when serialised in the SVG output. These are stored in a top-level `<defs>` element. In particular, font files are encoded in base64 format and embedded as text in `<style>` sub-elements using the CSS `@font-face` rule. Finally, each layer is serialised and added as children to the tree root, to produce a complete `lxml` intermediate representation of the design. To save a given design, calling the `to_svg` method generates a well-formed, pretty-printed string that can be written directly to an SVG file.

5.1.2 Design Templates

Design templates provide reusable blueprints for constructing commonly-used, visually-coherent print layouts. By leveraging the AST-based internal representation introduced in the previous section, templates abstract away the manual composition of nodes. These templates define arrangements of components such as images and text, enabling the rapid generation of designs with consistent structure, although at the expense of precision and control.

Each template can be parametrised with different input content, such as text strings, images, fonts and colours. This allows the templates to adapt to a range of design contexts without requiring structural changes. Each template function returns a fully constructed `Design` object, populated with appropriately configured `Layer`, `ImageComponent`, and `TextComponent` nodes.

There are many potential templates that could be developed. At the current stage of development, the following are implemented:

- **Captioned Image:** A layout featuring a central image with a single line of text optionally placed above and below. See Figure 5.1
- **Grid Collage:** Arranges multiple images in a uniform grid pattern. See Figure 5.2
- **Multi-line Text:** A simple design featuring centred rows of text.

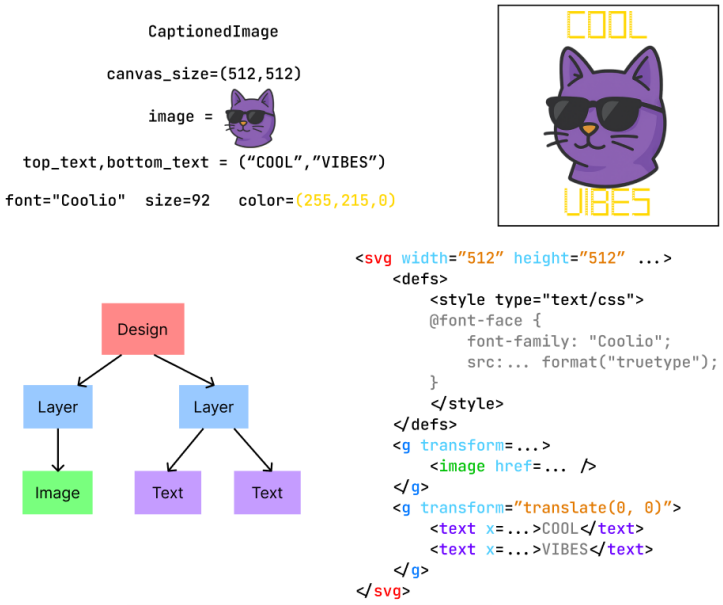

Figure 5.1: The Captioned Image layout, featuring a central image with a single line of text optionally placed above and below.

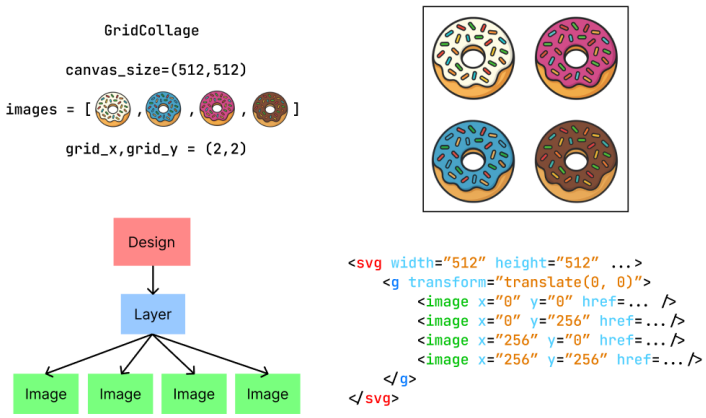

Figure 5.2: The Grid Collage layout arranges multiple images in a uniform grid pattern.

5.2 Component Generation

This section details the creation of design components that are used as input for the internal representation described in the previous section.

5.2.1 Image Component Generation

Prompt Generation

To make use of the insights gathered, such as the tags acquired from the multi-label classification and the design titles, prompts are engineered to combine these so that the descriptive tags modify the nouns inferred from the semantic content of the title. Initial experiments using Term Frequency-Inverse Document Frequency (TF-IDF) for ranking the importance of keywords are conducted and generally successfully extract the most meaningful keywords relating to the semantic content of the print design. However, the strong performance of using LLMs directly for contextual understanding and natural language synthesis makes this a significantly more practical and appealing approach to analyse titles and generate prompts.

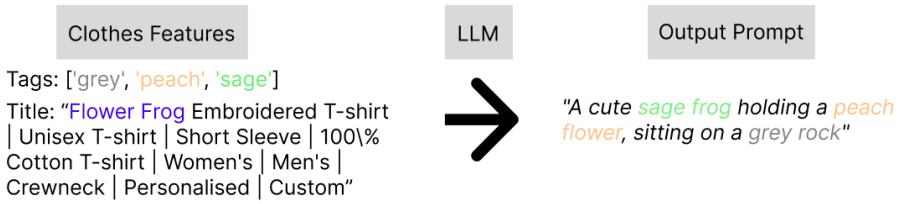

Figure 5.3: Example of prompt generation using features extracted from a clothes listing.

The prompt generated is then padded to instruct the image model to produce a vector-style image with no background, improving the relevance of the output to T-shirt print designs. Previous iterations mention the use case of T-shirt graphics in the prompt, but this leads to the undesirable output of the T-shirt itself being included.

Background Removal

Models that output raster images often do not incorporate alpha transparency for a clear background, which is contrary to the requirements of print designs graphics. Therefore, an image segmentation algorithm to remove background pixels is required as a post-processing step, which is challenging to do accurately and consistently enough to preserve the original image quality. To achieve this, the output image is stripped of its background using `rembg` by Gatis, which uses

a U2Net architecture for salient object detection [60] [61]. Then, the image is cropped to the minimal bounding box containing pixels above a threshold alpha.

Figure 5.4: Example of image background removal using the `rembg` library [60].

Tag Relevance

Initial experiments show that while tags such as `winter`, `coffee` and `halloween` have a perceptual relationship with a group of colours, their utility is limited when tagging generic designs by colour palette. For instance, the inclusion of a `winter` tag typically produces snowflakes regardless of the remaining content of the image. Including the `coffee` tag invariably includes cups of coffee, even if the other features are completely unrelated. These tags therefore detract from the design aesthetic, justifying the retention of only a subset of the assigned tags.

Population

The image components generated are recorded in a database to track relevant metadata, where the schema is outlined in Table 5.1.

Column	Description
`image_path`	Primary key, used to specify the file path of the generated image.
`source`	Foreign key, identifies the parent design from the `clothes` table.
`item_id`	Foreign key, identifies the parent design from the `clothes` table.
`prompt`	The main body of the prompt used to generate the image.
`palette_id`	Foreign key, for the nearest palette in the `palette` table.
`palette_distance`	Distance between extracted and matched palette.
`colour_score`	Used for evaluation of colour relevance.
`prompt_score`	Used for evaluation of prompt relevance.
`aesthetic_score`	Used for evaluation of aesthetic quality.

Table 5.1: Generated image population table for an evolutionary algorithm framework.

The initial population is defined by the collection of marketplace T-shirt print designs. The process outlined for the generation of new graphics based on the extracted features of a given T-shirt acts as a mutation operation in an EA framework. The scores relating to fitness of output designs are human-evaluated scores, where automated fitness calculation is left for future work.

Chapter 6

Evaluation

The proposed design generation pipeline is composed of multiple independent processes, each of which are inherently susceptible to error due to the technical complexity of the tasks. Due to the sequential nature of the pipeline, accumulation of error poses a challenge, where small inaccuracies can be amplified to be significant in the final analysis. This emphasises the importance of not only the evaluation of each component individually, but a holistic assessment of their integration.

With that said, even if the pipeline extracts features accurately and uses the insights appropriately to generate relevant designs, it is still important to determine the efficacy of the data pipeline in practical terms, as the project sets out to generate designs fit for market. To this end, this chapter outlines evaluation at two levels of analysis. First, the methodology and results for the evaluation of each component of the design generation pipeline is presented in isolation. This is followed by an integrated evaluation of the insights and final output designs to determine the efficacy at the system level.

6.1 Print Design Extraction

The task to segment print designs as a cleaning step is vital for the correctness of the insights drawn from the images. This section presents the evaluation of each approach for print design bounding box detection.

6.1.1 Method

Evaluation Metric

To evaluate each T-shirt design extraction algorithm, an evaluation metric is used to measure how well the output bounding box matches the labelled bounding box, by rewarding overlap and punishing excess segmentation. For object detection tasks, Intersection-over-Union (IoU) is a commonly used evaluation metric:

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

IoU is strict in its evaluation of bounding boxes, as it punishes any deviation from the labelled boundary. However, for the purposes of this task, there is typically a large area for which results are not negatively affected if excess is segmented. In particular, the T-shirt fabric area around a print design tends to have a uniform colour, so excess segmentation within the T-shirt area should be measured as an equally acceptable bounding box region. To account for this, a modified version of IoU (MIoU) is proposed, where the region that tolerates excess segmentation is denoted B_{tol}:

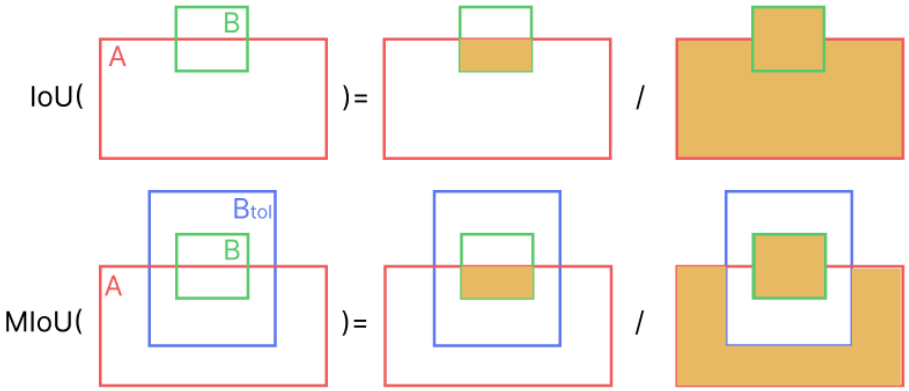

Figure 6.1: Visual depiction of IoU and MIoU metrics.

$$\text{MIoU}(A, B, B_{tol}) = \frac{|A \cap B|}{|(A \cap B_{tol}^c) \cup B|} \quad \text{where } B \subseteq B_{tol}$$

It can be shown that MIoU is a more general version of IoU, with MIoU $\geq$ IoU and equivalence for $B = B_{tol}$. Moreover, MIoU is bounded between 0 and 1, where 0 indicates no overlap with the design region and 1 indicates complete overlap while remaining in the tolerance area. A value significantly less than 1 will pick up too much non-design region, which worsens the results for colour analysis in particular.

Test Set

A test set consisting of 50 real-world T-shirt images with print designs is hand-labelled with two bounding boxes. The first represents the smallest region that covers the entire design, i.e. B. The second represents the largest region that covers the design, while remaining entirely within the T-shirt, i.e. the tolerance zone B_{tol}. Each bounding box is represented with four values: the x and y coordinate of the top-left corner and the width and height.

Hyper-parameter Tuning

Hyper-parameter tuning is performed via grid-search using an isolated, validation set of 20 labelled marketplace images. The separation of the validation set from the test set is important for a fair, accurate assessment - otherwise, the models could be made to over-fit and produce high evaluation scores, yet not generalise and perform as well in practice. The hyper-parameters that achieve the greatest mean MIoU score on the validation set are selected for the model.

6.1.2 Results

Running each algorithm on the test set yields the mean evaluation scores given in Table 6.1, truncated to 3 decimal places. The results suggest that the SegFormerB3 and Contour algorithms perform best with respect to the MIoU metric, and is a significant improvement over no design extraction. The IoU metric is greatest for the Contour algorithm, and is significantly greater than that of the SegformerB3 algorithm. This highlights the importance of the development of the custom metric, as it highlights the SegformerB3 algorithm is still effective for the desired task.

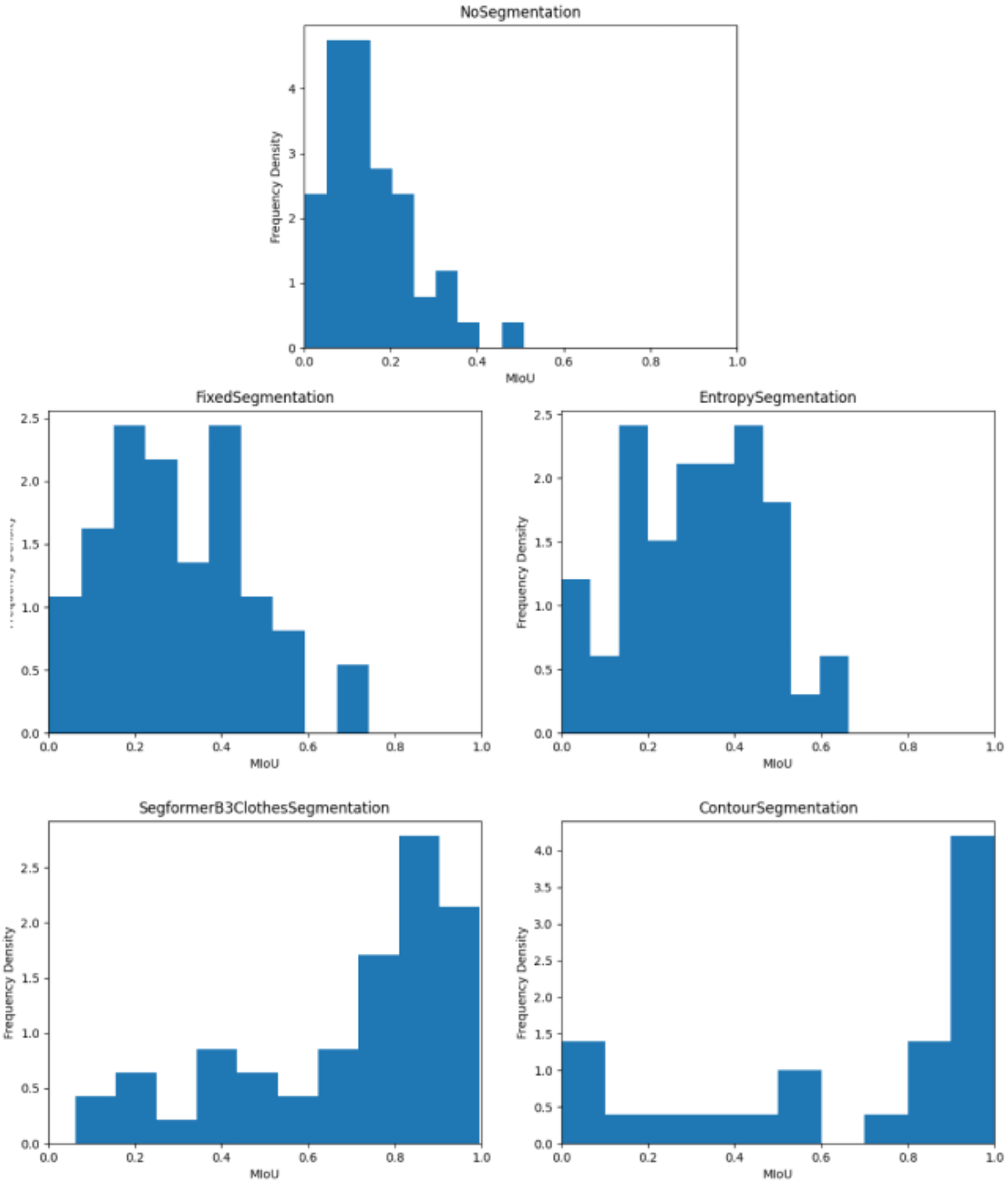

Figure 6.2: Histograms of MIoU scores for each segmentation algorithm.

Algorithm	IoU	MIoU	Time/s
None	0.108	0.150	≈ 0.0
Fixed	0.177	0.291	≈ 0.0
SegformerB3	0.269	0.688	1.927
Entropy	0.200	0.313	1.040
Contour	0.578	0.673	0.014

Table 6.1: Average performance of T-shirt design extraction algorithms with respect to IoU, MIoU and execution time.

Discussion

The SegformerB3 model provides a strong base-line and currently outperforms the other models with respect to the MIoU metric. The model's ability to detect upper clothing specifically makes it robust to noise in comparison to the procedural methods. However, the Contour algorithm on average performs similarly to the SegformerB3 algorithm, which is a significant result given it is a procedural method. In addition, when the model does detect the correct print design region, the bounding box is tight, as indicated by the higher IoU performance compared to the SegformerB3 algorithm. As demonstrated by the histogram, while the Contour approach has a greater rate of perfect segmentation, it also has a greater chance of completely missing the print design. This is expected as the lack of image understanding makes it particularly susceptible to error when multiple entities are contained in the image. Therefore, a hybrid approach could be experimented with to optimise these further. Since the SegformerB3 model is effective at isolating the context to within a T-shirt, following this with the Contour method could be effective.

The experimental entropy-based algorithm yields IoU and MIoU scores of 0.202 and 0.315. These scores suggest that while it appears to perform slightly better than a naive fixed segmentation, the results do not appear to significant for deployment. The algorithm consistently under-segments, and typically stops at the clothes-background boundary. This could potentially be useful as an initial noise removal step for before further segmentation using contours. The computational overhead of computing entropies for each candidate bounding box is significant, with the speed only being half of that of the Segformer approach. This could be optimised using by random uniformly sampling the pixels to reduce the calculations, instead of using the entire image.

Overall, the results suggest that the Contour and SegformerB3 algorithms perform best, though both with varying strengths and weaknesses. The Contour algorithm has a higher perfect segmentation rate and significantly higher IoU. The orders of magnitude increase in execution speed is also significant for the scalability of the feature extraction algorithm on a large dataset. However, the SegformerB3 does perform slightly better on average, with the greater MIoU score and generalisation capabilities. For this reason, the SegformerB3 algorithm is deployed for the remaining tasks, though a strong argument could be made for the deployment of the Contour algorithm instead.

6.2 Colour Analysis

6.2.1 Method

Evaluation of the results of colour analysis is divided into two steps. First the isolated tasks of palette extraction and dataset matching are evaluated. Then, the validity of labels for the print designs is measured for an integrated evaluation of the print design extraction and colour analysis procedures.

Colour Palette Matching

A colour difference of $\Delta E^*_{2000} < 3$ is generally considered to be perceptually identical. The $d_{Palette}$ distance metric used to find the dataset palette nearest to the palette extracted from clustering is calculated as the sum of 4 colour differences, so dividing this value by 4 gives the average colour difference of each pair and could be used to quantify if the matched colours are perceptually equivalent.

However, for the use case of assigning labels which are coarse-grained, an average $\Delta E^*_{2000} < 3$ is too strict. Moreover, the lighting conditions the images are taken in will affect the distance. To gain a better intuition into how well the palettes match, the results from 25 random print designs are visually compared manually. Instead of considering ΔE^*_{2000} scores, a colour is deemed suitable if it visually appears to match the category of colour of the extracted palette from the isolated print design. The following metric is used to quantify the result.

$$\texttt{Accuracy} = \frac{\texttt{Total number of suitable colours}}{\texttt{Total number of colours}}$$

Multi-Label Classification

The multi-label classification algorithm assigns up to 49 tags, as defined in the ColorHunt dataset, to each print design. Since this is a large number of tags, it is infeasible to produce a sufficiently large and diverse enough dataset for accurately evaluating the assignment of each individual tag independently. Instead, evaluation of the multi-label classification algorithm is focused on the model's ability to minimise the occurrence of false positives, which gives an indication into how much error accumulates through the pipeline. In particular, the Micro-Average Precision metric is used, where $\texttt{TP}_i$ and $\texttt{FP}_i$ denote the true and false positive counts for class i:

$$\texttt{Micro-Average Precision} = \frac{\sum_i \texttt{TP}_i}{\sum_i (\texttt{TP}_i + \texttt{FP}_i)}$$

To calculate this, a manual review of the output of the model is conducted on the same 25 palettes to count the occurrences of correctly and incorrect assigned tags.

6.2.2 Results

Colour Palette Matching

From the 584 entries with palettes extracted, the average palette matching score is $d_{Palette} \approx 33.502$ with average $\Delta E^*_{2000} \approx 8.375$. This suggests that the difference is not perceptually negligible, as expected.

	#Entries	**Mean**
Palettes	584	33.502

Table 6.2: Aggregated palette distance metrics, truncated to 3.d.p.

From a random sample of 25 images, an accuracy score of 0.71 is achieved. The accuracy of matched palettes demonstrates that the dataset and proposed $d_{Palette}$ metric are moderately effective, correctly matching approximately 3 of 4 colours on average.

During the manual review, it is clear that print designs with a monotone palette (i.e., black and white shades only) in particular perform poorly. This is because the ColorHunt dataset itself does not contain any monotone palettes. This is exacerbated by the fact monotone palettes are common among the T-shirts sampled. Moreover, shades of black and white are often prominent even in colourful designs, as these occur in line art of the print graphics or as background colours. The clustering algorithm is correct to pick these colours up, but these are not useful for

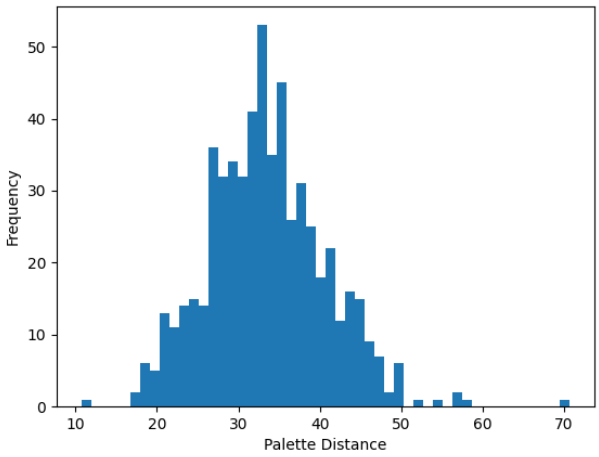

Figure 6.3: Colour Palette Distance Histogram

the labelling task for colourful designs. Therefore, a more sophisticated approach to filter "less important" colours in the clustering approach for classification could be explored in the future. Also, the $d_{Palette}$ metric could be modified to minimise the maximum distance of the paired colours to prevent outlier colours. This would also provide better intuition for the distance value obtained.

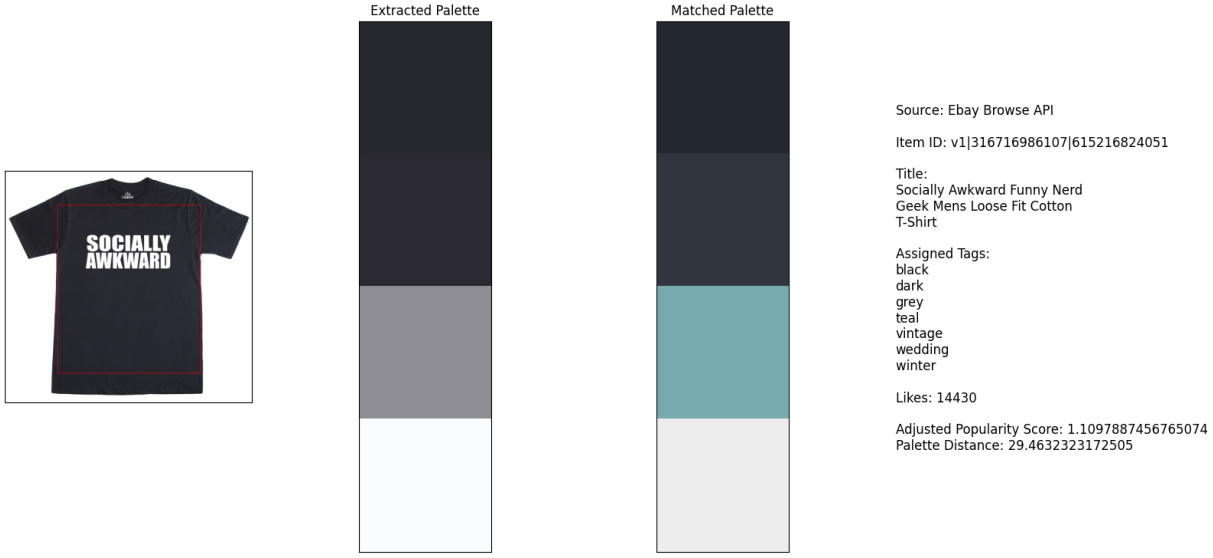

Figure 6.4: Example of a single inaccurate colour in monotone palette matching, $d_{Palette} \approx 30$.

Since colour palettes are added to the ColorHunt dataset only if they meet the aesthetic quality judgement of experts, the large distances may indicate that the colour palette of the design itself is not a selling point. Alternatively, it could be that the colour palette is novel and could be popular if submitted. Regardless, since the aesthetic quality assessment partly depends on the appeal of colour palettes, if the colour palette popularity cannot be reliably evaluated, print designs in the database with $d_{Palette}$ score greater than some threshold can be omitted for generation purposes.

Multi-Label Classification

From the same sample of 25 clothing items, the multi-label classification model achieves a micro-average precision score of 0.757, with an average of 6.44 tags assigned per item. From this it can

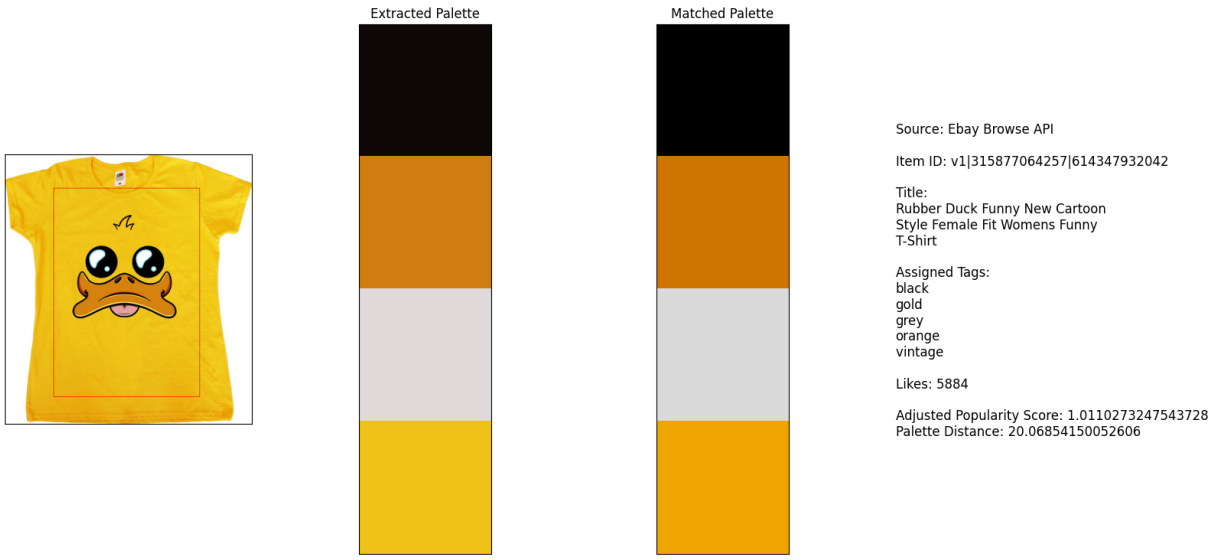

Figure 6.5: Example of a high accuracy palette match with a low $d_{Palette} \approx 20$.

be concluded that the tags are assigned with an acceptable level of precision, while providing a reasonable number of tags to work with.

It is noted that for a given design, an MIoU score of less than 1 in an over-segmenting case can still pick up a significant amount of background which results in the clustering algorithm choosing irrelevant colours for the palette and resultantly false positive tags. However, because the majority of the tags are identified correctly, the impact on generation using extra erroneous tags may be minimal, if the overall colour scheme matches.

6.3 IR Expressiveness

6.3.1 Method

The utility of the AST representation is evaluated by assessing whether it is expressive enough to capture a wide variety of print design graphic styles. A small, diverse sample of print designs are collected from online marketplaces across multiple templates and styles. Designs are then manually replicated by writing code that uses the AST interface, adjusting component position parameters, fonts and colours to match that of the reference design. The corresponding SVG render is compared qualitatively against the reference design.

6.3.2 Results

The system was found capable of replicating the more simple structural patterns found in real-world examples. This is in particular with designs where text and image components are drawn distinctly. However, complicated structures that are more artistic in nature are difficult to replicate, and require further extensions to the node classes. This is in particular the case for graphics where text and images are merged in an artistic manner, or where complicated gradients, shapes or line art are used in text designs. In these cases, fonts are limited and probably require their generation in raster graphics instead with generative models.

SVG supports many other attributes that have not been implemented in the internal representation such as text curvature, outlines, and a wide range of other rendering styles. The current state of the proposed system limits the visual complexity that can be achieved. However, each node is designed to be extendible and supporting these features is only a matter of including attributes for their SVG compilation.

6.4 Design Generation

6.4.1 Method

The quality of the designs output by the pipeline is highly dependent on the capabilities of the models used to generate the designs. For local development, the `stable-diffusion-v1.5` model has very poor and inconsistent outcomes out-of-the-box for image generation, for a commercial context. OpenAI's `DALL-E3` model is a significant improvement, although results still demonstrate inconsistencies at times. Therefore, the `DALL-E3` model is used in evaluating design generation.

To assess the contributions of the pipeline as independently of the model as possible, the evaluation focuses on the ability of the pipeline to function as a mutation operation of the reference design. Since the extracted features are from the colour palettes and title content, the following subjective scoring approaches are applied to a generated population of 100 graphics:

- **Palette score:** Measures the utility of the tags for maintaining the parent image colour scheme. Some deviation is acceptable due to the coarse assignment of colours. The following criteria are used to assess palette score:
  - The presence of colours in the original design.
  - Excess colours that dominate, but are not in the original design.
- **Prompt relevance:** Measures how appropriate the contents of the prompt used to generate the design is compared to the parent design and title:
  - The key subjects/themes of the parent design and title are retained.
  - Prompt structure with natural language where adjectives modify nouns is better.
  - Any extra content mentioned is relevant to the original design.
- **Aesthetic quality:** Measures how aesthetically appealing is the design in relative to the reference design, taking into account:
  - Viability of the content of the design.
  - Similarity to the parent design.
  - Level of detail.
  - Lack of erroneous segmentation from background removal.

Each score is number assigned between 0 and 5, with 5 indicating strong results. An acknowledged drawback of this evaluation, however, is the susceptibility to bias with one human scorer. A more robust evaluation of these metrics would survey more people.

6.4.2 Results

The histograms in Figure 6.6 present the distribution of colour, prompt and aesthetic scores for the 100 generated designs.

- **Palette Score:** The mean palette score is 3.18, suggesting that for most generated images, most dominant colours of the parent image are retained.
- **Prompt Relevance:** The mean prompt score is 2.81. Since image generation draws on concrete concepts from the title information, irrelevant outcomes occur for designs with vague titles.
- **Aesthetic Quality:** The mean aesthetic score is 2.23. 24 of the 100 items are scored 4 or 5.

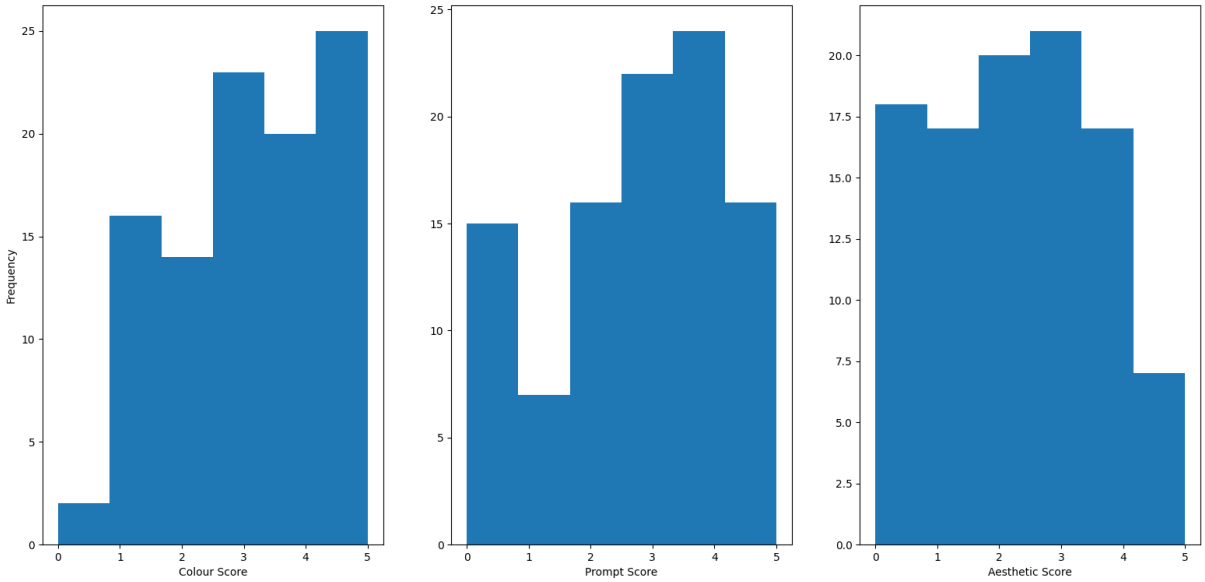

Figure 6.6: Histograms of colour (left), prompt (middle) and aesthetic (right) scores.

The existence of low-quality designs indicates the need for improved filtering or data pruning of the database prior to generation. For design titles that do not contain concrete descriptions of the semantic content, prompt generation is difficult and typically requests the model to produce abstract art which often appears nonsensical and unappealing. In other cases, where the contents of the design refer to real people, trademarks, or contentious themes such as violence, the models are aligned so that these are not reproduced.

Performant designs extract well-defined semantic content from the titles, generally have diverse and appropriate colour tags are assigned. In particular, print designs with animals tend to perform consistently well, as demonstrated by the examples highlighted in Figure 6.7.

Figure 6.7: Example designs that score highly across each metric.

Optimisation

With these results in mind, the prompt generation method is optimised to state the explicit need for a concrete, centred object. While this limits the variance in the possible output styles, the stronger scores for this pattern of design justifies this focus. An additional 12 designs are evaluated, with the average results outlined in Table 6.3.

These results are a strong indication that experimentation with prompt engineering and result

Entries	Palette Score	Prompt Score	Aesthetic Score
12	4.08	3.83	3.25

Table 6.3: Scores for designs generated using the optimised prompt.

filtering may improve model output further.

Chapter 7

Conclusion

This project developed a pipeline to generate T-shirt print designs, by combining a series of machine learning tasks for image segmentation and high-level feature extraction, to prompt and image generation. Among the print design isolation algorithms explored, the procedural Contour and transformer-based SegformerB3 algorithms perform best with the modified IoU metric scores of 0.688 and 0.673 respectively. K-means clustering for colour palette extraction and subsequent K-NN search for multi-label classification of colour-related textual tags results in a 0.757 micro-average precision score with an average of 6.44 tags assigned per item. Finally, the design generation using the insights of this pipeline achieves a 24% viability rate for the use of print design graphic components, showcasing the potential for commercial applications.

7.1 Future Work

Due to the horizontal nature of this project, there are many possible directions for further experimentation, optimisation and extension at each stage of the pipeline. Multimodal methods can be developed to better filter for higher-quality marketplace T-shirt designs during the ingestion phase. The reliability of the segmentation models can be made more accurate, for instance using a hybrid approach of the SegformerB3 and contours segmentation, or by optimising the UNet model. The palette distance metric formulation can be experimented with to minimise the maximum colour distance pairings instead of the sum. OCR models can be employed to label text-based designs. These are all promising areas that may improve the measured outcomes. However, a key area for future work in particular is the development of the evolutionary algorithms approach.

Evolutionary Algorithms

Metrics such as the adjusted popularity score and palette distance are developed for assessing colour palette aesthetic quality, though remain untested. Many other metrics can be included to engineer a fitness function to optimise for semantic quality and alignment. CLIP can be used to generate and compare embeddings of the post-processed design with the prompt - including this in the fitness measure would ensure the contents of post-processed designs in the population stay relevant. Aesthetic scoring models can also be incorporated. The optimal combination of these metrics into a unified metric is challenging to determine, but remains an interesting potential area for future work.

Chapter 8

Bibliography

[1] Statista. Apparel, footwear and accessories retail e-commerce revenue in the United States from 2017 to 2029 (in billion U.S. dollars) [Graph]. Statista; 2024. Available from: https://www.statista.com/statistics/278890/us-apparel-and-accessories-retail-e-commerce-revenue/.

[2] Chen KT, Luo J. When Fashion Meets Big Data: Discriminative Mining of Best Selling Clothing Features. In: Proceedings of the 26th International Conference on World Wide Web Companion. WWW '17 Companion. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee; 2017. p. 15–22. Available from: https://doi.org/10.1145/3041021.3054141.

[3] Ooi KB, Tan GWH, Al-Emran M, Al-Sharafi MA, Capatina A, Chakraborty A, et al. The Potential of Generative Artificial Intelligence Across Disciplines: Perspectives and Future Directions. Journal of Computer Information Systems. 2025;65(1):76-107. Available from: https://doi.org/10.1080/08874417.2023.2261010.

[4] Liu Y, Wang L. MYCloth: Towards Intelligent and Interactive Online T-Shirt Customization based on User's Preference. In: 2024 IEEE Conference on Artificial Intelligence (CAI); 2024. p. 955-62.

[5] Skenderi G, Joppi C, Denitto M, Scarpa B, Cristani M. The multi-modal universe of fast-fashion: the Visuelle 2.0 benchmark. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW); 2022. p. 2240-5.

[6] Intellectual Property Office. Exceptions to Copyright. Intellectual Property Office. 2021. [Date Accessed: 16 January 2025]. Available from: https://www.gov.uk/guidance/exceptions-to-copyright.

[7] Miao Z, Chen K, Fang Y, He J, Zhou Y, Zhang W, et al. Cost-Effective Online Trending Topic Detection and Popularity Prediction in Microblogging. ACM Trans Inf Syst. 2016 Dec;35(3). Available from: https://doi.org/10.1145/3001833.

[8] Ravi A, Patro A, Garg V, Rajagopal AK, Rajan A, Banerjee RH. Teaching DNNs to design fast fashion; 2019. Available from: https://arxiv.org/abs/1906.12159.

[9] Singrodia V, Mitra A, Paul S. A Review on Web Scrapping and its Applications. In: 2019 International Conference on Computer Communication and Informatics (ICCCI); 2019. p. 1-6.

[10] Andriyanov N, Dementev V. Combining Computer Vision and Word Processing to Classify Film Genres. In: 2023 25th International Conference on Digital Signal Processing and its Applications (DSPA); 2023. p. 1-5.

[11] Wahed MA, Alzboon MS, Alqaraleh M, Ayman J, Al-Batah M, Bader AF. Automating Web Data Collection: Challenges, Solutions, and Python-Based Strategies for Effective Web Scraping. In: 2024 7th International Conference on Internet Applications, Protocols, and Services (NETAPPS); 2024. p. 1-6.

[12] eBay Inc. eBay Browse API; 2023. [Accessed: 18 January 2025]. https://developer.ebay.com/api-docs/buy/browse.

[13] eBay Inc. eBay Marketplace Insights API; 2023. [Date Accessed: 18 January 2025]. https://developer.ebay.com/api-docs/sell/marketplace-insights.

[14] Amazon com, Inc. Amazon Product Advertising API; 2022. [Date Accessed: 18 January 2025]. https://webservices.amazon.com/paapi5/documentation/.

[15] Canny J. A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1986;PAMI-8(6):679-98.

[16] OpenCV. Contours: Getting Started; 2025. [Accessed: 25 May 2025]. Available from: https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html.

[17] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Commun ACM. 2017 May;60(6):84–90. Available from: https://doi.org/10.1145/3065386.

[18] Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017;39(6):1137-49.

[19] Redmon J, Divvala S, Girshick R, Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016. p. 779-88.

[20] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N, Hornegger J, Wells WM, Frangi AF, editors. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. Cham: Springer International Publishing; 2015. p. 234-41.

[21] Oquab M, Bottou L, Laptev I, Sivic J. Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition; 2014. p. 1717-24.

[22] Weller H. Color Spaces; 2021. [Accessed: 21 March 2025]. Available from: https://cran.r-project.org/web/packages/colordistance/vignettes/color-spaces.html.

[23] Sharma G, Wu W, Dalal EN. The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. Color Research & Application. 2005;30(1):21-30. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/col.20070.

[24] Printify. How to create a t-shirt design: Full guide; 2025. [Accessed: 20 January 2025]. https://printify.com/blog/how-to-create-a-best-selling-t-shirt-design/.

[25] Dong C, Loy CC, He K, Tang X. Image Super-Resolution Using Deep Convolutional Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2016;38(2):295-307.

[26] Li TM, Lukáč M, Michaël G, Ragan-Kelley J. Differentiable Vector Graphics Rasterization for Editing and Learning. ACM Trans Graph (Proc SIGGRAPH Asia). 2020;39(6):193:1-193:15.

[27] Xing X, Hu J, Liang G, Zhang J, Xu D, Yu Q. Empowering LLMs to Understand and Generate Complex Vector Graphics. ArXiv [Preprint]; 2025. Available from: https://arxiv.org/abs/2412.11102.

[28] Henstridge J. GIMP Python Documentation; 1999. Version 0.4, accessed on 2025-05-27. Available from: https://www.gimp.org/docs/python/index.html.

[29] The Inkscape Project. Inkex Documentation; 2025. Accessed: 2025-05-27. Available from: https://inkscape.gitlab.io/extensions/documentation/.

[30] Sohl-Dickstein J, Weiss E, Maheswaranathan N, Ganguli S. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In: Bach F, Blei D, editors. Proceedings of the 32nd International Conference on Machine Learning. vol. 37 of Proceedings of Machine Learning Research. Lille, France: PMLR; 2015. p. 2256-65. Available from: https://proceedings.mlr.press/v37/sohl-dickstein15.html.

[31] Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B. High-Resolution Image Synthesis with Latent Diffusion Models . In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Los Alamitos, CA, USA: IEEE Computer Society; 2022. p. 10674-85. Available from: https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.01042.

[32] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial networks. Commun ACM. 2020 Oct;63(11):139–144. Available from: https://doi.org/10.1145/3422622.

[33] Cully A. Introduction to Machine Learning: Module 7.2 History and General Concepts. Imperial College London; 2024. [Date Accessed: 15 January 2025]. Available from: https://intro2ml.pages.doc.ic.ac.uk/autumn2024/modules/module7/history-general-concept.

[34] Grabe I, Zhu J, Agirrezabal M. Fashion Style Generation: Evolutionary Search with Gaussian Mixture Models in the Latent Space. In: Martins T, Rodríguez-Fernández N, Rebelo SM, editors. Artificial Intelligence in Music, Sound, Art and Design. Cham: Springer International Publishing; 2022. p. 84-100.

[35] Cully A, Demiris Y. Quality and Diversity Optimization: A Unifying Modular Framework. IEEE Transactions on Evolutionary Computation. 2018;22(2):245-59.

[36] Gatys L, Ecker A, Bethge M. A Neural Algorithm of Artistic Style. arXiv. 2015 08.

[37] Gatys LA, Ecker AS, Bethge M. Image Style Transfer Using Convolutional Neural Networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016. p. 2414-23.

[38] OpenAI. OpenAI API; 2024. Accessed 2025-05-25. https://platform.openai.com.

[39] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17. Red Hook, NY, USA: Curran Associates Inc.; 2017. p. 6000–6010.

[40] Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, et al.. LoRA: Low-Rank Adaptation of Large Language Models; 2021. Available from: https://arxiv.org/abs/2106.09685.

[41] Ding S, Chen X, Fang Y, Liu W, Qiu Y, Chai C. DesignGPT: Multi-Agent Collaboration in Design. In: 2023 16th International Symposium on Computational Intelligence and Design (ISCID); 2023. p. 204-8.

[42] Abdul Salam M. Sentiment Analysis of Product Reviews Using Bag of Words and Bag of Concepts. International Journal of Electronics. 2019 12;11:49-60.

[43] Alharbi NM, Alghamdi NS, Alkhammash EH, Al Amri JF. Evaluation of Sentiment Analysis via Word Embedding and RNN Variants for Amazon Online Reviews. Mathematical Problems in Engineering. 2021;2021(1):5536560. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/5536560.

[44] Park K, Park S, Joung J. Contextual Meaning-Based Approach to Fine-Grained Online Product Review Analysis for Product Design. IEEE Access. 2024;12:4225-38.

[45] Chen J, An J, Lyu H, Kanan C, Luo J. Learning to Evaluate the Artness of AI-Generated Images. IEEE Transactions on Multimedia. 2024;26:10731-40.

[46] Kao Y, He R, Huang K. Deep Aesthetic Quality Assessment With Semantic Information. IEEE Transactions on Image Processing. 2017;26(3):1482-95.

[47] Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, et al. Learning Transferable Visual Models From Natural Language Supervision. In: International Conference on Machine Learning; 2021. Available from: https://api.semanticscholar.org/CorpusID:231591445.

[48] Shir G. Color Hunt; 2015. Available from: https://colorhunt.co/.

[49] Virkus D. Dafonts Free Dataset; 6 March 2022. Available from: https://github.com/duskvirkus/dafonts-free.

[50] DeepSeek-AI, Guo D, Yang D, Zhang H, Song J, Zhang R, et al.. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning; 2025. Available from: https://arxiv.org/abs/2501.12948.

[51] Afridi S. Segformer B3 Fine-Tuned for Clothes Segmentation; 2024. [Accessed: 29 March 2025]. https://huggingface.co/sayeed99/segformer_b3_clothes.

[52] Xie E, Wang W, Yu Z, Anandkumar A, Alvarez JM, Luo P. SegFormer: simple and efficient design for semantic segmentation with transformers. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. NIPS '21. Red Hook, NY, USA: Curran Associates Inc.; 2021. .

[53] Shannon CE. A mathematical theory of communication. The Bell System Technical Journal. 1948;27(3):379-423.

[54] Bradski G. The OpenCV Library. Dr Dobb's Journal of Software Tools. 2000.

[55] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Advances in Neural Information Processing Systems 32. Curran Associates, Inc.; 2019. p. 8024-35. Available from: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[56] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. Journal of machine learning research. 2011;12(Oct):2825-30.

[57] Rubner Y, Tomasi C, Guibas LJ. A metric for distributions with applications to image databases. In: Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271); 1998. p. 59-66.

[58] Kuhn HW. The Hungarian method for the assignment problem. Naval Research Logistics Quarterly. 1955;2(1-2):83-97. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

[59] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods. 2020;17:261-72.

[60] Gatis D. rembg: Remove image background; 2020. Accessed: 2025-06-09. https://github.com/danielgatis/rembg.

[61] Qin X, Zhang Z, Huang C, Dehghan M, Zaiane O, Jagersand M. U2-Net: Going Deeper with Nested U-Structure for Salient Object Detection. vol. 106; 2020. p. 107404.

Chapter 9

Declarations

9.1 Use of Generative AI

Research for this project uses OpenAI's ChatGPT-4 model (https://chat.openai.com/) as a search engine for research, querying software documentation and to support report writing. No content generated by AI has been presented as my own work.

9.2 Ethical Issues

This study relies on datasets gathered through web scraping, a practice that raises ethical and even legal concerns depending on the methods employed and the nature of the data. Automated web scraping may breach a website's terms of service, as it may add unnecessary burden on its servers, or they may intend to monetise this data. The data gathered in this study does not include personal information. While business competition is inherently competitive in the marketplace, an argument could be made that the scalability of automated design generation using design data collected from existing sellers to compete against those same sellers requires careful ethical consideration.

Another ethical concern arises from the use of generative AI to produce designs. particularly with regards to copyright infringement. AI-generated designs could infringe on intellectual property rights if they resemble pre-existing designs or trademarks too closely. Detecting copyright violations at scale poses a significant challenge and is beyond the scope of this project, however it is worth acknowledging this limitation for future commercial use. Additionally, the use of AI-generated content risks the production of maligned content. For instance, text designs with threatening or abusive content or explicit imagery may not only be against the terms of service of the sales platform, but may also break the law. Mitigation of this risk varies by model, with extensive alignment methods used to filter harmful content. While this study does not focus on content moderation, appropriate steps have been taken with the choice of models and data sources gathered from to limit this risk.

9.3 Availability of Data and Materials

The source code is made publicly available at:
https://gitlab.doc.ic.ac.uk/shc220/fyp-selling-shirts.

Datasets are located in the GitLab repository under the `data/` directory. The relational database is found in `data/db/dev_database.db`. Pickled DataFrames objects containing online marketplace clothing data used for validation testing of the segmentation algorithms can be found in the `data/dataframes/labelled_image_bboxs.pickle` file. UNet training data and models are found in the `data/images/` and `data/models` directory respectively.