Department of Computer Science

BSc (Hons) Computer Science (Artificial Intelligence)

Academic Year 2023 – 2024

An Effective Workflow for Generating Comic Book Prototypes with Consistent Characters

Daniel C. Rodrigues

2017796

A report submitted in partial fulfilment of the requirements for the degree of

Bachelor of Science

Brunel University London
Department of Computer Science
Uxbridge
Middlesex
UB8 3PH
United Kingdom
T: +44 1895 203397
F: +44 (0) 1895 251686

Abstract

Text-to-image diffusion models can generate highly detailed images from text prompts. However, as the generation of these images is an inherently random process, artists in particular find it hard to use these models to prototype their artwork. This problem is magnified severalfold when it comes to comic books, as comics require the same characters to appear multiple times throughout their pages to weave together a coherent narrative. Thus, if artists want to use these diffusion models to help with prototyping comics, conditioning the models' random generation process using external techniques is necessary to produce consistent characters. This project tackles this problem by proposing five different strategies to maintain consistent characters through the use of different conditioning techniques based on the latest add-ons and extensions to the Stable Diffusion models and their surrounding ecosystem. The strategies are implemented and rigorously evaluated against a set of five different success criteria and are scored based on their performance. The best performing strategies were found to be the ControlNet and LoRA training strategies, and the project explains how to best utilise them. Based on the findings, a final workflow for artists was proposed, which allows artists to generate consistent characters across different images for their comics. The workflow is entirely based on open-source models and tools, providing the artist complete control over their own processes as compared to other alternatives with Midjourney, Dall-E and such. The workflow proposed also works with different amounts of data, giving the artist options even if they don't have a lot of images of their original character, and even better options if they happen to have more than a couple dozen images.

Acknowledgements

I would like to take this opportunity to thank my parents for their undying love and support through all my endeavours. I also thank my supervisor, Dr. Mahir Arzoky for all the help, support, and valuable insight he has provided all throughout the dissertation process.

I would also like to extend my genuine gratitude towards the efforts of the open-source AI community, and their complete dedication towards democratising AI and making the fruits of their labour available to everyone around the world for free.

I certify that the work presented in the dissertation is my own unless referenced.

Signature  _____

Date 21-03-2024_____

Total Words: 15666

Table of Contents

List of Figures

List of Tables

Glossary

Artifact	A type of accidental distortion or illogical feature in an AI-generated image. Such as extra fingers or hands, twisted features, fewer/extra eyes, fused clothing, etc.
Diffusion models	A type of Neural Network that generates an image based on a natural language prompt. Most commonly used to generate images of nearly anything in high-resolution, with these models being trained on billions of images of people, everyday items, and life.
Embedding	An embedding is a high-dimensional representation of the semantic connections between words. For example, on a 2D graph, if the words, "computer" and "program" were charted, they might appear to be in close proximity to one another, because the words are semantically related in natural language. An embedding is such a representation; however, it contains vector representation of words, and can exist in multiple dimensions above the familiar two or three.
Fine-tune	The process of re-training an existing, pre-trained Neural Network. This is done when the new concept being trained is similar to the model's original training and is utilised so that the whole training process does not need to be repeated from scratch.
GANs	Generative Adversarial Networks, a type of generative AI that was invented and gained popularity in the 2010s.
Natural language	Plain language, such as English, spoken or written. Using natural language in the context of deep learning entails using plain English with these AI models, as opposed to code or any kind of formalised language that is syntactic in nature.
Open Source	Software for which the source code is publicly distributed and may be modified or used by anyone.
SD	Short for Stable Diffusion, a type of open-source diffusion model.

1. Introduction

1.1. Project Introduction

Diffusion models power the recent advances in generative AI, where AI models can generate and output an image based on a text description in English provided by a user (Yang, et al., 2023). As such, these tools can help artists quickly prototype new work, visualise ideas, and iterate on existing concepts far more effectively than traditional AI techniques like GANs allow for (u/scifivision, 2023) (Dhariwal & Nichol, 2021). Moreover, they are also widely used by regular laymen wanting to foray into the world of digital media creation, but who do not possess the required skill to do so.

However, there is a lack of robustness regarding more complex generative work (Boutin, et al., 2023), with the main problem being that it is very challenging to maintain consistency between characters, environments, and art styles through different prompts (Packer, 2023) (An, 2023) (u/MysteryInc152, 2022) (Monzon Media, 2023) (Meccai, 2023). That is, diffusion models work by the user inputting a text prompt, which is then used by the model to denoise a random noise sample using the text input as a CLIP embedding (Ramesh, et al., 2022). Thus, by its very nature, this process is random, and the exact same text prompt, if re-run, will generate a completely different image every time (Computerphile, 2022). Thus, even if a user may desire to generate digital art of a character performing a certain action, the character in question will be different every single time a new image is generated. Moreover, the art style and environment will be different too, and little details like the characters' clothing, items, accessories, and differences in expression will nearly always look different, even if, by some chance, the overall character happens to look similar between two different generations.

This is a major drawback when using diffusion models to generate any kind of consistent character, and this problem is multiplied exponentially when generating a comic book page, because every panel in the page needs to have the same characters and backgrounds, while also maintaining a consistent art style. Comic book generation is also limited in the way a coherent narrative can be depicted from panel-to-panel in a page (Pan, et al., 2024) (Victor Gnarly, 2023). That is, the story unfolding across the panels on the page must not be the characters performing random actions. Rather, they must tell a clearly ordered story that can be followed and understood when reading through the page.

This project aims to propose a systematic workflow pipeline that artists can use alongside a free, open-source diffusion model system, Stable Diffusion, to quickly prototype comic books (Rombach, et al., 2022a). Stable Diffusion will be used for this project because it is open source, allowing any user to run any pretrained stable diffusion models and follow the workflow from their own local machines (StabilityAI, 2022a). Alternative models like Dalle (Ramesh, et al., 2022) and Midjourney (Midjourney, 2022) exist, however, they're paid, non-open-source software, which means the artists don't get to maintain any control. This project's solution is a workflow developed after identifying the best techniques and strategies a person can use to achieve consistent characters. Thus, the artist will have the ability to use any stable diffusion model they desire, including ones they train themselves, and thereby retain as much control over their workflow as possible.

1.2. Aims and Objectives

The aim of this project is to propose a streamlined workflow, so artists are able to train their own models or use pretrained models for a particular art style by using stable diffusion on their personal devices to prototype comics efficiently. To achieve this, the following objectives needed to be met:

1. Conduct a thorough literature review to gain a definitive understanding on the current bleeding edge of AI art generation technologies and what toolkits, models, techniques, and alternative workflows have already been developed to tackle this problem.
2. Data collection and preprocessing; gather a hundred comic book pages and a hundred comic book panels and write text descriptions for every single image.
3. Fine-tune two stable diffusion models using DreamBooth on the pages dataset and the panels dataset using the prepared data and evaluate their performance. Choose a single one of the two models for the workflow going forward.
4. Experiment with a number of different community-trained stable diffusion models publicly available and select up to two models that best suit a comic art style.
5. Create and implement a clear set of up to 5 different strategies to generate consistent characters and implement them using the stable diffusion ecosystem.
6. Evaluate the decided strategies on a number of different success criteria such as character consistency, background consistency, control over final output, the presence of artifacts in an image, and so on.

7. Using the most effective strategies, formulate a streamlined workflow that artists can follow to quickly prototype comics while making sure their characters remain consistent throughout.

1.3. Project Approach

This project will be examining and experimenting with a variety of models within stable diffusion, along with five different strategies to generate consistent characters. A formal design methodology, Double Diamond, will be used (British Design Council, 2005). This will be further elaborated upon in the methodology chapter. For the project, first, a thorough literature review will be carried out, and alternative solutions will be explored. An exploration into the current state-of-the-art open-source generative AI will also be carried out, by examining the latest work posted in AI art generation communities online on Discord and Reddit, as these communities are populated with people in the field as well as strong enthusiasts that push the limits of what's currently possible.

Next, it must be determined whether to generate an entire comic page from a single prompt, or if generating each panel by itself is a better idea. For both ideas, an existing model of stable diffusion will be fine-tuned using 100 images of each, pages and panels. Both models will be evaluated, and a single approach will be chosen. Based off the chosen style, two more pretrained models will be downloaded off CivitAI (CivitAI, 2022), a model sharing website for enthusiasts. This is so cross-evaluation can be carried out to ensure that the strategies for consistent characters can be replicated across any model and doesn't exclusively work on the two models fine-tuned earlier.

Finally, five strategies will be decided upon using the latest add-ons and extensions to the stable diffusion ecosystem, such as img2img, ControlNet, and LoRA training, and the models will be evaluated across different metrics for each strategy. All work will be tracked and managed through Trello, a project management software. Code will be written and stored on Google Colab on Google Drive, which has its own inbuilt versioning system. Miscellaneous code written for utility that's not stored in Colab will be uploaded and version-tracked through GitHub. All models trained will be stored on HuggingFace.

1.4. Dissertation Outline

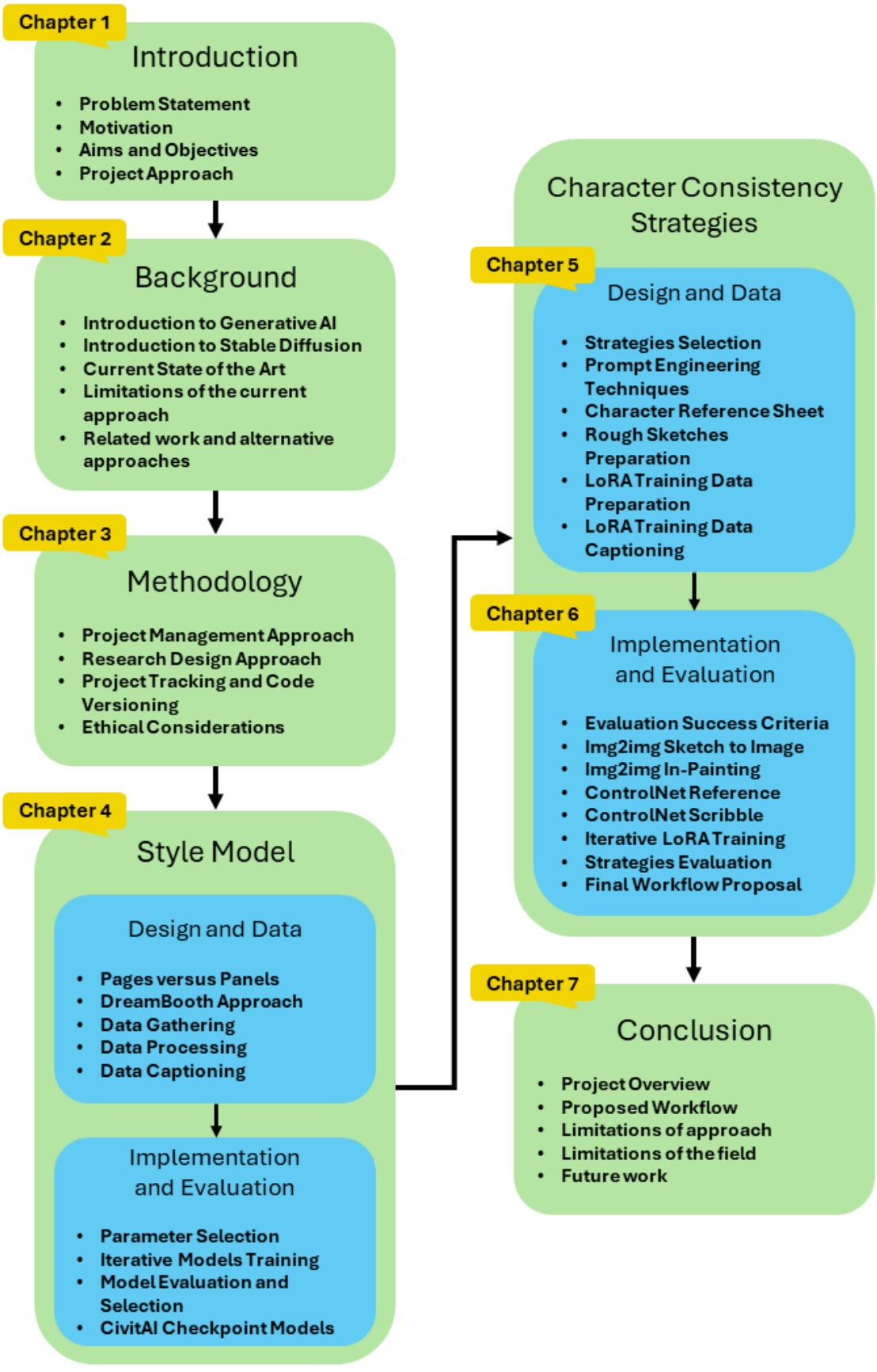


Fig. 1.1: Outline of the Dissertation

Figure 1.1 indicates the outline of the dissertation, detailing the content of each chapter along the way. Within the figure, every dialog box is either a chapter, or a couple of chapters (in the case of chapters 5 and 6), and the bullet points detail the key concepts that will be tackled within. This is not a one-to-one mapping for the sections and subsections within each chapter, but rather represents the overall ideas that will be conveyed through the chapter. The ideas and concepts will be tackled systematically in the same order as the bullet points, which is aimed at providing a comprehensive, step-by-step information flow to the reader, where each concept logically feeds into the next.

2. Background

2.1. Overview

To fully provide the reader the context necessary to appreciate the motivation behind the project, this chapter will report on the beginnings of generative AI, the progress achieved to modern day, and the limitations that still exist. A thorough explanation of the Stable Diffusion ecosystem is also given, including the alternative tools existing to provide digital artists with easy ways to prototype their work. This formed the bulk of the literature review and requirements gathering phase, and as such, this chapter will be insightful in elucidating the decisions taken in the later chapters to build the strategies for generating consistent characters.

2.2. Generative AI

2.2.1. GANs

Generative Adversarial Networks, abbreviated as GANs, were created in 2014 (Goodfellow, et al., 2014), and were one of the first pioneering generative AI approaches, in an era where the word "creative" had never been used to describe AI. StyleGAN2 (Karras, et al., 2020), one of the crowning achievements of this architecture, was introduced by NVIDIA in 2019, and achieved state-of-the-art results in generating high-fidelity images of faces, objects, and scenes.

GANs consist of two neural networks, a generator and a discriminator, that are trained together in a process inspired by game theory (Nash, 1951). The key idea is that the generator aims to create new data instances resembling its training data, while the discriminator tries to distinguish real data from the generator's fakes. This technique allows the AI to produce a final output that is remarkably realistic and seemingly creative. However, this technique ultimately suffers from a severe lack of control, as it ultimately only produces data that mimics a target distribution: its training data. For example, the instance of StyleGAN2 on www.thispersondoesnotexist.com/ (Wang, 2019) was trained on human faces and can only generate faces. There is no option for variety and the user cannot retain creative control over the outputs. Moreover, GANs also suffer from artifacts and challenges in evaluating the diversity of its outputs (Saxena & Cao, 2021).

2.2.2. Diffusion Models

Diffusion models were an absolute game changer in the field of generative AI (Ramesh, et al., 2022) (Rombach, et al., 2022a) (Podell, et al., 2023) (Rombach, et al., 2022b). Based on several different research breakthroughs in the past decade, these models are by far the most undeniably creative AI models that have ever been made (Jonas, 2022). Giving the user a considerable amount of control through text prompts, these models are capable of generating high-quality, photorealistic images that are difficult for humans to tell apart from real images (Ha, et al., 2024) (Nichol, et al., 2021). The first diffusion model was introduced in 2015, inspired by non-equilibrium thermodynamics (Sohl-Dickstein, et al., 2015). It proposed a method to destroy a data structure by adding noise systematically until the result was a pure noise distribution.

The key insight then applied was a way for the models to iteratively reverse the process over dozens of steps and restore the original sample from the pure noise data, an entropy-reversal function. One of the biggest breakthrough achieved in this field was when OpenAI devised a way to control the denoising process based on their prior work with the transformer architecture, natural language supervision, and multimodal learning (Ramesh, et al., 2022). This process was called Contrastive Language-Image Pre-training—CLIP, for short—wherein the text prompt entered by a user is transformed into a dense vector representation that captures the semantic features of the prompt in latent space (Radford, et al., 2021). The model then uses the CLIP embedding to score images based on how likely they are to be classified under a given text prompt. The original image has the highest score, and the image with pure noise has the lowest. Each step of the denoising process along the way is also scored, and the aim of the model is to attain the best score as the denoising process progresses. It does this by modifying the image and removing a little bit of noise each time such that the edited image has a slightly higher score than the prior image.

OpenAI's Dall-E 2 (Ramesh, et al., 2022) was the first model that created the biggest splash in the field using a combination of a diffusion model with a CLIP embedding, which was then followed by multiple other models, most notably, Imagen (Saharia, et al., 2022), Midjourney (Midjourney, 2022), and Stable Diffusion (Rombach, et al., 2022a).

2.2.3. Limitations of Diffusion Models

Diffusion models are not flawless. The most notable issues are the resolution and training requirements. Most diffusion models are best trained using consistently sized training images

with the same resolution, such as 512 by 512 pixels for Stable Diffusion (Rombach, et al., 2022a). The output images at inference time are bound by this same resolution. Any images generated *under* this resolution are very bad, and those generated *over* the resolution tend to be noisy and of low quality, requiring different models to upscale the image and improve its resolution. When training, diffusion models consume extensive computing resources, and take a very long time to train. Scaling up model size and quality thus requires not just time, but also an enormous amount of computational power. However, when looking at this purely from the perspective of artists or enthusiasts wanting to generating images to help their workflow and increase their productivity, there are worse problems.

The first is image artifacts (Yin, et al., 2022), which is by far the most commonly encountered issue within online AI art communities (some examples: r/StableDiffusion, reddit.com: search results - "artifact"). Generative AI models are known for producing severe distortions that immediately provide a clue that an image is AI generated. For example, features like extra eyes, hands, pieces of clothing and other items, twisted and distorted limbs and facial features, etc. that emerge when generating pictures of people. The other problem is the randomness, lack of consistency, and lack of fine-grained control over the final output. As described above, the iterative process of denoising an image according to a prompt is pseudorandom, and every time an image is generated, it will be different even if the prompt is the exact same. The only way to change this is to use the same seed, which then outputs the exact same image every time, which is not ideal either. This is a big issue when using the models for art, as an artist would naturally want to keep their characters and background consistent if they're generating images for a specific story or setting.

These issues are being addressed through the application of additional conditioning techniques on top of the base diffusion model, which will be explored in the next section on Stable Diffusion. Finally, the last way to generate consistent, yet creative characters, objects, and styles is to fine-tune the model being used. Fine-tuning retrains the last few layers of the model with new data, while retaining all previous weights and biases, and is not a concept original to diffusion models. This is also explored in the next section.

2.3. Stable Diffusion

2.3.1. Stable Diffusion Models

Stable Diffusion—SD, for short—are a series of diffusion models trained by the company StabilityAI (StabilityAI, 2019). They are being used in this project because they are the best open-source diffusion models available. Dall-E and Midjourney are also available for public use, however, they are not open source; users need to purchase a subscription to use them and retain very little control over the models and conditioning techniques. SD version 1.5 (StabilityAI, 2022a) is the most widely used model as it is the best available model that doesn't require as much training time and compute resources as the newer SD models like SDXL (Podell, et al., 2023). **Thus, SD 1.5 is also what this project will be focusing on going forward** due to the same reason.

2.3.2. Automatic1111 Web UI

Automatic1111 Web UI (Automatic1111, 2022) is a community-developed web-based graphical user interface that allows users to explore the stable diffusion ecosystem and run SD models along with the multiple different conditioning techniques devised for them, such as img2img, ControlNet, and textual inversion. Other alternatives such as ComfyUI and InvokeAI exist, however as Automatic1111 is the most widely used GUI in the community, it has a lot more tutorials available to help new users get started, and also gets all the latest updates to the SD ecosystem, as its massive userbase is quick to adapt new changes. **It will be the primary tool used to generate images in this project going forward.**

2.3.3. Image Generation

There are two main categories in SD, namely, txt2img and img2img. The former is the main functionality of the model, wherein the user inputs a text prompt in natural language and the model generates an image. There are a number of parameters that can be tuned and modified to control the final output more, such as the generation seed, model samplers, sampling steps, batch size, etc. These can be finely tuned if needed, but basic, common settings used in the community work very well for most use cases. The next category, img2img, is based on a technique developed by Google Brain (Saharia, et al., 2022), and allows the user to input an additional image along with the text prompt. This section has the same settings and inputs as the prior txt2img category, in addition to allowing an extra input, a user's own image. This image will then be modified according to the text prompt, and the denoising strength

parameter, which is a floating-point number from 0 to 1, controls how much of the image is changed. For example, just the hair colour of a person could be changed, or their entire face and body could be swapped.

Img2img's Sketch function and In-Painting function will be focused on in this project, as they are the most useful when it comes to art generation. Img2img's sketch function allows a rough sketch of anything to be transformed into a finished image in the spirit of the sketch. Whereas in-painting allows the user to mask or mark out a portion of the input image by colouring over it, and the generation process then changes only the marked or selected section of the image. This is useful when the overall scene and poses of characters generated by the model is good, but certain character attributes, such as their clothing, hair, facial features, etc. need to be changed.

2.3.4. Checkpoint Models

The base Stable Diffusion models are only good for generating images based on a distribution of their training data, which is comprehensive and large, but is definitely not enough for most artistic use cases. As such, these models can be fine-tuned, wherein the pretraining of the base diffusion model is reused and the existing model is further trained on a smaller dataset of images, ranging anywhere from five images to thousands. This is a far cry from the billions of images that Stable Diffusion was originally trained on (StabilityAI, 2022b). The process of fine-tuning helps adapt the model for any particular style needed, such as different artistic styles, comic styles, anime styles, extreme photorealism, styles of famous painters, Disney movie styles, or even certain actors, people, characters, environments, natural landscapes, or cities. A specific trigger word can be set during this process and images generated henceforth by the fine-tuned model will be in the new style that was trained, if the trigger word is used.

Such fine-tuned models on a specific style are called checkpoint models. The most popular fine-tuning process for Stable Diffusion is through DreamBooth training (Ruiz, et al., 2023), a technique created by Google Brain for their own internal diffusion model, Imagen. Today, it is a widely used technique for fine-tuning SD, and a truly vast number of checkpointed models trained by various enthusiasts and users in the community on various styles are available for download for free on the CivitAI website (CivitAI, 2022).

2.3.5. LoRAs

Low-Rank Adaptations of Large Language Models—LoRA, for short—is a kind of fine-tuning process that sits atop a base or checkpoint model but doesn't produce a new checkpoint model (Hu, et al., 2021). Rather it results in a new, far smaller model—around 2 to 500 MB in size, as compared to the 2 to 20 GB that checkpoint models can be. LoRAs are used alongside a base or a checkpoint model. Besides its extremely small size, the greatest other benefit offered by LoRAs are that multiple LoRAs can be used along with a single checkpoint model. Generally, when generating an image, only a single checkpoint or base model can be used, which means that only a single specific style that was fine-tuned can be used. However, this changes with LoRAs since they can be fine-tuned on the same concepts as checkpoint models, but since multiple LoRAs can be used, different concepts can be fused into a single generated image. This is especially immensely helpful for comic art, since the comic art style can be the focus of the checkpoint model, whereas LoRAs can be trained for each individual character, style, background, city, building, etc. and all be used together in the same image.

The caveat with LoRAs is that they cannot work alone. They must be used on top of a checkpoint model. Without a checkpoint model acting as the base, they are useless. With a checkpoint that complements their style, however, they are incredibly powerful. LoRAs trained by the community can again be found in plentiful on CivitAI.

The most common way to train LoRAs is by using the trainers developed by a user in the community, Kohya (kohya_ss, 2022). A number of wrappers are built around Kohya's trainers, such as Google Colab notebooks (hollowstrawberry, 2023), or Web-GUIs (Maltais, 2022).

2.3.6. ControlNet

ControlNet is a conditioning technique developed by Stanford University researchers for Stable Diffusion (Zhang, et al., 2023). It introduces an additional neural network on top of SD to provide finer control over the generated output to the user. It takes an extra image as an input in addition to the usual text prompt, just like img2img, and outputs a latent vector to condition the incoming generation. ControlNet has options for multiple conditioning techniques that the user can choose, including but not limited to, canny, scribble, depth, reference, in-paint, and super-resolution. In this project, the following two will be given focus due to their relevancy to the problems being addressed:

- Scribble – Provided a rough sketch, the model will guide SD to generate a more refined and finished image based on the sketch. Similar to img2img's Sketch category.

- Reference – Provided a reference image with a particular character or object, the model guides SD to generate different images of a similar-looking character or object in different situations and poses (lllyasviel, 2023). This is often difficult to get right on the first try, and sometimes a lot of experimentation with tuning the generation parameters may be needed.

2.3.7. Stable Diffusion Architecture

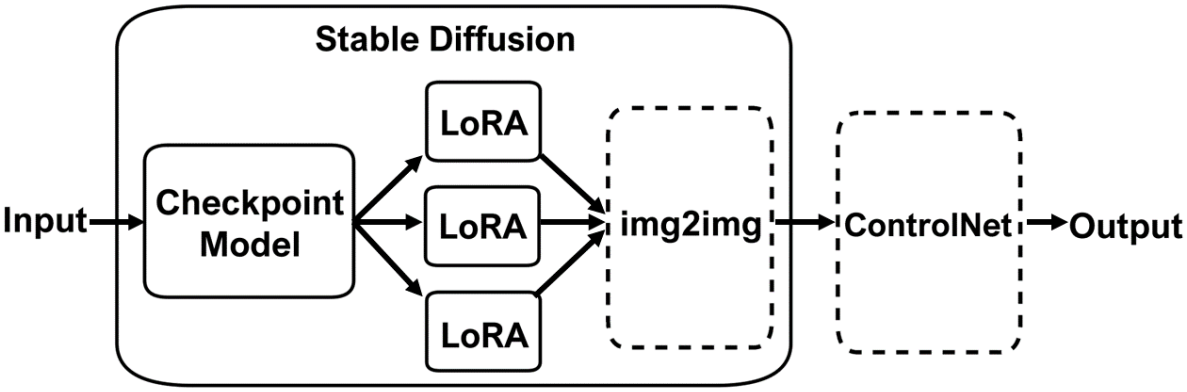


Fig. 2.1: Architecture Overview of Stable Diffusion

Figure 2.1 displays an overview of the SD ecosystem. In essence, a single input can be given to a single checkpoint model. As such, only a single checkpoint model can be used for a generation, while multiple LoRAs sitting on top of the checkpoint model can feed into the generation. An img2img input can optionally be introduced into the mix to provide the model a strong base image as the starting point for the generation. Finally, a ControlNet layer can optionally be used to guide and condition the final output even more.

2.3.8. Prompt Engineering

As stated earlier, diffusion models produce images based on text written by the user in plain English. However, there are several techniques one can use to maximise control and effectiveness of the text written in order to elicit maximum control over the desired output by the model. This is referred to as prompt engineering (Wang, et al., 2023) (Liu & Chilton, 2022). Text prompts can be specially crafted depending on the training data used to train the base model and fine-tune the checkpoint. For instance, certain photorealistic checkpoint models were trained on fluent descriptions in English, so prompting using the same technique would work best. But most anime-styled prompts are trained on booru tags (Mobo, 2017), thus crafting prompts for them also requires using similar styles. For example, a prompt a photorealistic model would use is “a man wearing a green parka standing in a park outside on

a warm, sunny day", where a booru-tag trained, anime model would use words and short phrases such as "1man, solo, standing, daytime, sun, park, trees, green parka".

Thus, when writing prompts, it is important to know how the checkpoint model was trained, and crafting the vocabulary, syntax, and structure of the prompt based on that. Additionally, negative prompts can also be given to the model as a request of what not to include. For instance, if the user wants pictures of a man only, they usually need to enter words like "woman, long hair, feminine, female" and such into the negative prompt category. If this is not done, sometimes the model will also return pictures of women even if the prompt asked for men (which is another limitation of diffusion models). Another example is if the model was trained on pictures of a lot of people with earrings, most generated pictures will also feature them unless "earrings, piercings" and such is provided to the model as a negative prompt.

Lastly, textual inversions are tiny files that can be trained as text embeddings and added on top of existing checkpoint models, just like LoRAs, and they are very effective in prompt engineering (Gal, et al., 2022). They are widely used in AI art communities to reduce undesirable effects in generated images (u/QinnShou, 2023). The two most popular ones used in this project are Easy Negative (Havoc, 2023) and Bad Prompt v2 (Nerfgun3, 2023).

2.4. Related Work

2.4.1. AI Art Communities

A lot of the review and requirements gathering phase of this project was undertaken by spending time in AI art communities online and simply noting common concerns and issues faced by artists when using these tools to improve their productivity. A lot of learning can also be done via simply reading through discussions in these communities, as they are created by enthusiasts to help other enthusiasts and share work and ideas amongst each other. Communities that were joined to research this project were CivitAI, bycloud's latent space, Movie Machine, OpenAI, and Two Minute Papers on Discord, and r/StableDiffusion, r/DalleE2, r/AIArt, and r/MediaSynthesis on Reddit.

2.4.2. Alternative Approaches

Currently, two approaches stand above the rest when generating comic book pages. They are, however, plagued by multiple restrictions. The first approach, using Stable Diffusion, is able to

generate a coherent page but only with well-known characters from popular media franchises (Jin & Song, 2023). The next best approach to tackle this has been headed by an online cartoonist in his comprehensive MovieMachine guide that aims to generate animated comic books while also aiming to maximise character consistency and coherency between panels (Victor Gnarly, 2023). However, his workflow relies on remix injection, a feature available only in the Midjourney models, which is a paid service and can only be accessed through Midjourney's official discord server—which is rather inconvenient compared to running a model on one's own device.

There are also some models on CivitAI for generating comic pages, as well as certain comic-layout styling prompts written by users in the community that are available on Discord and Reddit. However, these models and techniques are riddled with bugs and artifacts, and none of them are robust or effective enough to for real artists to use.

3. Methodology

3.1. Double Diamond Methodology

This project is broken into two main phases: initially training a comic style to serve as a base checkpoint model, then examining different strategies to maintain consistent characters between generations. As such, the project will be using the Double Diamond approach since the project is a research-based one and is exploratory in nature. Double Diamond is a formal design methodology defined by the British Design Council (British Design Council, 2005), and is divided into four phases: discover, define, develop, and deliver, as seen in figure 3.1 below.

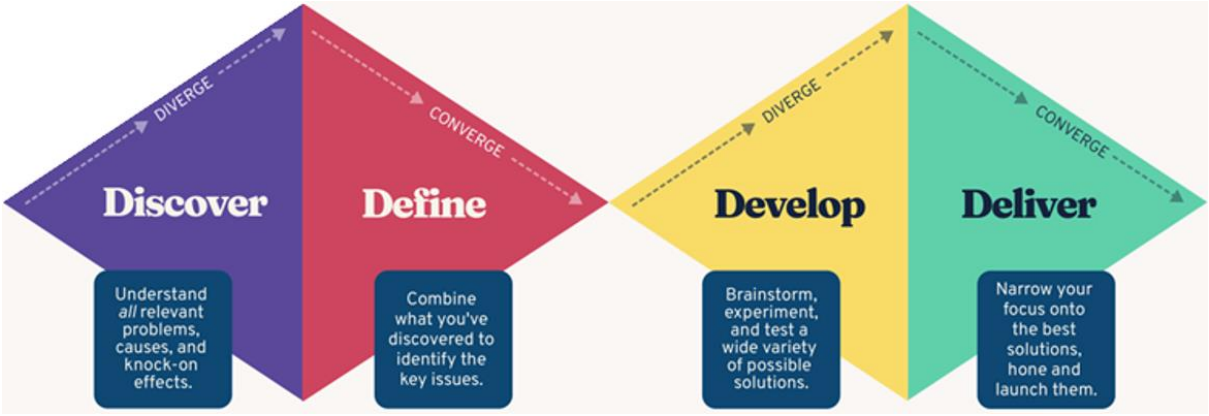


Fig. 3.1: The Double Diamond Methodology

This approach was selected for the project as its four phases allow for iterative research and exploration, which is necessary for an in-depth exploration of the problem space. The methodology itself is based upon the principle of divergence and convergence (Banathy, 1996). The former is undertaken as an exploratory tactic to study the problems and existing literature in-depth, and once sufficient ideas have been generated, the researcher may begin to focus on a few solutions and converge upon certain results and answers. This then logically carries on into the next iterative phase of the project that uses divergence and convergence once more by building upon the previous phase's solutions and work to explore the next part of the problem in-depth.

In the case of this project, the methodology's initial two phases, "discover", and "define", will be consist of a preliminary study where a decision will be made regarding the style of the workflow going forward, and all checkpoint models to be used will be finalised. This will be expanded upon in the next section this chapter. The latter two phases, "develop" and "deliver"

will consist of the final study, where five different strategies will be implemented and evaluated to generate consistent characters. Ultimately, the results of the evaluation will be delivered along with the final workflow proposal.

3.2. Research Approach

3.2.1. Preliminary Study

The preliminary study is concerned with the choice of the models style to take forward for the final workflow. That is, there are two ways for artists to go around generating comics. The first way is to generate an entire comic book page, complete with panelling and layouts, whereas the second way is to generate each individual panel separately. A panel is a rectangular, sub-image within a comic page, with a black boxed outline. As seen, there are five panels in the second page of figure 3.2, so using the second method, five different panels would have to be created and then edited together into one page using an image-editing software. Thus, the second method requires manually putting together multiple generated images in a program like Photoshop.

Fig. 3.2: Two comic book pages

Figure 3.2 depicts two comic book pages laid out side-by-side. The first method would aim to create either one of these two images (as an example) in a single generation. The second method would aim to create every single panel separately.

This preliminary study will consist of the discover and define phases of the Double Diamond approach. For the "discover" phase, code will be developed and adapted from tutorials and existing online repositories to undertake a DreamBooth fine-tuning of SD 1.5. See chapter 2, section 2.3.1 for the decision of using this model. This phase will also entail the gathering, processing, and captioning of the data for the fine-tuning process. This first phase is the divergence segment of the divergence and convergence approach for the first diamond of the methodology.

The "define" phase will then act as the convergence segment, wherein the data prepared will be used to fine-tune two different SD 1.5 models, one to generate images of pages, and another to generate images of panels. The pages model will be trained solely on data of 100 comic book pages and the panels model will be trained on pictures of 100 panels. All images will be processed so they are 512 by 512 pixels in size. The data will be gathered and sourced from the Invincible digital comics (Kirkman, et al., 2023), available for purchase online. As per current copyright laws, training a diffusion model on comic data for academic purposes falls under fair dealing (Intellectual Property Office, 2021), and should not result in a copyright violation as long as the images gathered are for a non-profit academic project and are not publicly distributed. This "define" phase will be iterative in nature, as selecting the best values for the different DreamBooth training parameters will mean different efficacies for the resultant model. Expectations are that the captioning and labelling of the training data will need to be modified several times since the text descriptions for each training image determines the prompt engineering that will need to be done when using the model later. The captioning is also important because the model learns the associations between the caption's words and the content of the accompanying image, so the choice of words and syntax used impact the model' learning.

Once the two models are finalised, they will be evaluated against each other by each generating 20 images, 10 using a rough sketch—5 with img2img, 5 with ControlNet—and 10 using plain text prompts. The only metric being compared between the two models' outputs is the coherency of the image. Essentially, whether the image output is easy to understand from the perspective of a reader, and if the scenario conveyed through the picture actually matches the description of the prompt. The better-performing model will decide the generation style for the final workflow. An additional two checkpoint models will then be downloaded from CivitAI for

the next two phases of the final study. This is required because it is mandatory to know whether the character consistency strategies implemented are able to be generalised across different models. The whole point of the project is to let the user have as much control and freedom as possible, so naturally, different artists will likely have their own favourite checkpoint models to use as the base.

3.2.2. Final Study

The final study will consist of the develop and deliver phases, and will be aiming to implement and evaluate the following five character-consistency strategies:

- Prompt Engineering and Style Saving
- Img2img Sketch and In-Painting
- ControlNet Scribble
- ControlNet Reference
- LoRA Training

All strategies will be elaborated upon in chapter 6.

The first, divergent phase, "develop", will consist of an initial research, data gathering, and preparatory work required to set up all the environments and data, and also the implementation of the strategies. This is where character reference sheets will be generated using existing checkpoint models downloaded in the preliminary study, rough sketches will be drawn to test the different strategies on, and training data for the LoRA will be prepared. This data is very similar to the data prepared for DreamBooth training, except with fewer images, as character LoRAs usually need around 5 to 30 images for good results. The images gathered will need to be resized to 512 by 512 and captioned using text descriptions as well. This training will also be iterative since modifying the captions data, training images, or training parameters could result in improvements to the final model.

The final phase of the project, "deliver", will consist of the implementation and evaluation of the consistency strategies. For this, 40 images will be generated using each of the 5 strategies. Then these images will be compared against 5 different success criteria, which were selected after careful consideration of what techniques are required for the smooth usage of generative AI for art (u/GrodanHej, 2024) (u/not-me-374892, 2022) (u/dicklim39, 2023) (u/wonderflex, 2022).

Thus, the decided success criteria were the following:

- Character Consistency (clothing, hair, body type, facial features, accessories)
- Presence or lack of Artifacts (presence of accidental distortions and mistakes)
- Fine-grained Control (tendency to closely follow all details in the prompt)
- Coherency (ease of understanding what the image is depicting)
- Background Consistency (more lenient on this aspect, as the strategies do not aim to maximise this)

Character consistency was chosen as it is the main aim of this project. The presence or lack of artifacts and coherency were then chosen because these are known problems impacting the performance of diffusion models (Various Reddit users, 2023), and if these problems persist in the workflow proposed by this project, it will naturally make generating clean artwork an issue. Fine-grained control was chosen as it is in the best interest of the artist to retain as much creative control over the final output as possible. This is because if the models' outputs don't match the artists' creative vision, they might drop trying to use the workflow entirely. Finally, background consistency is also important as if the environment and setting of the story in the comic changes every panel or page, it will not make for a very immersive experience for the readers.

Each strategy will be rated out of 5 based on how many of their 40 images pass the success criteria. For example, all images being coherent for img2img, would give it a 40/40 score on Coherency, which evaluates to a 5 out of 5 rating. If only 32 out of 40 images were coherent, the score would be 4/5.

Formula: `round_to_1_decimal(images_passed / (40 / 5))`

The final workflow will be proposed based on the best performing strategies across all success criteria.

3.3. Project Management and Tracking

The four phases in the project will be broken down into key tasks and tracked using Trello (Johnson, 2017) with a simple Kanban board technique (Dalton, 2018) along with a Gantt chart (Ramachandran & Karthick, 2019). These are primarily used for time management, as having a visual indicator of what tasks need to be completed, what is being worked on in the moment, and what has been completed to date is a very helpful to put things in context for the big picture while also providing focus for the task at hand. See appendix C.1 for screenshots of the Trello board used, and appendix F for a screenshot of the Gantt chart.

Python will be used for the entire project, with the code written in machine learning notebooks (.ipynb files). The author's personal computer is not powerful enough to run the training and image generation sessions, so Google Colab will be used (Google, 2017), and the .ipynb notebooks will be stored on Google Drive. All Colab files stored on Drive come with automatic file versioning and tracking, and all changes made to the code are automatically tracked. As such, Git will not be used here. At the end of the project, the plain notebooks without any output data will be uploaded to a GitHub repository along with the code written for the preprocessing of images (resizing and cropping images to fit the 512 by 512px requirement for SD). This preprocessing code will be written using VS Code and tracked using GitHub (Microsoft, 2023) (GitHub, 2023).

Finally, all of the trained models and metadata will be uploaded to HuggingFace (Hugging Face, 2024). This is a website that acts like GitHub for machine learning models and their metadata. Most diffusion model checkpoints are very large (2GB to 20GB) and are not feasible to upload to Git. Instead, sites like HuggingFace are used.

3.4. Ethical Considerations

Initially, the project's evaluation involved asking human participants to rate and evaluate the character consistency strategies and their corresponding output images, so the ethics application for the project to the Research Ethics Committee was requested as requiring human participants. The project received approval for this, and the ethics letter is attached in the Appendix. However, the scope later shifted since it was more logical to analyse the images using success criteria instead, so human participants were not involved at all. As such, this project was transformed into a No Ethics Required project. Additional approval was not obtained, as approval had already been received for a higher-risk ethics application.

4. Style Model Training

4.1. Introduction

This chapter will go over the work done for the “discover” and “define” phases of the project—the preliminary study. As described in the prior chapter, the preliminary study entails the creation of two models—the pages model and the panels model. Thus, this is what will be referred to as the “style” training for this project, as the model found in this study to be better performing will be chosen as the style and structure of the project going forward—that is, whether the character consistency strategies of the final study will be applied to page generation or panel generation.

4.2. Design and Data

4.2.1. Overview

As discussed in the methodology chapter, the purpose of this preliminary study is to make a decision about whether the workflow would be better suited to generate entire comic book pages, or if the slower approach of generating each panel individually is worth the effort. To do this, two SD 1.5 models will be trained, the first on 100 pages of the Invincible comics, and the second on 100 panels of the same comic.

DreamBooth was set up in a .ipynb notebook after following tutorials online and helpful advice from the AI art communities on discord. The ML notebooks for DreamBooth training by TheLastBen and ShivamShrirao were used as references (TheLastBen, 2023b) (Shrirao, 2023), and the functionalities that were required were adapted into the notebook for this project. Additionally, code from other sources was also used in helping build a different notebook to run Stable Diffusion using the Automatic1111 Web UI from Google Colab in a gradio instance. Both of these notebooks would together allow both fine-tuning SD 1.5 and running it to generate images using Google’s GPUs.

Once the notebook was set up and the code was tested, the DreamBooth training process was ready to be started. More on this in section 4.3.

4.2.2. Data Gathering, Processing, and Captioning

The data gathering and processing consumed as much time as iteratively training the models later took. The first couple dozen issues of the Invincible digital comics were purchased a few years ago for non-academic purposes. As those images were available on hand, they worked best as training data. As addressed last chapter, this counted as fair dealing.

When scoping out which pages to use as training data, certain criteria were considered. That is, since the model requires as much varied data as possible, differences in pages such as number of panels, singular panel pages, wide angle shots, amount of text and speech bubbles, character expressions, close up shots, interior shots, outdoor shots, daytime, nighttime, talking shots, action, fighting, different lighting shots, etc. were all taken into account. The images for the 100 panels were also selected in a similar manner.

Once the data was gathered, it needed to be processed for SD 1.5. All images needed to be 512 by 512px. This was straightforward to accomplish using online tools for the panels dataset, as tools like Presize (atinylittleshell, 2023) allow the user to resize and scale their images in bulk. The tool was used to crop the pages in such a way that only a single panel was in the shot. It then automatically resized all images.

This tool did not work for full pages however, as there were now three main requirements: that the pages not be cropped and fully visible, that the left and right side of the pages be filled with white gutter space to match the natural layout of the pages, and that the quality not be diminished too much, since lower resolution was a lot more noticeable for pages than singular panels. Therefore, to accomplish all of this, I needed to write my own python code to process the pages and convert them into a format best digestible by SD 1.5. The code was written, tested, and pushed to git.

Figure 4.1 and 4.2 below depict the code written to preprocess the page images collected. As seen, it was first checked if the image was already square, because if so, then the image only needed to be resized. Otherwise, the image first needed to be made into a square by adding white gutter space. The gutter space needed to be accurately calculated, then added to the image. Finally, the image could be resized and saved. Figure 4.3 depicts pages with white gutter space added.

```python
def resize_image(filepath):
    img = Image.open(filepath)
    final = img.copy

    if not is_square(img):
        final = change_to_square(img)

    return resize_square_image(final)


def resize_all():
    input_path = 'input-images/'
    img_quality = 100
    imgs = os.listdir(input_path)

    for img in imgs:
        res = resize_image(input_path + img)
        res.save('output-images/' + img, 'PNG', quality=img_quality)
        print('Completed image:', img)

    print('COMPLETED ALL!')


if __name__ == '__main__':
    resize_all()
```

Fig. 4.1: Preprocess Pages Code (1)

```python
def is_square(img):
    return img.size[0] == img.size[1]


def calculate_gutters(img):
    return round((max(img.size) - min(img.size)) / 2)


def do_padding(img, padding, pad_x):
    w = max(img.size)
    h = w  # since we're making the img square

    res = Image.new(img.mode, (w, h), (255, 255, 255))
    left, top = 0, 0

    if pad_x:
        top = padding
    else:
        left = padding

    res.paste(img, (left, top))
    return res


def change_to_square(img):
    # if not pad_x, then pad y. Increase width along that specific axis.
    pad_x = img.size[0] > img.size[1]
    padding = calculate_gutters(img)
    return do_padding(img, padding, pad_x)


def resize_square_image(img, target_size = 1024):
    return img.resize((target_size, target_size), Image.LANCZOS)
```

Fig. 4.2: Preprocess Pages Code (2)

Once all 100 images for both datasets were ready—200 images in all—they were ready to be captioned. Captioning the images was, by far, the most time-consuming process in the data processing phase. Captioning entailed writing textual descriptions for every single image in English in either around 2-3 detailed and descriptive sentences, or using multiple phrases to capture the overall mood, context, and content of the scene, such as “clouds, clear sky, grey backgrounds, daytime, blue lighting, stormy weather, city, landscape, flying, costume” and so on. Great attention needed to be paid to make sure all aspects of an image were captured by these captions so that the model would learn the associations correctly.

Figures 4.3 and 4.4 depict two pages and two panels respectively, along with the captions that needed to be written for them. The syntax structure used was a combination of English sentences, and short phrases, a style that is common among enthusiasts in the AI art communities online. The captions are part of the image in the figures below, however when training, each image is provided separately, and the caption for it is provided in an accompanying text file with the same name as the image file—for example, an image, “comicpage-12.jpg” would have its caption in a text file, “comicpage-12.txt”.

comicpage with a single panel, full page shot, man talking, man looking ahead, speaking while gesturing, close up shot of head, shoulders and hand, interior shot, yellow lighting

comicpage with five panels, 1boy, shirtless boy in bedroom, laying on bed and staring at the ceiling, sitting up on bed, walking up to the bedroom window, blue lighting, interior shot, lying on bed

Fig. 4.3: Two resized and processed comic pages with text captions

a cmcpl of a close up shot of a boy in a blue shirt clicking an earpiece with his index and middle finger while speaking, sitting in a lecture auditorium, benches, pen in hand, 1boy, interior shot

a cmcpl of a boy in a red shirt and grey pants flying in the sky with a sad, sullen expression, city below, buildings and trees, flying in the sky, clouds, moon, night time, outside, blue lighting, 1boy

Fig. 4.4: Two resized and processed comic panels with text captions

4.3. Implementation and Evaluation

4.3.1. Automatic1111 Web UI Colab Implementation

The majority of this project relies upon running stable diffusion as the entire point was to optimise a number of different training and image generation strategies. But since a powerful GPU was required and Google Colab was to be used, the standard implementation of the Automatic1111 Web UI interface was unusable and needed to be set up specially in order to be run in a Google Colab environment. Fortunately, several instances of Colab notebooks online were available to use as references (catboxanon, 2023). However, some of them, such as (camenduru, 2023), required reinstallation of all the Web UI files any time the notebook was run, and all the generations and models were left in temporary Google Drive storage that would be wiped upon the Colab GPU runtime being disconnected. Thus, a notebook that maintained persistent files within Google Drive was used as the main reference (TheLastBen, 2023a). This notebook contained a lot of extraneous code that was not required for the execution of this project, so the other notebooks were used in conjunction to adapt only the code that was needed to run SD 1.5 as required for the aims of this project.

Figure 4.5 displays the code required to install the A1111 repository to Google Drive for the first time in a persistent manner, as well as the installation of the dependencies for the

execution environment needed to run the project. Figure 4.6 downloads SD 1.5 to use as the base for generating images, if the model already does not exist in the files within Google Drive. Thus, this download only needed to be completed once, and any subsequent times the Web UI was run, the model would be automatically detected and loaded from Drive.

Figure 4.7 depicts part of the code required to install ControlNet, again, only a procedure that needed to be completely just once. The links all point to the original trained ControlNet models by the first author of the ControlNet paper (Illyasviel, 2023) (Zhang, et al., 2023). Figure 4.8 contains all the necessary to finally run the repository once all the installations have been completed and will open an instance of the Web UI in a new tab. The complete version of the code is available in an ML notebook linked in Appendix C.2. This same ML notebook created to run the Web UI will be used in all remaining chapters of the dissertation, as all of the implementation and evaluation required henceforth just makes use of the base SD Web UI and its surrounding ecosystem.

```
[ ]  # install or update the A1111 repo

     if not os.path.exists('/content/gdrive'):
         print('please connect to google drive before proceeding')


     with capture.capture_output() as cap:
         fgitclone = "git clone --depth 1"
         %mkdir -p /content/gdrive/MyDrive/sd
         %cd /content/gdrive/MyDrive/sd
         !git clone -q --branch master https://github.com/AUTOMATIC1111/stable-diffusion-webui
         !mkdir -p /content/gdrive/MyDrive/sd/stable-diffusion-webui/cache/
         os.environ['TRANSFORMERS_CACHE'] = f"/content/gdrive/MyDrive/sd/stable-diffusion-webui/cache"
         os.environ['TORCH_HOME'] = f"/content/gdrive/MyDrive/sd/stable-diffusion-webui/cache"
         !mkdir -p /content/gdrive/MyDrive/sd/stable-diffusion-webui/repositories
         !git clone https://github.com/AUTOMATIC1111/stable-diffusion-webui-assets /content/gdrive/MyDrive/sd/stable-diffusion-webu

     with capture.capture_output() as cap:
         %cd /content/gdrive/MyDrive/sd/stable-diffusion-webui/
         !git reset --hard
         !git checkout master
         time.sleep(1)
         !rm webui.sh
         !git pull
     print('Completed A1111 Repo Download/Update!')
```

```
[ ]  # Requirements

     print('Installing requirements...')

     with capture.capture_output() as cap:
         %cd /content/
         !wget -q -i https://raw.githubusercontent.com/TheLastBen/fast-stable-diffusion/main/Dependencies/A1111.txt
         !dpkg -i *.deb

         if not os.path.exists('/content/gdrive/MyDrive/sd/stablediffusion'):
             !tar -C /content/gdrive/MyDrive --zstd -xf sd_mrep.tar.zst
         !tar -C / --zstd -xf gcolabdeps.tar.zst
         !rm *.deb | rm *.zst | rm *.txt
         !pip install spandrel==0.1.6 --qq

         if not os.path.exists('gdrive/MyDrive/sd/libtcmalloc/libtcmalloc_minimal.so.4'):
             %env CXXFLAGS=-std=c++14
             !wget -q https://github.com/gperftools/gperftools/releases/download/gperftools-2.5/gperftools-2.5.tar.gz && tar zxf g
             !wget -q https://github.com/TheLastBen/fast-stable-diffusion/raw/main/AUTOMATIC1111_files/Patch
             %cd /content/gperftools
             !patch -p1 < /content/Patch
             !./configure --enable-minimal --enable-libunwind --enable-frame-pointers --enable-dynamic-sized-delete-support --enab
             !mkdir -p /content/gdrive/MyDrive/sd/libtcmalloc && cp .libs/libtcmalloc*.so* /content/gdrive/MyDrive/sd/libtcmalloc
```

Fig. 4.5: A1111 Installation and Requirements

```
# Model Download/Load
# Download Stable Diffusion 1.5 as the base model if it does not exist

def download_sd1_5():
    version = '1.5'
    model = '/content/gdrive/MyDrive/sd/stable-diffusion-webui/models/Stable-dif
    link = 'https://huggingface.co/runwayml/stable-diffusion-v1-5/resolve/main/v

    if not os.path.exists(model):
        !gdown --fuzzy -O $model $link
        if os.path.exists(model):
            print('Completed!')
        else:
            print('ERROR! Something went wrong.')
    else:
        print('Model already exists!')

    return model


model = download_sd1_5()
```

Fig. 4.6: A1111 Model Download/Load

```
with open("CN_models.txt", 'r') as f:
    mdllnk = f.read().splitlines()

!rm CN_models.txt


# ControlNet model Links
Canny='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11p_sd15_canny.pth'
Depth='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11f1p_sd15_depth.pth'
Lineart='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11p_sd15_lineart.pth'
MLSD='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11p_sd15_mlsd.pth'
Normal='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11p_sd15_normalbae.pth'
OpenPose='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11p_sd15_openpose.pth'
Scribble='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11p_sd15_scribble.pth'
Seg='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11p_sd15_seg.pth'
ip2p='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11e_sd15_ip2p.pth'
Shuffle='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11e_sd15_shuffle.pth'
Inpaint='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11p_sd15_inpaint.pth'
Softedge='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11p_sd15_softedge.pth'
Lineart_Anime='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11p_sd15s2_lineart_anime.pth'
Tile='https://huggingface.co/lllyasviel/ControlNet-v1-1/resolve/main/control_v11f1e_sd15_tile.pth'


if v1_Model == "All (21GB)":
    for lnk in mdllnk:
        download(lnk, mdldir)

elif v1_Model == "T2iadapter_Models":
    mdllnk = list(filter(lambda x: 't2i' in x, mdllnk))
    for lnk in mdllnk:
        download(lnk, mdldir)

elif v1_Model == "None":
    pass
    print('Nothing to download.')

else:
    download(globals()[v1_Model], mdldir)
    print('Completed downloading ControlNet successfully!')
```

Fig. 4.7: A1111 ControlNet Installation

```
[ ]  # Start Stable Diffusion

     from IPython.utils import capture
     import time
     import sys
     import fileinput
     import re


     with capture.capture_output() as cap:
         %cd /content/gdrive/MyDrive/sd/stable-diffusion-webui/modules/
         !wget -q -O extras.py https://raw.githubusercontent.com/AUTOMATIC1111/stable-diffusion-webui/master/modules/extras.py
         !wget -q -O sd_models.py https://raw.githubusercontent.com/AUTOMATIC1111/stable-diffusion-webui/master/modules/sd_models.py
         !wget -q -O /usr/local/lib/python3.10/dist-packages/gradio/blocks.py https://raw.githubusercontent.com/TheLastBen/fast-stable-d
         %cd /content/gdrive/MyDrive/sd/stable-diffusion-webui/

         !sed -i 's@shared.opts.data\["sd_model_checkpoint"] = checkpoint_info.title@shared.opts.data\["sd_model_checkpoint"] = checkpoi
         !sed -i "s@map_location='cpu'@map_location='cuda'@" /content/gdrive/MyDrive/sd/stable-diffusion-webui/modules/extras.py

         !sed -i 's@possible_sd_paths =.*@possible_sd_paths = [\"/content/gdrive/MyDrive/sd/stablediffusion\"]@' /content/gdrive/MyDrive
         !sed -i 's@\.\.\/@src/@g' /content/gdrive/MyDrive/sd/stable-diffusion-webui/modules/paths.py
         !sed -i 's@src/generative-models@generative-models@g' /content/gdrive/MyDrive/sd/stable-diffusion-webui/modules/paths.py

         !sed -i 's@print(\"No module.*@@' /content/gdrive/MyDrive/sd/stablediffusion/ldm/modules/diffusionmodules/model.py
         !sed -i 's@\["sd_model_checkpoint"\]@\["sd_model_checkpoint", "sd_vae", "CLIP_stop_at_last_layers", "inpainting_mask_weight", "


     ckptdir = ''

     try:
         model
         if os.path.isfile(model):
             !python /content/gdrive/MyDrive/sd/stable-diffusion-webui/webui.py --share --api --disable-safe-unpickle --enable-insecure-
         else:
             !python /content/gdrive/MyDrive/sd/stable-diffusion-webui/webui.py --share --api --disable-safe-unpickle --enable-insecure-
     except:
         !python /content/gdrive/MyDrive/sd/stable-diffusion-webui/webui.py --share --api --disable-safe-unpickle --enable-insecure-exte
```

Fig. 4.8: Run the A1111 Web UI

4.3.2. Iterative Training of Models

There are multiple tutorials and Google Colab notebooks online that demonstrate how to implement the code required to carry out a DreamBooth fine-tuning process. Most of these demos were implemented to focus on a particular type of fine-tuning, such as training faces, people, or just producing a comprehensive ML notebook that had options for training multiple models with a variety of styles and options. The majority of this was not needed, with this project requiring special functionalities and options. Thus, the code for the DreamBooth process for this project was implemented and adapted from the processes of three different notebooks (TheLastBen, 2023b) (Shrirao, 2023) (Penna & Bielejeski, 2023). The former notebook was the main source used, while the latter two were used to optimise options particular to this project.

The first few cells of the notebook were a few lines used to check the Colab GPU runtime and to connect the Google Drive account of the author. Figure 4.9 then shows the first key section of the code that was written to install all the dependencies required to perform the fine-tuning process. The dependencies were listed in the repository of one of the earlier cited notebooks that was used as a reference, so a direct installation using those dependencies could be performed. Next, in figure 4.10 was the model download, which downloaded stable diffusion

1.5 as the base model which would be used for the fine-tuning process. As used in the prior cited notebooks, a PyTorch diffusion model also needed to be used to aid in the training process.

```
[ ]  # Dependencies

     print('Installing dependencies...')

     with capture.capture_output() as cap:
         os.chdir('/content')
         !pip install -qq --no-deps accelerate==0.12.0
         !wget -q -i https://raw.githubusercontent.com/TheLastBen/fast-stable-diffusion/main/Dependencies/dbdeps.txt
         !dpkg -i *.deb
         !tar -C / --zstd -xf gcolabdeps.tar.zst
         !rm *.deb | rm *.zst | rm *.txt
         !git clone -q --depth 1 --branch main https://github.com/TheLastBen/diffusers
         !pip install gradio==3.16.2 --no-deps -qq

         if not os.path.exists('gdrive/MyDrive/sd/libtcmalloc/libtcmalloc_minimal.so.4'):
             %env CXXFLAGS=-std=c++14
             !wget -q https://github.com/gperftools/gperftools/releases/download/gperftools-2.5/gperftools-2.5.tar.gz
             !wget -q https://github.com/TheLastBen/fast-stable-diffusion/raw/main/AUTOMATIC1111_files/Patch
             %cd /content/gperftools
             !patch -p1 < /content/Patch
             !./configure --enable-minimal --enable-libunwind --enable-frame-pointers --enable-dynamic-sized-delete-s
             !mkdir -p /content/gdrive/MyDrive/sd/libtcmalloc && cp .libs/libtcmalloc*.so* /content/gdrive/MyDrive/sd.
             %env LD_PRELOAD=/content/gdrive/MyDrive/sd/libtcmalloc/libtcmalloc_minimal.so.4
             %cd /content
             !rm *.tar.gz Patch && rm -r /content/gperftools
         else:
             %env LD_PRELOAD=/content/gdrive/MyDrive/sd/libtcmalloc/libtcmalloc_minimal.so.4

         os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
         os.environ['PYTHONWARNINGS'] = 'ignore'

     print('Installation completed successfully!')
```

Fig. 4.9: DreamBooth Dependencies Installation

Figures 4.11 and 4.12 next show the sections that were used to upload the training data and the creation of a session respectively. The 100 images and captions created earlier could now be supplied to DreamBooth the code written here. The session creation was so that a prior training session could be restarted if anything had gone wrong during the process. This was required as the training process is very long, sometimes lasting more than an hour or two. If internet connection was lost during that time, the base 1.5 model and training data supplied would not have to be re-implemented due to the session being saved.

```
[ ] # Model Download
    # Skip this cell when loading a previous session that contains a trained model

    with capture.capture_output() as cap:
        os.chdir('/content')

    if os.path.exists('/content/gdrive/MyDrive/Dreambooth-Training/token.txt'):
        with open("/content/gdrive/MyDrive/Dreambooth-Training/token.txt") as f:
            token = f.read()
        auth = f'https://USER:{token}@'
    else:
        auth = "https://"


    def download_model():
        if os.path.exists('/content/stable-diffusion-v1-5'):
            !rm -r /content/stable-diffusion-v1-5

        os.chdir('/content')

        !mkdir /content/stable-diffusion-v1-5
        os.chdir('/content/stable-diffusion-v1-5')
        !git config --global init.defaultBranch main
        !git init
        !git lfs install --system --skip-repo
        !git remote add -f origin  "https://huggingface.co/runwayml/stable-diffusion-v1-5"
        !git config core.sparsecheckout true
        !echo -e "scheduler\ntext_encoder\ntokenizer\nunet\nvae\nmodel_index.json\n!vae/diffusion_pytorch_model.bin\n!*.safetenso
        !git pull origin main
        if os.path.exists('/content/stable-diffusion-v1-5/unet/diffusion_pytorch_model.bin'):
            !wget -q -O vae/diffusion_pytorch_model.bin https://huggingface.co/stabilityai/sd-vae-ft-mse/resolve/main/diffusion_p
            !rm -r .git
            !rm model_index.json
            time.sleep(1)
            wget.download('https://raw.githubusercontent.com/TheLastBen/fast-stable-diffusion/main/Dreambooth/model_index.json')
            os.chdir('/content')
            print('Done!')
        else:
            while not os.path.exists('/content/stable-diffusion-v1-5/unet/diffusion_pytorch_model.bin'):
                print('Something went wrong...')
                time.sleep(5)


    if not os.path.exists('/content/stable-diffusion-v1-5'):
        download_model()
        MODEL_NAME = "/content/stable-diffusion-v1-5"
    else:
        MODEL_NAME = "/content/stable-diffusion-v1-5"
        print("The v1.5 model already exists, using this model.")
```

Fig. 4.10: DreamBooth Model Download

```
[ ] # Instance Images

    # Set to False to keep any existing instance images.
    remove_existing_instance_images = False

    # instance images will be taken from the directory specified.
    images_folder = '/content/gdrive/MyDrive/Uni Work/third year/FYP/training data/panels/sep'

    # Make sure to upload the captions separately into the captions folder.


    with capture.capture_output() as cap:
        %cd /content


    if remove_existing_instance_images:
        if os.path.exists(str(INSTANCE_DIR)):
            !rm -r "$INSTANCE_DIR"
        if os.path.exists(str(CAPTIONS_DIR)):
            !rm -r "$CAPTIONS_DIR"

    if not os.path.exists(str(INSTANCE_DIR)):
        %mkdir -p "$INSTANCE_DIR"
    if not os.path.exists(str(CAPTIONS_DIR)):
        %mkdir -p "$CAPTIONS_DIR"

    if os.path.exists(INSTANCE_DIR+"/.ipynb_checkpoints"):
        %rm -r $INSTANCE_DIR"/.ipynb_checkpoints"



    while images_folder != "" and not os.path.exists(str(images_folder)):
        print('The images folder specified does not exist, copy the path in here:')
        images_folder = input('')
```

Fig. 4.11: DreamBooth Instance Images Upload

```
# Create/Load a Session

# ======================================================================================
# Enter the session name.
# If an older session with the same name exists, that will be loaded. Otherwise a new session will be made.
Session_Name = 'comicpanel2'


try:
    MODEL_NAME
    pass
except:
    MODEL_NAME = ""

PT = ""

while Session_Name == "":
    print('Input the Session Name:')
    Session_Name = input('')
Session_Name = Session_Name.replace(" ", "_")


INSTANCE_NAME = Session_Name
OUTPUT_DIR = "/content/models/" + Session_Name
SESSION_DIR = '/content/gdrive/MyDrive/Dreambooth-Training/Sessions/' + Session_Name
INSTANCE_DIR = SESSION_DIR + '/instance_images'
CAPTIONS_DIR = SESSION_DIR + '/captions'
MDLPTH = str(SESSION_DIR + "/" + Session_Name + '.ckpt')

if os.path.exists(str(SESSION_DIR)):
    mdls = [ckpt for ckpt in listdir(SESSION_DIR) if ckpt.split(".")[-1] == "ckpt"]
```

Fig. 4.12: DreamBooth Creating or Resuming a Session

```
# Training
#Start DreamBooth


# Set this to True to continue training the prior model.
Resume_Training = False

# ==============
# Training parameters

UNet_Training_Steps = 3000

# UNet_Learning_Rate = 2e-6
UNet_Learning_Rate = 1e-5
untlr = UNet_Learning_Rate

Text_Encoder_Training_Steps = 900

# keep low to avoid overfitting (1e-6 is higher than 4e-7)
# Text_Encoder_Learning_Rate = 1e-6
Text_Encoder_Learning_Rate = 2e-5
txlr = Text_Encoder_Learning_Rate

# Always set as True for style training. Not needed for faces.
Offset_Noise = True
# ==============



if os.path.exists(INSTANCE_DIR + "/.ipynb_checkpoints"):
    %rm -r $INSTANCE_DIR"/.ipynb_checkpoints"

if os.path.exists(CAPTIONS_DIR+"/.ipynb_checkpoints"):
    %rm -r $CAPTIONS_DIR"/.ipynb_checkpoints"
```

Fig. 4.13: DreamBooth Training Parameters

```
def dump_only_textenc(trnonltxt, MODELT_NAME, INSTANCE_DIR, OUTPUT_DIR, PT, Seed, precision, Training_Steps):
    !accelerate launch /content/diffusers/examples/dreambooth/train_dreambooth.py \
    $trnonltxt \
    --external_captions \
    $ofstnse \
    --image_captions_filename \
    --train_text_encoder \
    --dump_only_text_encoder \
    --pretrained_model_name_or_path="$MODELT_NAME" \
    --instance_data_dir="$INSTANCE_DIR" \
    --output_dir="$OUTPUT_DIR" \
    --captions_dir="$CAPTIONS_DIR" \
    --instance_prompt="$PT" \
    --seed=$Seed \
    --resolution=$TexRes \
    --mixed_precision=$precision \
    --train_batch_size=1 \
    --gradient_accumulation_steps=1 --gradient_checkpointing \
    --use_8bit_adam \
    --learning_rate=$txlr \
    --lr_scheduler="linear" \
    --lr_warmup_steps=0 \
    --max_train_steps=$Training_Steps
```

Fig. 4.14: Invoking the DreamBooth Training Process

Figures 4.13 and 4.14 show sections of the code used for setting the training parameters and invoking the training process respectively. A link to the full code is available in Appendix C.2.

The code implementation that was now complete was just the beginning of the actual implementation for the preliminary study, with the bulk of the remaining work here being the tuning of the training data, the training parameters, and iterating over the training process as many times as needed until the results were satisfactory.

Two main issues existed due to which the fine-tuning processes needed to be ultimately repeated and iterated over five times in total. As expected, finding the right training parameters was the primary concern, and needing to re-caption the training data was also very important so that the model better learned associations and trigger words. To address the second point first, during each iteration, the 100 captions written for each of the pages were intricately edited and re-written to ensure that any problems in the prior iteration were addressed in the new one. For example, additionally referring to superheroes as characters in costumes, and having a count of the number of different character types, such as “1boy”, “2men”, “1superhero”, etc. Another example is seen in figures 4.3 and 4.4. Figure 4.3 shows the captions starting with the words “comicpage with…”. This is not the best approach when training, because SD has already encountered tokens for the words “comic” and “page” before. Thus, the best approach seems to be when a completely new token is used to train a new concept. This is why in figure 4.4 the captions start with the words, “a cmcpl of…”. By the time of captioning the panels model, this lesson was already learnt, and instead of using a potentially pre-used token “comic panel”, an abbreviation, “cmcpl” was used. Similarly, all occurrences of “comicpage” in the captions for the pages dataset needed to be replaced with an abbreviation, which was set as “cmcpg”.

Regarding the training parameters, the key ones were the two learning rates for the text encoder and U-Net model, and the training steps for both the text encoder and the U-Net. These are typically set to very small numbers in most tutorials, such as 300 steps for the text encoder, and a learning rate of 2e-6 for the U-Net. However, after several iterations, the best results were achieved by a much higher number for all parameters, increasing the text encoder steps to 1200 and the U-Net learning rate to 1e-4. The discovery was that this project was training the models on a "style", whereas most tutorials simply trained them on people's faces, which required far fewer images in the dataset and far lower thresholds for the training parameters. As these datasets brought in a 100 images each and trained an abstract concept such as comic pages and panels, the parameters needed to be set a lot higher. Ultimately, the results were decent after 5 iterations, whereas the results of the first couple of models were atrocious—completely and utterly distorted, with no way of telling what was supposed to be happening in those images.

All trained model checkpoints and resulting metadata were uploaded to HuggingFace and are available to view from Appendix D.

4.3.3. Model Evaluation and Final Model Selection

As devised in the methodology section, the models were evaluated by comparing their outputs in terms of coherency achieved. 40 images were generated in total, 20 for the model trained on the pages dataset, and 20 on the panels dataset. Of these 20, 10 images for each model were generated using plain text prompts, whereas the remaining 10 were generated using an additional hand-drawn rough sketch as input, in the hopes of mimicking an artist's general artistic process. Of these 10, 5 were generated using img2img, and the remaining 5 using ControlNet. No major differences were noticed when using the two here since generation parameters were kept constant between both ControlNet and img2img. This was so that both tools had the opportunity to display their non-cherry-picked outputs in order to judge the models fairly.

Firstly, the following are the best two results of the pages model and the panels model from the 10 images that were generated for each model based on plain text prompts:

Pages prompt: *cmcpg with three panels, fighter jets flying over destroyed city, while a lone man standings in the middle of the city and looks at the destruction, close up shot of head, standing, fallen buildings, debris, smoke*

Fig. 4.15: Two best generated pages using the trained pages model

Figure 4.15 shows the two best results of the pages model out of the 10 images generated using the prompt provided above the figure.

Panels prompt: *a cmcpl of a destroyed city, a superhero in costume overlooks the city from a building, daytime, fallen buildings, debris, destruction*

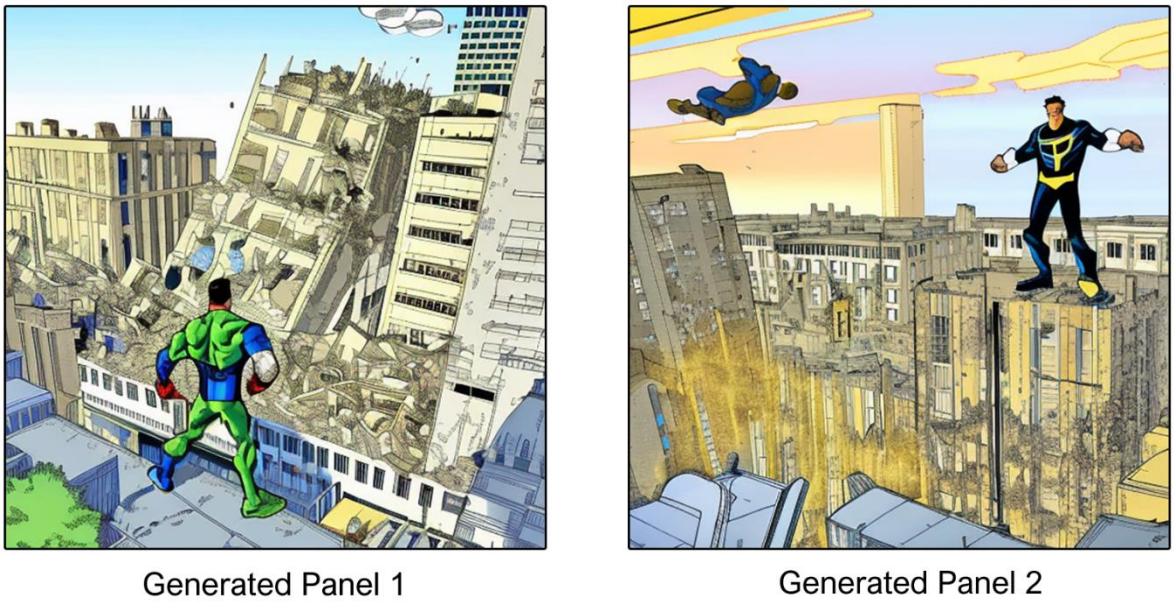


Fig. 4.16: Two best generated panels using the trained panels model

Figure 4.16 shows the two best results of the panels model out of the 10 images generated using the prompt provided above the figure.

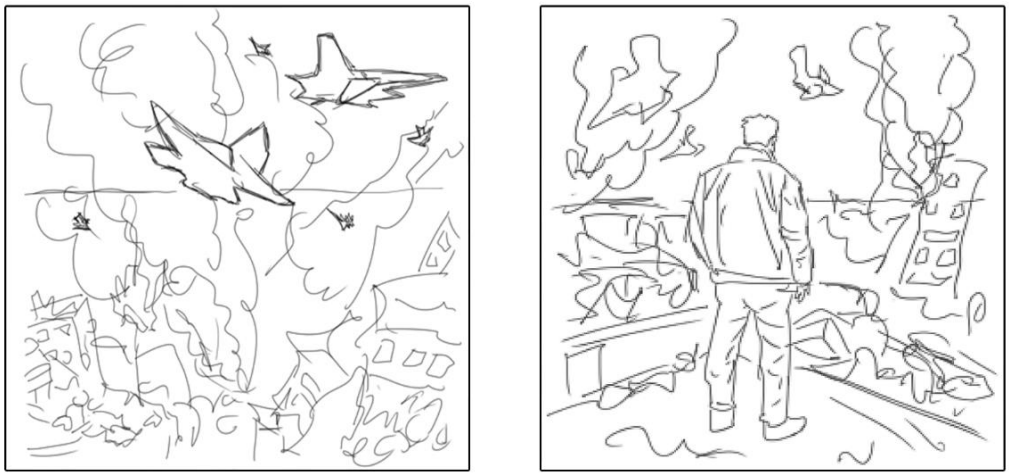
Next, the two best results of the pages model as well as the rough sketch used as an additional prompt (prompt used was the same as the plain text prompt used for the prior images):

Fig. 4.17: Two best generated pages using the trained pages model and an input sketch

Fig. 4.18: Rough sketch used as additional input for the pages model

Figure 4.17 shows the two best results of the pages model when the same plain text prompt from earlier was reused along with the rough sketch in figure 4.18, which was provided as an additional input this time.

Finally, the two best results of the panels model as well as the rough sketches used:



Sketch 1
Prompt: a cmcpl of fighter jets flying over a destroyed city, fallen buildings, rubble, debris, destruction, daytime, smoke, fire

Sketch 2
Prompt: a cmcpl of a man standing on top of a building with his back to the camera, overlooking a destroyed city, fallen buildings, rubble, debris, smoke

Fig. 4.19: Rough sketches used as additional input for the panels model

Generated Image 1

Generated Image 2

Fig. 4.20: Two best generated panels using the trained panels model and different input sketches

Figure 4.19 shows the rough sketches along with the corresponding prompts that were used for testing the panels model along with additional sketch inputs. Sketch 1 and its accompanying prompt resulted in generated image 1 from figure 4.20. Similarly, sketch 2 and its prompt resulted in the generation of image 2 from figure 4.20.

As seen, the panels model seems superior to the pages model when it comes to the coherency of the generated images. It is easier to understand what is supposed to be happening in the pictures of the panels model. This makes sense, as the pages model practically has to generate multiple images for a single page and have them all be coherent, whereas the panels model only needs to generate a single coherent image. Of course, the results for both models seem distorted with many artifacts; this is a consequence of training with only 100 images and is the biggest limitation of this project's scope. For the best results, at least 500 to 1000 images need to have been collected, processed, and captioned in detail. Unfortunately, time constraints did not permit the creation of such a dataset, and finding and downloading such a dataset online was not possible either since distributing comic book images would count as piracy.

The panels model will be selected to go forward with for this project. This means that the artist, if using the workflow developed in this project, will have to generate each panel individually and edit them all together manually. This will be a lot more effort than just generating a single finished page, but as seen, panel-by-panel generation is simply superior and vastly more coherent than generating entire pages at once. This is just a limitation of the state-of-the-art AI for now. But in the future. the coherency of diffusion models when generating complex imagery will likely just keep increasing if the current scaling laws were to continue (Kaplan, et al., 2020) (Hoffman, et al., 2022).

4.3.4. CivitAI Checkpoint Models

Now that the panels model was finalised as the model to be used for the workflow going forward, two additional models with different styles were downloaded from CivitAI. As discussed in chapter 3, this was necessary to make sure that the results of the character consistency study could be generalised across different models for different art styles.

The models downloaded from CivitAI for this purpose were two anime style, booru-tagged models, called AnimePastelDream (Lykon, 2023) and GalenaRedux. Unfortunately, both models are **highly NSFW** with explicit generations possible, and must be used with caution. However, these two were by far the most well-trained checkpoint models based on SD 1.5 and are able to generate a wide variety of characters, backgrounds, and results in anime art style. Furthermore, unless the user triggers the NSFW explicit creations specifically, they do not tend to show up in normal generations.

5. Character Consistency Strategies – Design and Data

5.1. Introduction and Strategies Overview

The final two phases of this project, "develop" and "deliver" make up the final study of the project, as described in the methodology. This is the bulk of the project's research from which conclusions will be drawn and investigates five different strategies to maximise the generation of consistent characters. This chapter will cover only the design and data portion of the final study, whereas chapter 6 will cover the implementation, evaluation, and results in great detail.

The five strategies were created after examining the best ways to leverage the different tools available in the SD ecosystem to help provide as much control over the final outputs as possible. The tools were explored by experimenting with the technology to generate generic images and were initially found through discussions revolving around them in the AI art communities joined during the literature review phase. Additionally, the results being generated by the prior trained pages and panels models from chapter 4 were not nearly as coherent or free of artifacts as compared to the images generated by the anime models downloaded from CivitAI. Thus, the decision was made to primarily use the CivitAI models for testing out the 5 strategies, as testing out the success criteria laid out in the methodology chapter will be far clearer if doing it with the CivitAI models. Naturally, the strategies will **also** be tested on the pages and panels models to ensure the results seen by the anime models are generalised across different models, but the anime models will make up the bulk of the testing.

The five strategies are as follows:

- **Prompt Engineering and Style Saving:** Write a thorough and detailed prompt along with a negative prompt and iteratively test results with small modifications to the prompt until the images output are consistency similar to the character in mind. Once there, use the "Save style" button in the Automatic1111 Web UI, which saves the entire prompt written to a database of saved prompts. All saved prompts can be re-used at any point in the future for any new prompt or generation. Thus, one need only describe a new scenario in a new prompt and inject the saved style from earlier to have the desired character be entered into the scenario.

- **Img2img Sketch and In-Painting:** The next strategy is to leverage img2img's sketch and inpainting categories as described in the background chapter. Img2img's sketch function will be used to transform rough sketches of panels into finished images, and once done, inpainting can then be used to mask out the characters in the panel. The masked-out character/s (single or multiple) can then be transformed into style of character desired, either by writing in a new prompt or leveraging the save style functionality discussed earlier.

- **ControlNet Scribble:** ControlNet's scribble model performs the same functionality as img2img's sketch function, where a rough sketch is transformed into a finished image. However, ControlNet has a big advantage: Scribble is a ControlNet model trained end-to-end and sits on top of SD, acting like a second arbiter over the generation process as compared to img2img which is part of the base stable diffusion model. This may improve results.

- **ControlNet Reference:** This is a feature that allows ControlNet to use an input image as a reference for generating new images: whether it is colours, lighting, characters, backgrounds, or even broad styles like "sci-fi" or "gothic". The strategy here is to feed a character reference sheet to ControlNet Reference to produce images of the same character in new and different scenarios.

- **LoRA Training:** The final idea is to use a trained LoRA model to generate a consistent character consistently. This technique has the highest probability of success but is also by far the most time consuming. Training will require around 15 to 30 images of the character in different poses and situations. Additionally, all the images will have to be captioned with detailed textual descriptions, just like with the DreamBooth training.

A decision was also made to stick to a singular character for testing all strategies. This is because generating a standard character that maintains the same character design and visual traits will make the comparison and evaluation of the strategies far easier, since the entire point is to test for character consistency. Thus, the character's traits locked in going forward are the following:

- Male
- Young adult
- Black hair
- Yellow eyes
- Fair skin
- Black turtleneck
- Yellow jacket
- Dark jeans

The traits decided will, in-turn, determine the engineering of the prompts, especially since most anime models, including the two used in this project, are booru-tagged.

To make comparisons between strategies clearer and to make the judgement process fairer, it was decided that the same prompt and input images would be used across all strategies. The prompt would be of the male character standing on a building while overlooking a destroyed city.

General prompt for all strategies: *1boy, solo, standing, full body, outdoors, standing on building, destroyed city, fallen buildings, smoke, rubble, from behind, male focus*

The sketch to be used as input to img2img and ControlNet Scribble is the same one used in the DreamBooth evaluations and can be seen in figure 5.1 below. The input image for ControlNet Reference is shown in figure 5.5 which will be explored in the upcoming section next.

Fig. 5.1: Rough sketch of a male character in a destroyed city

In the next section, the creation of the character reference sheet for the male character with all the previously described traits is explored.

5.2. Character Reference Sheet

A character reference sheet is a single image of a particular character in multiple poses. These are very common in any kind of digital media production environment, be it a comic studio or an animation production studio or a video game development company. The sheet acts as a concept art for the character in question and is the main reference for artists drawing the character correctly from there on.

Fig. 5.2: Sam Bridges (Death Stranding) Character Sheet

Figure 5.2 and 5.3 show character reference sheets from two different video games. Often times, the poses for the character in the sheet may be dependent on the company or the artist drawing it, but around 3–4 poses are usually common. This is even more essential when using character sheets with AI, as AI models require scores of datapoints and examples.

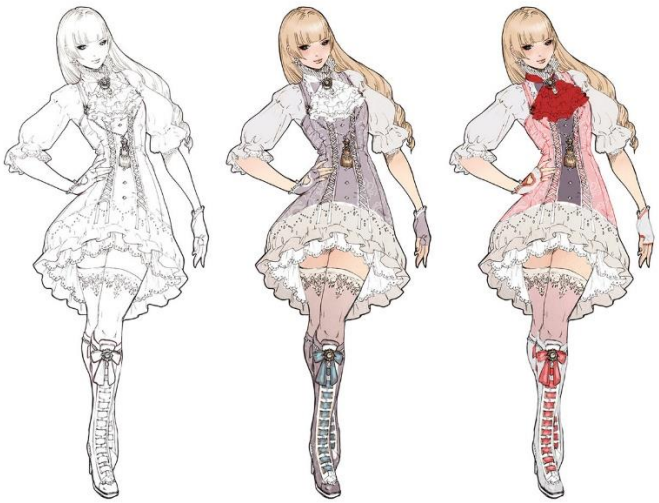


Fig. 5.3: Lili (Tekken 8) Character Sheet

A character reference sheet needed to be developed in a similar style for use in the ControlNet Reference and LoRA training strategies. The entirety of the ControlNet Reference strategy is dependent upon having a reference sheet to use, whereas LoRAs require multiple perspective views of the character it is being trained on, so reference sheets will naturally help there too.

To develop the sheet for the character described in the previous section, a great amount of prompt engineering needed to be done, with miniscule changes to descriptions of the character's jacket, colours, and overall traits. Multiple different models were used—more than the two anime models being focused on for these strategies—just to gain some insight into what type of prompts lead to better solutions. Moreover, a lot of artists will already have drawn character sheets for their original characters, so the model being used to generate the sheets is not of great importance. Some of the character sheets generated are depicted in figure 5.4.

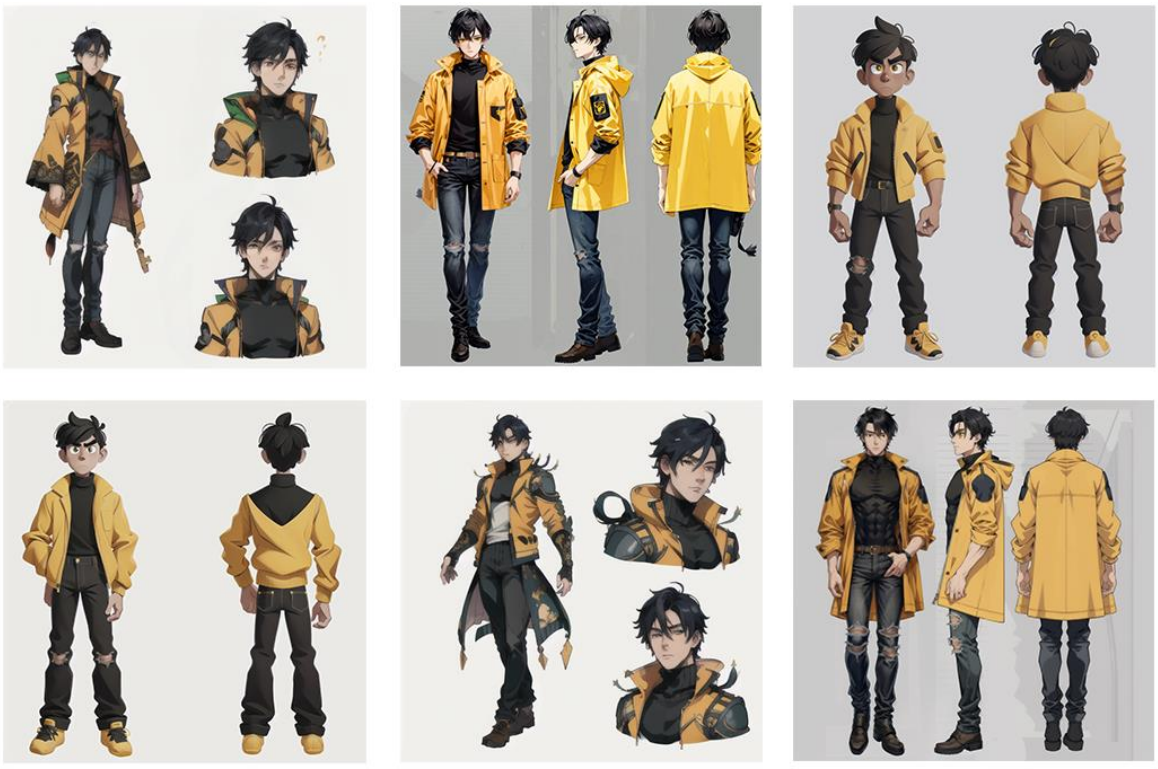


Fig. 5.4: Different Character Reference Sheets Generated

Out of the several sheets generated, the best-looking one was picked to go forward with, depicted in figure 5.5 below. With the reference sheet ready, the only remaining data to prepare was for the LoRA.

Fig. 5.5: Final Character Reference Sheet Chosen

5.3. LoRA Training Data Preparation

The training data required for the LoRA was very similar to the data required for DreamBooth. The main difference here was that a lot fewer images were needed to train characters, around 15-30, as compared to DreamBooth's 100 and above to train styles. However, as the character decided upon was an original character, obtaining images from online sources was not possible. The only solution was to draw all the images or generate them using AI. Thus, as 4 other strategies in addition to the LoRA to obtain consistent characters were already being researched, those same strategies were used to generate training data for the LoRA. LoRA training data needs to have characters with a very consistent appearance too, so the generations needed to be re-rolled dozens of times.

Moreover, as these images needed to be as varied as possible, they were different from the images being generated using the strategies for the project's evaluation, as the evaluation used the same prompt—and sometimes an input image—across all strategies to make the results easy to compare. For example, a few images of the boy in a destroyed city would be fine, but they needed to be balanced out by the boy in alternate scenarios. Ultimately, a number of varied situations were used, such as the boy on a beach, in a library, and in a bar. The character reference sheet developed was also used as training data.

33 images were generated in total, and next captions needed to be created to describe the content of the images using booru-tags, since the underlying checkpoint models being used—the CivitAI anime style models—were trained using booru-tags. When writing the captions, no tried-and-true method existed for describing the images well, but the consensus in the AI art community was that when training character LoRAs, everything *except* the character must be described. This is because when the LoRA learns what's in the images, it should regard the character as a single entity. Thus, describing the character's appearance and clothing would be counterproductive, as it may lead the model to infer that the character's clothing was variable and able to be changed. Finally, a trigger word must be given to the LoRA, which makes it easy to invoke when training as well as generating images. A rare token should preferably be used, as otherwise the model may associate the character with another concept it has previously trained on. The trigger word used here was "yjbl", short for "yellow jacket boy, Lei", where "Lei" was just a random name I referred to the boy as. Figure 5.6 shows examples of different images prepared and the captions written.

Fig. 5.6: LoRA Training Images with Captions

6. Character Consistency Strategies – Implementation and Evaluation

6.1. Introduction and Evaluation Success Criteria

This chapter details the work carried out for the generation of consistent characters. With all preparation complete and the strategies for achieving character consistency set, the project entered its final stage. However, before beginning the implementation, the success criteria were finalised. Each of the five strategies will have 40 images generated using every single strategy. From the 40 images, 4 will be made using the DreamBooth-trained panels model from chapter 4. Only 4 will be used because the results of this model are far inferior to the models downloaded from CivitAI. Moreover, the models trained in the preliminary study were primarily to choose whether to test the character consistency strategies on page generation or panel generation. Regarding the actual models themselves, the strategies should be compatible with any model as long as they generate singular images that can be used as panels. Thus, for the remaining 36 images for testing, 18 will be made using the AnimePastelDream model as the base, another 18 will be made using GalenaRedux. The images will then be evaluated against the 5 success criteria to measure their performance and compare each strategy against the others. To do this, the final scores of each strategy on all of the criteria will be counted and converted into a range from 0 to 5. For a more detailed explanation of the rating formula, see subsection 3.2.2 of chapter 3. The success criteria finalised are as follows:

- **Character Consistency:** Does the image display the traits of the character as previously finalised? (See section 5.1 of chapter 5 for the list of traits.) Can the character be easily recognised from these traits? Images that pass get a score of 1.

- **Presence or absence of Artifacts:** Does the image possess unsightly distortions, protrusions, or warped features such as clothing items fused to skin, extra/fewer hands, eyes, or mistakes of such nature? Images that possess artifacts get a score of 0, else 1.

- **Fine-grained Control:** Does the image generated follow the prompt or the input image (sketch or character reference sheet)? If most details present in the prompt are included in the output image, the image passes and gets a score of 1. If there are more

than a couple of details missing in the generation that were in the prompt, the score is 0.

- **Coherency:** Is it easy to understand the content of the image? If the prompt were not available, would a viewer find the image easy to parse? If yes, the score is 1, else 0.

- **Background Consistency:** Is the background consistent with the prompt descriptions or input sketch? As long as the background looks similar, the score is 1, else 0. For example, if a prompt describes a destroyed and ruined city with fallen buildings and smoke, but the image displays a pristine, beautiful city, the score is 0. This criteria can afford to be more lenient, since none of the strategies aim to maximise this.

Next, the project finally looks into implementing the different strategies and measuring their performance against the success criteria.

6.2. Strategy 1 – Prompt Engineering and Style Saving

Prompts, negative prompts, and the two aforementioned textual inversions in chapter 2 were used to generate images of a character with the traits mentioned in chapter 5. The prompts were iteratively changed and modified in small ways to gauge what differences they would result in. Ultimately, when the characters being generated were primarily of the type similar to the finalised traits, the style was saved.

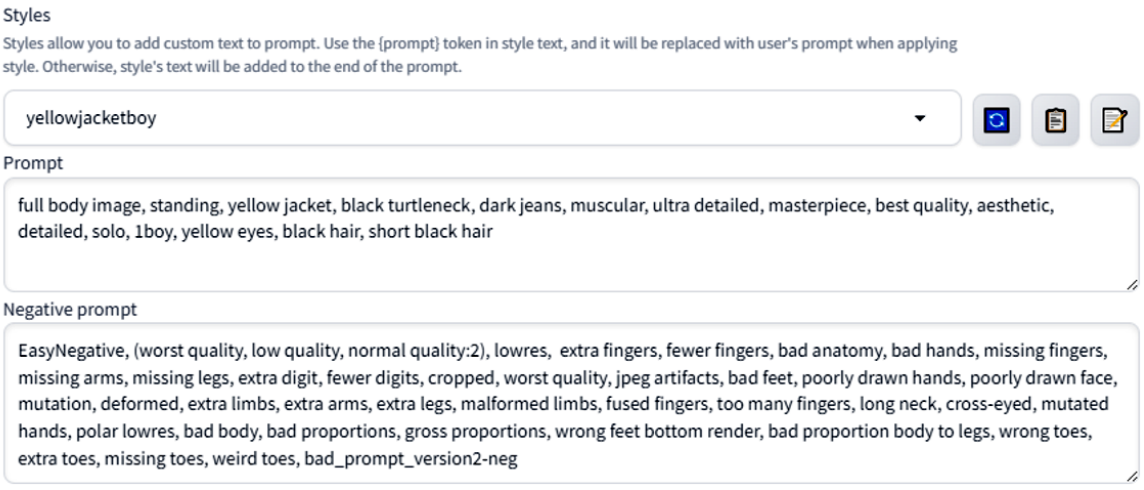


Fig. 6.1: Saved Style of the Character for Prompt Injection

Figure 6.1 depicts the details of the saved style and figure 6.2 depicts characters generated using that style. Now that the style was available to select and inject into different prompts, the

general prompt finalised in chapter 5 ("*1boy, solo, standing, full body, outdoors, standing on building, destroyed city, fallen buildings, smoke, rubble, from behind, male focus*") was used along with the injected style saved to evaluate the strategy. Some of the images generated are displayed in figure 6.3.

Fig. 6.2: Images of the character generated solely using the prompt of the saved style

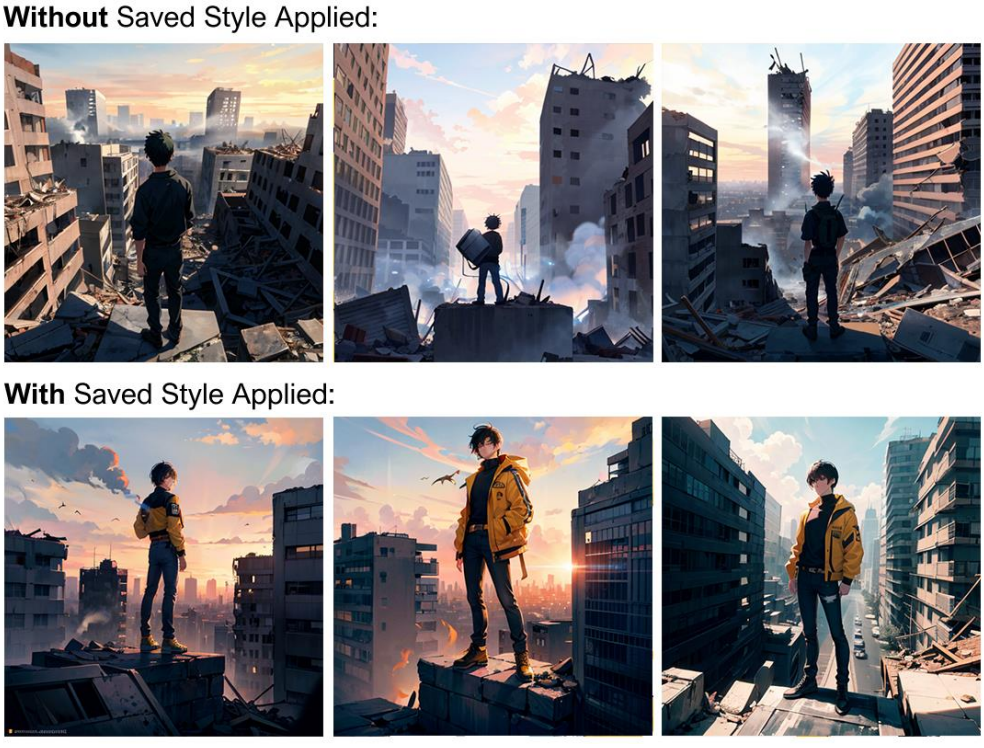


Fig. 6.3: Examples of images with and without the Saved Style applied

As seen, when the saved style is applied, the generated character becomes quite consistent. However, this comes at other costs. The city looks more destroyed before the style was applied, and applying the style seemed to have reduced the model's ability to generate rubble and destroyed buildings. This was the case across several of the images generated. Moreover, while the pose of the character was generally correct without the style—back facing the camera, with the style injected, the pose often came out wrong. The evaluated scores against the success criteria were calculated as follows:

- Character Consistency: 3.5/5 (28/40)
- Presence or absence of Artifacts 3.1/5 (25/40)
- Fine-grained Control: 2.8/5 (22/40)
- Coherency: 3.9/5 (31/40)
- Background Consistency: 2.6/5 (21/40)

6.3. Strategy 2 – Img2img

The rough sketch depicted in figure 5.1 along with the same previous general prompt were used as the inputs to generate images using img2img. Different denoising strength parameters were explored, and a value of 0.6 was found to work the best. Once all generation parameters seemed to be at their best, the 40 images for evaluation were then generated using img2img's sketch function, examples of which are seen in figure 6.4 below. A lot of the images unfortunately have a white, washed out look because the original sketch input was a drawing on a white background.

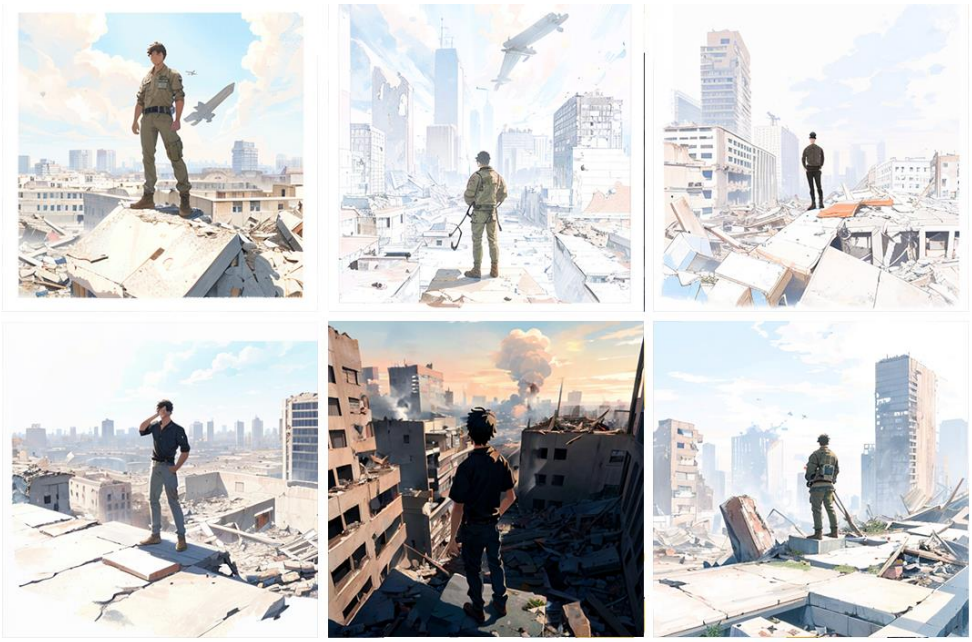


Fig. 6.4: Images generated using img2img's Sketch to Image function

Once the images from the sketch category were ready, the characters were then completely masked out and sent back into img2img, this time into the inpainting function. Next, inpainting required a description of how to change the characters masked out, where either a prompt or a saved style could be used. As the saved style for the boy in the yellow jacket was already available from the previous strategy's implementation, it was reused here. After the generation, some of the images were distorted, but most still turned out well, as seen in figure 6.6.

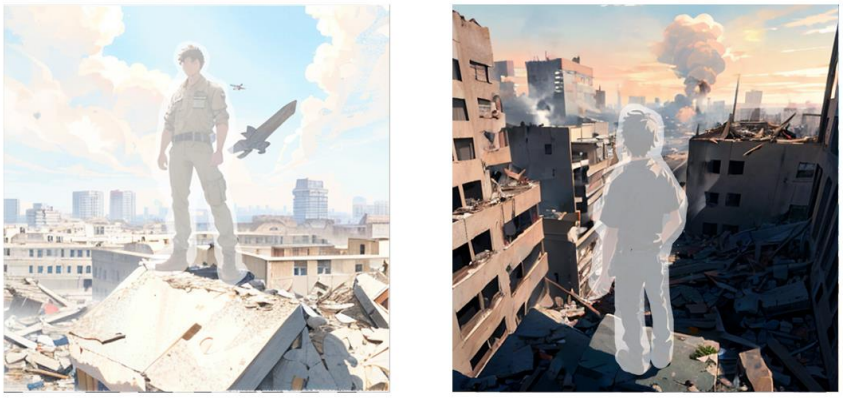


Fig. 6.5: img2img's prior outputs with the characters masked out for inpainting

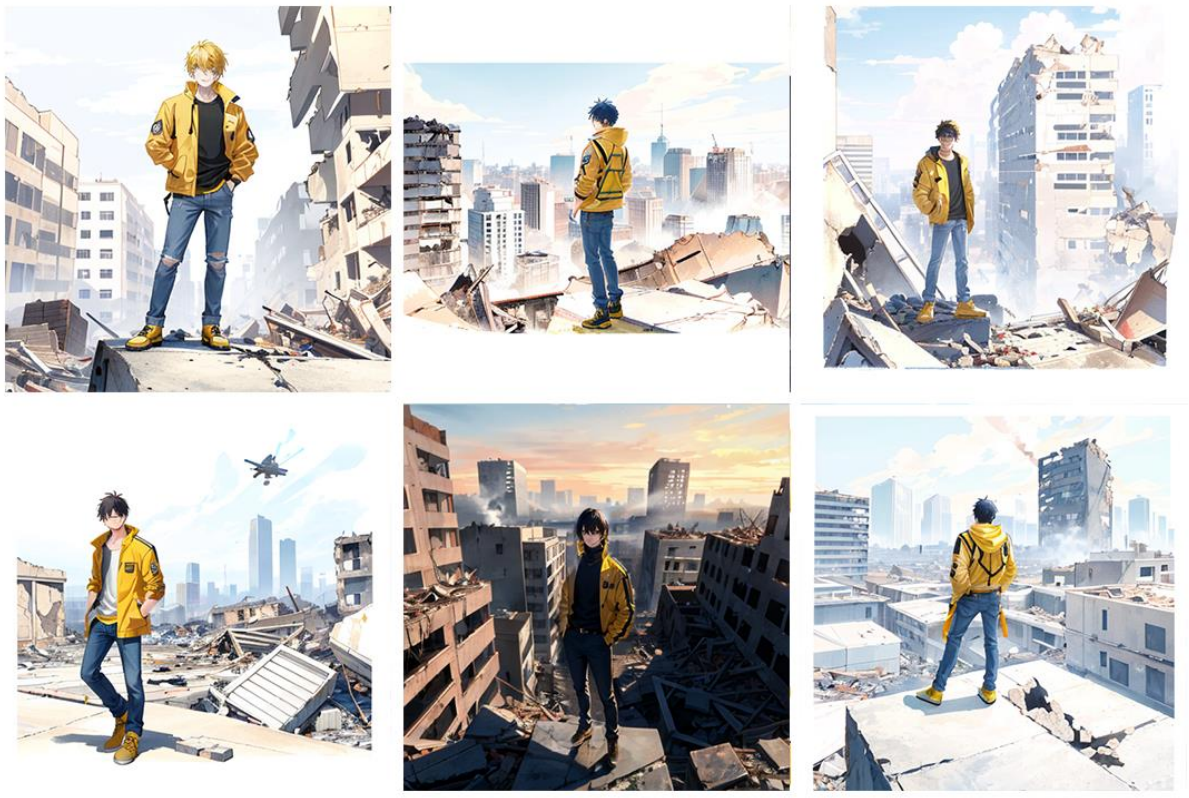


Fig. 6.6: Images from earlier now modified using Inpainting

From the results, it can be seen that img2img is reasonably decent at producing consistent characters. However, the poses from the original were altered a lot, and often the placement of the character in the scene was also changed. This is probably the result of the lower denoising strength value, which grants img2img more freedom to edit the original image as it wishes. However, a higher denoising strength value simply produces garbled and distorted images as seen in figure 6.7, so it was important for a balance to be struck.

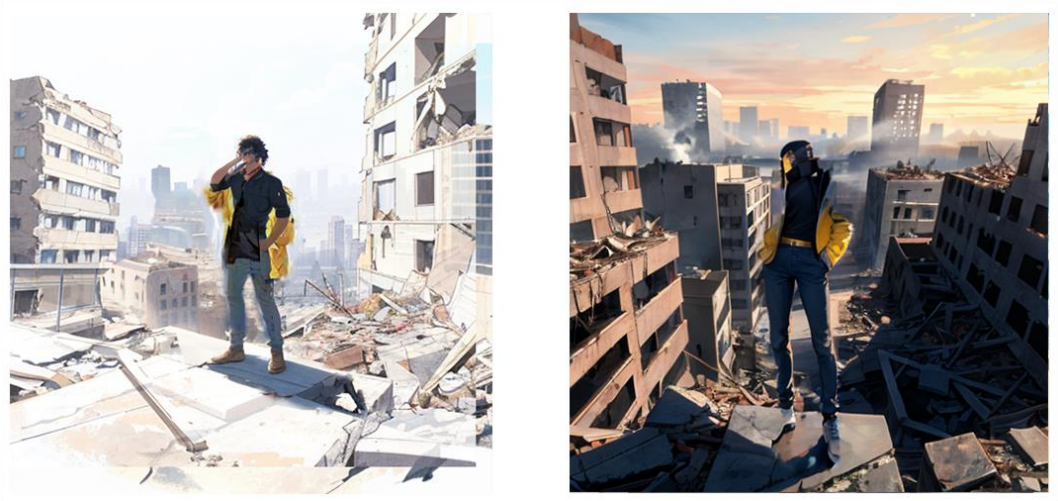


Fig. 6.7: Images generated by inpainting with a denoising strength of 8 or above

Final tallies against the success criteria:

- Character Consistency: 3.6/5 (29/40)
- Presence or absence of Artifacts: 3.8/5 (30/40)
- Fine-grained Control: 3.1/5 (25/40)
- Coherency: 3.9/5 (31/40)
- Background Consistency: 4.5/5 (36/40)

6.4. Strategy 3 – ControlNet Scribble

Using ControlNet Scribble was very similar to using img2img sketch, minus the whole second run with inpainting. The ControlNet extension was enabled, and the input sketch was provided along with the text prompt. The parameters of Pixel Perfect and the option to balance ControlNet's conditioning with the base model's output were selected as they returned better results. Other standard generation parameters were also tweaked until results were at their best. Finally, same as with img2img's inpainting run, the saved style from strategy 1 needed to be reused here. That is because ControlNet still needed a way to acquire data about the

character being generated. It was better at recognising characters within sketches and thus did not need a separate run like img2img and could convert the sketch and correctly produce the character at the same time. Figure 6.8 shows the results.

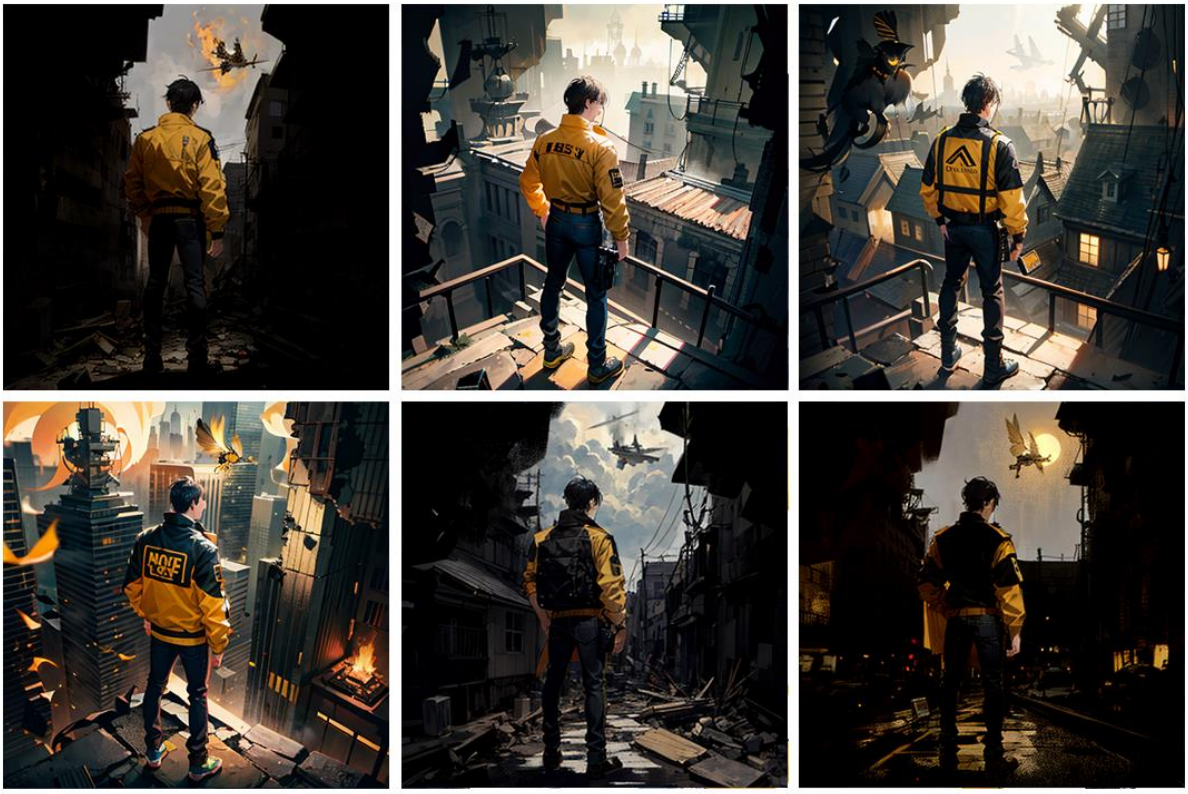


Fig. 6.8: Images generated by ControlNet Scribble

As seen, ControlNet Scribble was excellent in converting the sketch into a full-fledged picture with proper lighting. The white, washed-out look from img2img did not occur at all, and it nearly always got the pose, orientation, and placement of the character extremely accurately. However, some other problems did exist, such as the background being a hit-or-miss in terms of following the prompt to create a destroyed city, or the character's jacket not always being fully yellow. Another downside as compared to img2img is that it takes a lot of time to generate a single image, as ControlNet is an additional neural network on top of the base SD model. But this is a non-issue for powerful GPUs. Final tallies against the success criteria:

- Character Consistency: 4/5 (32/40)
- Presence or absence of Artifacts: 4.9/5 (39/40)
- Fine-grained Control: 3.8/5 (30/40)
- Coherency: 4.5/5 (36/40)
- Background Consistency: 2.9/5 (23/40)

6.5. Strategy 4 – ControlNet Reference

The character reference sheet—see figure 5.5—created during the design and data phase earlier now comes into play. ControlNet was enabled, and the reference function was selected. The finalised character sheet was used as the image input, as the exact same general prompt of the ruined city from earlier was fed in. The saved style from strategy 1 was not selected this time, as ControlNet was meant to do the work of keeping the character consistent. Pixel Perfect and the Balanced option were checked again. The other parameters were reused from ControlNet's Scribble run, as they provided the best results. The results are seen in figure 6.9.

Fig. 6.9: Images generated by ControlNet Reference

As seen, this technique has its pros and cons. On one hand, it is very decent at getting the character and the destroyed city background generated consistently. On the other hand, the pose and orientation of the character is wrong a lot of the time. Perhaps this technique could be combined with ControlNet Scribble, which nearly always gets the pose correct. It is, however, clear that ControlNet Reference is very good at extracting character appearance details from just a single reference image.

The tallied results are as follows:

- Character Consistency: 4.4/5 (35/40)

- Presence or absence of Artifacts: 3.9/5 (31/40)
- Fine-grained Control: 4.1/5 (33/40)
- Coherency: 4.2/5 (34/40)
- Background Consistency: 3.8/5 (30/40)

6.6. Strategy 5 – LoRA Training

The final and the most time-consuming strategy was the LoRA implementation. The data generation and captioning took a lot of time and effort, and once the 33 images and their captions were ready, the kohya_ss Web UI tool was run. There were a number of different training parameters available, but after some experimentation, the best results were observed with the following parameters:

- Repeats: 10
- Epochs: 10
- Learning Rate Warmup Ratio: 0.05
- Network Alpha: 8
- Convolution Alpha: 4
- Learning Rate Scheduler: cosine with restarts
- U-Net Learning Rate: 5e-4
- Text Encoder Learning Rate: 1e-4
- Network dimension: 16
- Convolution dimension: 8
- Learning Rate Scheduler Number: 3

The LoRA was trained on the AnimePastelDream checkpoint model as the base, since LoRAs cannot be standalone, and can only be trained or generate new images when using another model as the base. In the case of this project, AnimePastelDream and GalenaRedux were the two consistently best-performing models, so it was a clear choice to select one of them as the base. Finally, it is important to note that the selection of a certain model as a base when training does not hamper the LoRA from later generating images using another, different model as the new base.

The training had to be run iteratively 6 times in total before the final LoRA's results were satisfactory. The iterative improvement process included editing the captions of certain images as well as tuning and testing out different training parameters—before arriving at the ones listed above. Once the finalised LoRA was ready, the same general prompt from earlier was used to generate 40 images of the character in a destroyed city. Of course, a slightly modification was made to the prompt, which was the inclusion of this LoRA's trigger word, "yjbl" at the start of the prompt, as well as the text, "<lora:yellowjacketboylei:1>" which was necessary to invoke the usage of the LoRA in the first place. The ":1" at the end set the strength of the LoRA to full

strength. This was a number that could vary from 0 to 1 and a lower number meant the effects of the LoRA would not be as prominent. Figure 6.10 shows the generated results for the prompt provided.

Fig. 6.10: Images generated using the trained LoRA model

Character consistency was observed to be on the mark for most of the images using a LoRA. The orientation and pose of the character were faulty every now and then, and were not as perfect as ControlNet Scribble's results, however, that is to be expected as the LoRA did not use an additional sketch as input, which would definitely have had at least a slight improvement in its posing accuracy. The backgrounds were sometimes not consistent with the prompt, but this wasn't frequent enough to be a problem. Overall, the LoRA's results were quite good. The tallied score:

- Character Consistency: 4.8/5 (38/40)
- Presence or absence of Artifacts: 4.5/5 (36/40)
- Fine-grained Control: 4.2/5 (34/40)
- Coherency: 4.6/5 (37/40)
- Background Consistency: 3.9/5 (31/40)

6.7. Evaluating the Strategies

Given below is a metrics table, compiled from all the prior results of the evaluations of each strategy against each success criteria:

	Strategies					
Success Criteria		Style Saving	Img2Img	ControlNet Scribble	ControlNet Reference	LoRA
	Character Consistency	3.5	3.6	4.0	4.4	4.8
	Presence or absence of Artifacts	3.1	3.8	4.9	3.9	4.5
	Fine-grained Control	2.8	3.1	3.8	4.1	4.2
	Coherency	3.9	3.9	4.5	4.2	4.6
	Background Consistency	2.6	4.5	2.9	3.8	3.9

Table 6.1: Strategy Evaluation Metrics

When it comes to achieving character consistency, the LoRA's scores were the highest, which stands to reason, as it also had the most amount of training data provided, while the other strategies are far quicker to implement. However, if training data is available, then training a LoRA to achieve consistent characters will undoubtedly provide the best results. LoRAs are also helpful because multiple LoRAs can be used within a single prompt. Which means if an artist has multiple LoRAs, for all their different characters, every one of those characters can be generated consistently just by including their LoRAs within the prompt. Besides the LoRA technique, ControlNet's scores—both Reference as well as Scribble—are great too. Img2Img has the best scores when it comes to generating the desired background correctly, and no other strategy could quite reach its level. However, as this project aims to maximise character consistency, there is less of an emphasis on the background.

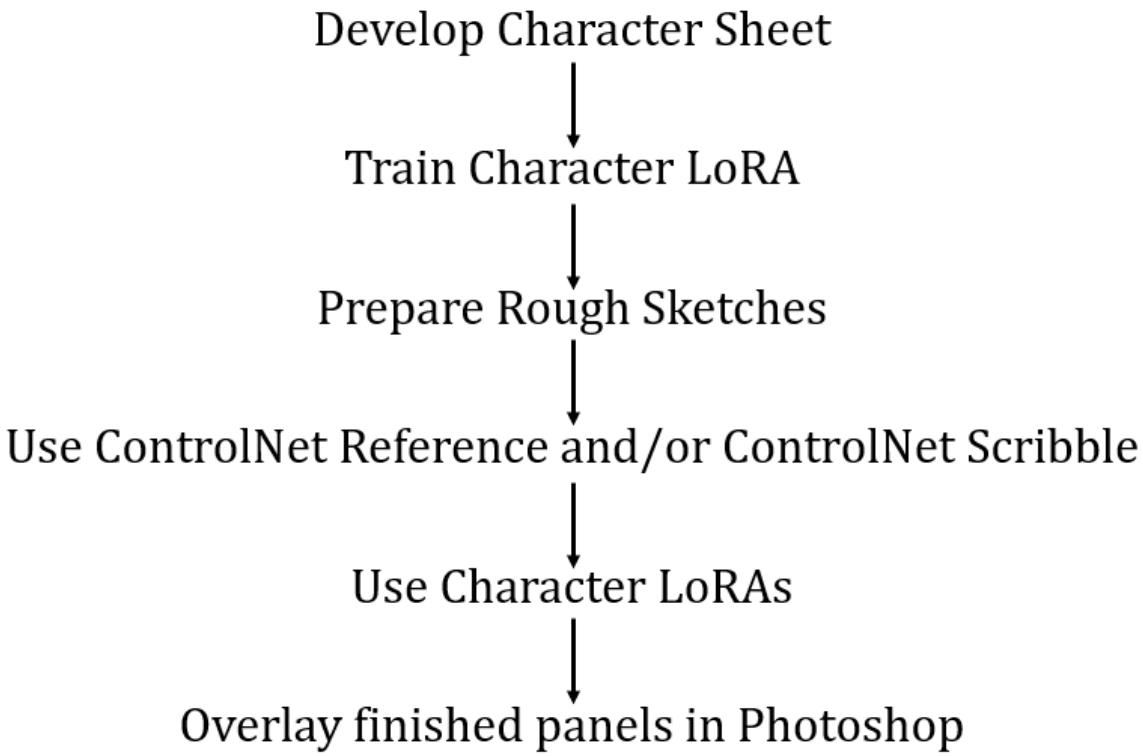
Even when considering the ControlNet strategies and the LoRA, there are trade-offs to be made when it comes to certain aspects. For instance, the character LoRAs may need a different strategies to balance out their weakness at backgrounds. Meanwhile, ControlNet Reference is

great for pulling data from the provided reference image but suffers the presence of artifacts slightly more than the LoRA and ControlNet Scribble do.

In the end, there is no strong reason to stick to just one strategy, so a combination of two or more of the implemented strategies will also work when incorporating these concepts into one's own artistic workflow. Thus, if an artist has a LoRA trained for a couple of characters but does not want to train a new one from scratch for a minor, one-off, non-recurring character, then using ControlNet's Scribble and Reference functions together might be a good idea. Especially since when used together, they can cover for each other's blind spots and balance out the images more.

6.8. Final Workflow Proposal

All points considered; the final workflow proposed is as follows:



To elaborate, the development of character sheets must always come first since they will form the backbone of the character design. They can also be used to train LoRAs. If using LoRAs, this will be a good time to train and test them out. Once they're ready, the artist is set to begin working on their comic and can prepare rough page layouts and sketches. The sketches can then be fed into ControlNet Scribble, all while making sure that at the same time they're also

using another strategy to maintain a consistent character design; either via ControlNet Reference along with the character sheet, or by using a trained character LoRA. Finally, all the generated images can be put together into a proper page in an image editing program like Photoshop.

Addendum: Earlier, it was concluded that the pages model would not be used since generative AI as a field is simply not at the state where the generation of multiple coherent images as one image returns good results. However, such models can still be used as inspiration for page layouts. That is, while the actual content of the panels within the generated page may be an incoherent mess, the layout of the panels themselves may be reusable, especially if an artist wants to ideate a number of possible layouts and desires inspiration to visualise something new.

7. Conclusion

7.1. Summary of Findings

The project explored a number of different strategies to find an effective workflow that artists can use to prototype comic books quickly, while being able to maintain consistent characters throughout their generated images. This is an especially hard task as diffusion models rely on an inherently random process to generate their images, meaning that generating consistent characters requires the addition of conditioning techniques to this random procedure, which is a decently tricky process. Initially, two checkpoint models were trained to test whether generating entire pages instead of individual panels would be more efficient. This was found to be an infeasible approach due to the current limitations of the field. The character consistency strategies were successfully implemented and evaluated within the project. They were demonstrated in this dissertation and tested rigorously against five different success criteria. The best performing strategies were found to be the LoRA training, if around 20 to 30 varied images of the target character are available, and ControlNet's Scribble and Reference functions, if working with limited data.

The final workflow proposed based on the findings was to use a combination of the suggested strategies. This entails beginning with a character reference sheet, training a LoRA, preparing some rough sketches for the panels, then using ControlNet Scribble to convert the sketches into finished images while using ControlNet Reference or a character LoRA to maintain character consistency. The final step is to assemble all the generated images into a comic page via any image-editing software.

Overall, the proposed workflow can help artists prototype comics with consistent characters far quicker than drawing them by hand. The workflow offers two main advantages over the alternate approaches to generating comics as discussed in chapter 2. Namely, the models used here are open source, granting the artist complete freedom and control over the entire process and art style. They also have the ability to adapt the workflow to their original characters, since this approach is not tied to any specific pre-existing popular characters. The artists are free to choose or train any model that suits their art style and can run all the software needed from their own devices, provided they possess a reasonably powerful GPU. If not, cloud computing services like Google Colab can be used, and while this means the artists won't be running the

tools on their own device, they still maintain complete control over the process as the tools are all open source.

7.2. Limitations

While valuable insights were obtained and the end goal of the project was a success, several limitations exist that offer opportunities for future exploration. The field of diffusion models is a rapidly evolving landscape. The field has continued moving forward since the beginning when the decision was made for this project to use Stable Diffusion 1.5, and newer models like SD 2.1, SDXL, and Stable Cascade have been released, which offer huge improvements in capability and performance. Applying the strategies evaluated in this project to those models could yield better results.

The scope of the project was also somewhat limited, only focusing on character consistency, whereas when drawing comics, the backgrounds are equally as important. Focusing on both, characters and backgrounds would be out of the scope of a project lasting only a few months, so this is definitely an area that needs more detailed research. Moreover, other crucial areas, such as the presence of artifacts or making sure the model sticks to the details requested in the prompts were evaluated, but no strategies were implemented to actually maximise their occurrences or absence. Thus, expanding the scope of the project would provide a more comprehensive understanding of how to maintain consistency not just across different characters, but across entire images.

Finally, the greatest limiting element was the lack of participation from real artists. Given more time, the project could have explored how real people producing art would fare when using the insights gleaned through this project, and whether they would identify certain inconsistencies or problems with the project. The incorporation of this workflow into the daily work of most comic artists would have no doubt provided great insights into how to best improve the workflow and potentially highlight crucial aspects of the research that might have been neglected.

7.3. Future work

Comic generation with AI has been explored by this project, as well other a few others as noted in the related work section in chapter 2. However, no one has yet to build a fully deployed

application that artists can access and use on a daily basis as part of their workflow. A future project could look into building such an app using the insights already obtained by the likes of this project, or perhaps act as a more comprehensive project, and examine comic generation and consistency across images in greater detail while also implementing a deployable application.

As generative AI is currently a bleeding-edge, rapidly progressing field, multitudes of advances are being made every few months. However, there are still problems that need addressing and there is a lot of work to yet be done. The biggest example is to address the presence of artifacts, distortions, and illogical things in an image that a human would never draw. This is due to the lack of a world model, that is, the AIs do not really understand the world and how everything works, but rather do their best to approximate the training data they were given.

The creation of an internal world model would bring a magnitude more believability to AI art, and even start to touch on some other limitations, such as the existence of distorted text within images, which is no doubt of great importance when it comes to generating comics and entertainment media. Most diffusion models are absolutely terrible with generating text within images. When tried, there is a near definite certainty that any generated text will be a twisted, garbled mess. Newer models like Dall-E3 with GPT-4Vision by OpenAI and Stable Cascade by Stable Diffusion are making great strides in addressing this problem (Betker, et al., 2023) (StabilityAI, 2024), and a lot of untapped potential exists. A longer context window would also work wonders when it comes to the generation of entire pages of comics, since the AI will be needing to generate multiple sub-images as well as text as part of a page.

Ultimately, comics are just a form of entertainment media, and generative AI has the potential to upend the entire entertainment industry and redefine what it means to be an artist. For instance, OpenAI's Sora model is able to create 1-minute-long photorealistic videos, while Suno v3 looks poised to start generating humanlike music with lyrics (Brooks, et al., 2024) (SunoAI, 2024). Such generative AI are for now completely prompted with text descriptions of things in natural language, such as plain English. Such models, in-field, are referred to by terms such as Text-to-Image, Text-to-Video, and Image-to-Image—such as Stable Diffusion's functionality, as explored in this project—etc. In the medium to long-term future, once adequately high-resolution Brain-Computer Interfaces are available, the greatest work of relevance could very well become Brain-to-Image, Brain-to-Video, and Brain-to-Audio, essentially transforming the thoughts and imaginations from one's mind into an endless source of digital media (Urban, 2017). General people lacking traditional artistic skill might then be able to tangibly visualise their mind's eye, setting free the inner artist within all humanity.

References

1. An, J., 2023. *Trick Methods to Create Consistent Human Characters in Stable Diffusion.* [Online]
   Available at: https://jerryan.medium.com/trick-methods-to-create-consistent-human-characters-in-stable-diffusion-92907dce2cf2
   [Accessed 08 11 2023].

2. atinylittleshell, 2023. *PreSize.io: Bulk Preprocess, Resize and Crop Your Images.* [Online]
   Available at: https://github.com/atinylittleshell/presize
   [Accessed 26 12 2023].

3. Automatic1111, 2022. *StableDiffusion Web UI.* [Online]
   Available at: https://github.com/AUTOMATIC1111/stable-diffusion-webui
   [Accessed 13 10 2023].

4. Banathy, B. H., 1996. *Designing Social Systems in a Changing World.* s.l.:Springer US (https://link.springer.com/book/10.1007/978-1-4757-9981-1).

5. Betker, J. et al., 2023. Improving Image Generation with Better Captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf 2, no. 3,* p. 8.

6. Boutin, V. et al., 2023. Diffusion Models as Artists: Are we closing the gap between Humans and Machines?. *arXiv preprint arXiv:2301.11722.*

7. British Design Council, 2005. *The Double Diamond: A universally accepted depiction of the design process.* [Online]
   Available at: https://www.designcouncil.org.uk/our-resources/the-double-diamond/
   [Accessed 19 10 2023].

8. Brooks, T. et al., 2024. *Creating Video from Text.* [Online]
   Available at: https://openai.com/sora
   [Accessed 20 02 2024].

9. camenduru, 2023. *stable-diffusion-webui-colab.* [Online]
   Available at: https://github.com/camenduru/stable-diffusion-webui-colab
   [Accessed 23 12 2023].

10. catboxanon, 2023. *stable-diffusion-webui - Online Services.* [Online]
    Available at: https://github.com/AUTOMATIC1111/stable-diffusion-webui/wiki/Online-Services#google-colab
    [Accessed 23 12 2023].

11. CivitAI, 2022. *CivitAI.* [Online]
    Available at: https://civitai.com/
    [Accessed 29 11 2023].

12. Computerphile, 2022. *How AI Image Generators Work (Stable Diffusion / Dall-E) - Computerphile.* [Online]
    Available at: https://youtu.be/1CIpzeNxIhU?si=S9uJ16YaaPqvgRI2
    [Accessed 18 10 2023].

13. Dalton, J., 2018. *Great Big Agile: An OS for Agile Leaders.* s.l.:Apress.

14. Dhariwal, P. & Nichol, A., 2021. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems 34,* pp. 8780-8794.

15. Gal, R. et al., 2022. An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion. *arXiv preprint arXiv:2208.01618.*

16. GitHub, 2023. *About GitHub.* [Online]
Available at: https://github.com/about
[Accessed 15 01 2024].

17. Goodfellow, I. J. et al., 2014. Generative Adversarial Networks. *Advances in Neural Information Processing Systems 27.*

18. Google, 2017. *Google Colaboratory.* [Online]
Available at: https://colab.google/
[Accessed 12 01 2024].

19. Ha, A. Y. J. et al., 2024. Organic or Diffused: Can We Distinguish Human Art from AI-generated Images?. *arXiv preprint arXiv:2402.03214.*

20. Havoc, 2023. *EasyNegative.* [Online]
Available at: https://civitai.com/models/7808/easynegative
[Accessed 18 01 2024].

21. Hoffman, J. et al., 2022. Training Compute-Optimal Large Language Models. *arXiv preprint arXiv:2203.15556.*

22. hollowstrawberry, 2023. *Lora Trainer by Hollowstrawberry.* [Online]
Available at: https://colab.research.google.com/github/hollowstrawberry/kohya-colab/blob/main/Lora_Trainer.ipynb
[Accessed 05 02 2024].

23. Hu, E. J. et al., 2021. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685.*

24. Hugging Face, 2024. *Hugging Face.* [Online]
Available at: https://huggingface.co/huggingface
[Accessed 03 01 2024].

25. Illyasviel, 2023. *lllyasviel/ControlNet-v1-1.* [Online]
Available at: https://huggingface.co/lllyasviel/ControlNet-v1-1/tree/main
[Accessed 27 12 2023].

26. Intellectual Property Office, 2021. *Exceptions to Copyright.* [Online]
Available at: https://www.gov.uk/guidance/exceptions-to-copyright#non-commercial-research-and-private-study
[Accessed 14 11 2023].

27. Jin, Z. & Song, Z., 2023. Generating coherent comic with rich story using ChatGPT and Stable Diffusion. *arXiv preprint arXiv:2305.11067.*

28. Johnson, H. A., 2017. Trello. *Journal of the Medical Library Association: JMLA, 105(2),* p. 209.

29. Jonas, O., 2022. The Creativity of Text-to-Image Generation. *Proceedings of the 25th International Academic Mindtrek Conference,* pp. 192-202.

30. Kaplan, J. et al., 2020. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361.*

31. Karras, T. et al., 2020. Analyzing and Improving the Image Quality of StyleGAN. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition,* pp. 8110-8119.

32. Kirkman, R., Ottley, R. & Rauch, J., 2023. *Invincible.* [Online]
Available at: https://imagecomics.com/comics/series/invincible
[Accessed 2023 12 27].

33. kohya_ss, 2022. *sd-scripts.* [Online]
Available at: https://github.com/kohya-ss/sd-scripts
[Accessed 05 02 2024].

34. Liu, V. & Chilton, L. B., 2022. Design Guidelines for Prompt Engineering Text-to-Image Generative Models. *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems,* pp. 1-23.

35. lllyasviel, 2023. *[Major Update] Reference-only Control.* [Online]
Available at: https://github.com/Mikubill/sd-webui-controlnet/discussions/1236
[Accessed 26 01 2024].

36. Lykon, 2023. *Anime Pastel Dream.* [Online]
Available at: https://civitai.com/models/23521/anime-pastel-dream
[Accessed 03 01 2024].

37. Maltais, B., 2022. *Kohya's GUI.* [Online]
Available at: https://github.com/bmaltais/kohya_ss?tab=readme-ov-file#kohyas-gui
[Accessed 05 02 2024].

38. Meccai, Z., 2023. *How to use Stable Diffusion to Create Comic Strips.* [Online]
Available at: https://blog.segmind.com/how-to-use-stable-diffusion-to-create-comic-strips/
[Accessed 21 10 2023].

39. Microsoft, 2023. *Visual Studio Code: Code Editing. Redefined.* [Online]
Available at: https://code.visualstudio.com/
[Accessed 02 01 2024].

40. Midjourney, 2022. *Midjourney.* [Online]
Available at: https://www.midjourney.com/home
[Accessed 17 10 2023].

41. Mobo, S., 2017. *danbooru-tags.* [Online]
Available at: https://github.com/stmobo/Machine-Learning/blob/master/danbooru-tags.csv
[Accessed 01 02 2024].

42. Monzon Media, 2023. *The Truth About Consistent Characters In Stable Diffusion.* [Online]

Available at: https://www.youtube.com/watch?v=WQsUeyzq-VA
[Accessed 24 10 2023].

43. Nash, J., 1951. Non-Cooperative Games. *The Annals of Mathematics, Second Series, Volume 54, Issue 2,* pp. 286-295.

44. Nerfgun3, 2023. *bad_prompt Negative Embedding.* [Online]
Available at: https://civitai.com/models/55700/badprompt-negative-embedding
[Accessed 18 01 2024].

45. Nichol, A. et al., 2021. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. *arXiv preprint arXiv:2112.10741.*

46. Packer, D., 2023. *Character Consistency in Stable Diffusion (Part 1).* [Online]
Available at: https://cobaltexplorer.com/2023/06/character-sheets-for-stable-diffusion/
[Accessed 16 10 2023].

47. Pan, X. et al., 2024. Synthesizing coherent story with auto-regressive latent diffusion models. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision,* pp. 2920-2930.

48. Penna, J. & Bielejeski, D., 2023. *DreamBooth Google Colab JoePenna.* [Online]
Available at: https://colab.research.google.com/github/JoePenna/Dreambooth-Stable-Diffusion/blob/main/dreambooth_google_colab_joepenna.ipynb
[Accessed 30 12 2023].

49. Podell, D. et al., 2023. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis. *arXiv preprint arXiv:2307.01952.*

50. Radford, A. et al., 2021. Learning Transferable Visual Models From Natural Language Supervision. *International conference on machine learning, PLMR,* pp. 8748-8763.

51. Ramachandran, K. K. & Karthick, K. K., 2019. Gantt Chart: An Important Tool of Management. *International Journal of Innovative Technology and Exploring Engineering,* 8(7).

52. Ramesh, A. et al., 2022. Hierarchical Text-Conditional Image Generation with CLIP Latents. *arXiv preprint arXiv:2204.06125, 1.2, 3.*

53. Rombach, R. et al., 2022a. High-Resolution Image Synthesis with Latent Diffusion Models. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition,* pp. 10684-10695.

54. Rombach, R., Blattmann, A. & Ommer, B., 2022b. Text-Guided Synthesis of Artistic Images with Retrieval-Augmented Diffusion Models. *arXiv preprint arXiv:2207.13038.*

55. Ruiz, N. et al., 2023. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pp. 22500-22510.

56. Saharia, C. et al., 2022. Palette: Image-to-Image Diffusion Models. *ACM SIGGRAPH 2022 Conference Proceedings,* pp. 1-10.

57. Saharia, C. et al., 2022. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. *Advances in Neural Information Processing Systems 35,* pp. 36479-36494.

58. Saxena, D. & Cao, J., 2021. Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions. *ACM Computing Surveys (CSUR), 54(3),* pp. 1-42.

59. Shrirao, S., 2023. *DreamBooth_Stable_Diffusion.* [Online]
Available at:
https://colab.research.google.com/github/ShivamShrirao/diffusers/blob/main/examples/dreambooth/DreamBooth_Stable_Diffusion.ipynb
[Accessed 30 12 2023].

60. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N. & Ganguli, S., 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. *International conference on machine learning, PLMR,* pp. 2256-2265.

61. StabilityAI, 2019. *StabilityAI.* [Online]
Available at: https://stability.ai/about
[Accessed 02 11 2023].

62. StabilityAI, 2022a. *runwayml/stable-diffusion-v1-5.* [Online]
Available at: https://huggingface.co/runwayml/stable-diffusion-v1-5
[Accessed 22 10 2023].

63. StabilityAI, 2022b. *Stable Diffusion v1 Model Card.* [Online]
Available at: https://github.com/CompVis/stable-diffusion/blob/main/Stable_Diffusion_v1_Model_Card.md#training
[Accessed 17 02 2024].

64. StabilityAI, 2024. *Introducing Stable Cascade.* [Online]
Available at: https://stability.ai/news/introducing-stable-cascade
[Accessed 15 02 2024].

65. SunoAI, 2024. *Suno.* [Online]
Available at: https://www.suno.ai/#about
[Accessed 20 02 2024].

66. TheLastBen, 2023a. *fast_stable_diffusion_AUTOMATIC1111.* [Online]
Available at: https://colab.research.google.com/github/TheLastBen/fast-stable-diffusion/blob/main/fast_stable_diffusion_AUTOMATIC1111.ipynb
[Accessed 23 12 2023].

67. TheLastBen, 2023b. *fast-DreamBooth.* [Online]
Available at: https://colab.research.google.com/github/TheLastBen/fast-stable-diffusion/blob/main/fast-DreamBooth.ipynb
[Accessed 30 12 2023].

68. u/dicklim39, 2023. *(reddit) Need Help - Why SD not follow my prompts ?.* [Online]
Available at:
https://www.reddit.com/r/StableDiffusion/comments/141w0h3/need_help_why_sd_not_follow_my_prompts/
[Accessed 14 11 2023].

69. u/GrodanHej, 2024. *(reddit) How do I get AI to follow the prompt? Even when it writes the prompt itself it ignores it..* [Online]
Available at:
https://www.reddit.com/r/aiArt/comments/198ja8w/how_do_i_get_ai_to_follow_the_prompt_even_when_it/
[Accessed 20 01 2024].

70. u/MysteryInc152, 2022. *Introducing Comic-Diffusion. Consistent 2D styles in general are almost non-existent on Stable Diffusion so i fine-tuned a model for the typical Western Comic Book Style Art..* [Online]
Available at:
https://www.reddit.com/r/StableDiffusion/comments/yfsdx0/introducing_comicdiffusion_consistent_2d_styles/
[Accessed 18 10 2023].

71. u/not-me-374892, 2022. *(reddit) How to get IMG2IMG to more closely follow prompt?.* [Online]
Available at:
https://www.reddit.com/r/StableDiffusion/comments/xun6dp/how_to_get_img2img_to_more_closely_follow_prompt/
[Accessed 10 11 2023].

72. u/QinnShou, 2023. *A simple comparison of Easy Negative and Bad Prompt v2; r/StableDiffusion.* [Online]
Available at:
https://www.reddit.com/r/StableDiffusion/comments/12z10h2/a_simple_comparison_of_easy_negative_and_bad/
[Accessed 18 01 2024].

73. u/scifivision, 2023. *How has AI art made you a better artist?.* [Online]
Available at:
https://www.reddit.com/r/StableDiffusion/comments/1054w7k/how_has_ai_art_made_you_a_better_artist/
[Accessed 22 10 2023].

74. u/wonderflex, 2022. *(reddit) Tutorial: Creating characters and scenes with prompt building blocks.* [Online]
Available at:
https://www.reddit.com/r/StableDiffusion/comments/z7pbjn/tutorial_creating_characters_and_scenes_with/
[Accessed 14 11 2023].

75. Urban, T., 2017. *Neuralink and the Brain's Magical Future.* [Online]
Available at: https://waitbutwhy.com/2017/04/neuralink.html
[Accessed 21 02 2024].

76. Various Reddit users, 2023. *(reddit) Search results - "artifact".* [Online]
Available at:
https://www.reddit.com/r/StableDiffusion/search/?q=%22artifact%22&restrict_sr=1
[Accessed 19 12 2023].

77. Victor Gnarly, 2023. *MovieMachine AI Filmmaking Guide.* [Online]
Available at:
https://docs.google.com/document/d/1xkPPu_dD2O2gkZgV71Omr73gduCo1lDVakoN5cDRAWU/edit?usp=sharing
[Accessed 15 10 2023].

78. Wang, J. et al., 2023. Review of large vision models and visual prompt engineering. *Meta-Radiology,* p. 100047.

79. Wang, P., 2019. *This Person Does Not Exist.* [Online]
Available at: https://thispersondoesnotexist.com/
[Accessed 17 10 2023].

80. Yang, L. et al., 2023. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys 56, no. 4,* pp. 1-39.

81. Yin, Y., Huang, L. & Huang, K., 2022. DiffGAR: Model-Agnostic Restoration from Generative Artifacts Using Image-to-Image Diffusion Models. *Proceedings of the 2022 6th International Conference on Computer Science and Artificial Intelligence,* pp. 55-62.

82. Zhang, L., Rao, A. & Agrawala, M., 2023. Adding conditional control to text-to-image diffusion models. *Proceedings of the IEEE/CVF International Conference on Computer Vision,* pp. 3836-3847.

A. Appendix – Personal Reflection

A.1. Reflection on Project

Looking back on things, I can identify a few things that could have been done differently. For starters, I meandered a lot when selecting a topic for the project, as I initially wanted to work with a brain-computer interface and do a project related to that in some way. However, after experimenting with the devices I could obtain, I realised that all of my ideas were too ambitious and would most probably not work out as the spatial and temporal resolution of these devices is very poor. After realising this would not pan out, I spent even more time researching possible avenues for the project, wanting to work with the latest bleeding-edge AI technology. However, this led me down multiple rabbit holes as the field of generative AI was progressing so quickly and there was so much to learn about. In the end, I went with comic book generation due to my hobby of digital illustration and reading comics. I cannot help but feel if I had converged upon an idea sooner, I would've had more time to research things actually pertinent to the project and might have ended up with a superior project overall.

The project was also too ambitious to begin with, and I initially believed I could not only identify and evaluate techniques for consistent characters, but also implement the techniques into a fully deployable app. I then aimed to host the app online and have a user evaluation with real artists using my app and evaluating my findings. This is why the BREO application for this project indicates that a user evaluation would be carried out. However, the scope of the research was already too demanding, and I did not get the time to even start building an app, ultimately needing to rely solely on the success criteria-based evaluation. Additionally, in hindsight, the entire page generation approach not working out was a foregone conclusion, since generating multiple images as part of a single one was just too complex for these models. Thus, instead of spending ages on collecting, processing, and captioning 200 images in detail, it may have been better to skip the DreamBooth fine-tuning process entirely and instead have just built the deployable app. After all, the models trained were just for research, the workflow itself was always meant to be generalisable, regardless of the model being used.

Regarding the character consistency strategies, I believe there was a lot more room for mixing and matching different strategies to bring about better results. For instance, ControlNet Scribble and ControlNet Reference were looked into, and both had very decent results on their own. However, a combined strategy using both Scribble and Reference at the same time should

also have been carried out, just as a combined strategy using img2img's Sketch and Inpainting functions were carried out. Similarly, the style saving approach could have been applied with ControlNet or LoRAs to test for more combined strategies.

Finally, with more time, two or more-character LoRAs could have been trained, allowing me to test multiple character generations at once. In theory, this should definitely work, and working examples of it are not uncommon in AI art spheres online, which is why this isn't listed in the limitations section in the conclusion. But even if it works in theory, actually testing it out may have proven valuable.

A.2. Personal Reflection

There were a few moments where I felt inundated by all the work to be done, not just for the FYP, but also for the other modules and especially for job applications. I studied coding challenges and did interview preparation, none of which I ultimately ended up using as I got a return offer from my company where I did my industrial placement. In hindsight, I could have used this time more wisely instead of procrastinating on the FYP and working on other things instead.

Conversely, one aspect I think I handled well was managing my workload once I did get past any lingering procrastination impulses. Had I started to do this earlier, my time may have been managed even more efficiently. I regularly used the Trello board mentioned in the dissertation to keep track of my work and manage my work per week well. The success of this approach is definitely in part thanks to the Agile/Kanban boards that my team in my company used at work during my placement. All of our tasks as a team were thoroughly tracked using JIRA, using very similar "To do", "In Progress", "Blocked", and "Done" columns, and due to this, I was in the habit of regularly checking my progress against the board and putting my completed and pending work back into perspective with the big picture. This helps a lot against hyper-focusing on the present details and consequently spending too much time on a single task and related tangents, which was definitely a problem for me during year 2.

To conclude, while I was successful in attaining all the objectives and aims laid out for this project, I believe there are a number of ways in which I could have spent my time more wisely. There are also things I did well, such as managing my workload which gives me insight into what I do well, thus poising myself to better leverage my strengths in the future.

B. Appendix – Ethics Approval

The following is the letter of approval granted to me by the Brunel Research Ethics Committee. This letter is for a user-based evaluation, however as mentioned, conducting a user evaluation ended up being out of the scope of the project. Thus, a personal evaluation against success criteria was carried out, which would be a lower risk, “No Ethics Required” project. As I was granted permission for a higher-risk project, I did not reapply for a new letter.

College of Engineering, Design and Physical Sciences Research Ethics Committee
Brunel University London
Kingston Lane
Uxbridge
UB8 3PH
United Kingdom

www.brunel.ac.uk

28 November 2023

LETTER OF APPROVAL

APPROVAL HAS BEEN GRANTED FOR THIS STUDY TO BE CARRIED OUT BETWEEN 15/01/2024 AND 09/02/2024

Applicant (s): Mr Daniel Rodrigues

Project Title: An effective workflow for the conversion of text prompts and sketches of pages into rendered comic book pages

Reference: 46143-LR-Nov/2023- 48105-1

Dear Mr Daniel Rodrigues

The Research Ethics Committee has considered the above application recently submitted by you.

The Chair, acting under delegated authority has agreed that there is no objection on ethical grounds to the proposed study. Approval is given on the understanding that the conditions of approval set out below are followed:

- **The agreed protocol must be followed. Any changes to the protocol will require prior approval from the Committee by way of an application for an amendment.**
- **Please ensure that you monitor and adhere to all up-to-date local and national Government health advice for the duration of your project.**

Please note that:

- Research Participant Information Sheets and (where relevant) flyers, posters, and consent forms should include a clear statement that research ethics approval has been obtained from the relevant Research Ethics Committee.
- The Research Participant Information Sheets should include a clear statement that queries should be directed, in the first instance, to the Supervisor (where relevant), or the researcher. Complaints, on the other hand, should be directed, in the first instance, to the Chair of the relevant Research Ethics Committee.
- Approval to proceed with the study is granted subject to any conditions that may appear above.
- The Research Ethics Committee reserves the right to sample and review documentation, including raw data, relevant to the study.
- If your project has been approved to run for a duration longer than 12 months, you will be required to submit an annual progress report to the Research Ethics Committee. You will be contacted about submission of this report before it becomes due.
- You may not undertake any research activity if you are not a registered student of Brunel University or if you cease to become registered, including abeyance or temporary withdrawal. As a deregistered student you would not be insured to undertake research activity. Research activity includes the recruitment of participants, undertaking consent procedures and collection of data. Breach of this requirement constitutes research misconduct and is a disciplinary offence.

Professor Simon Taylor

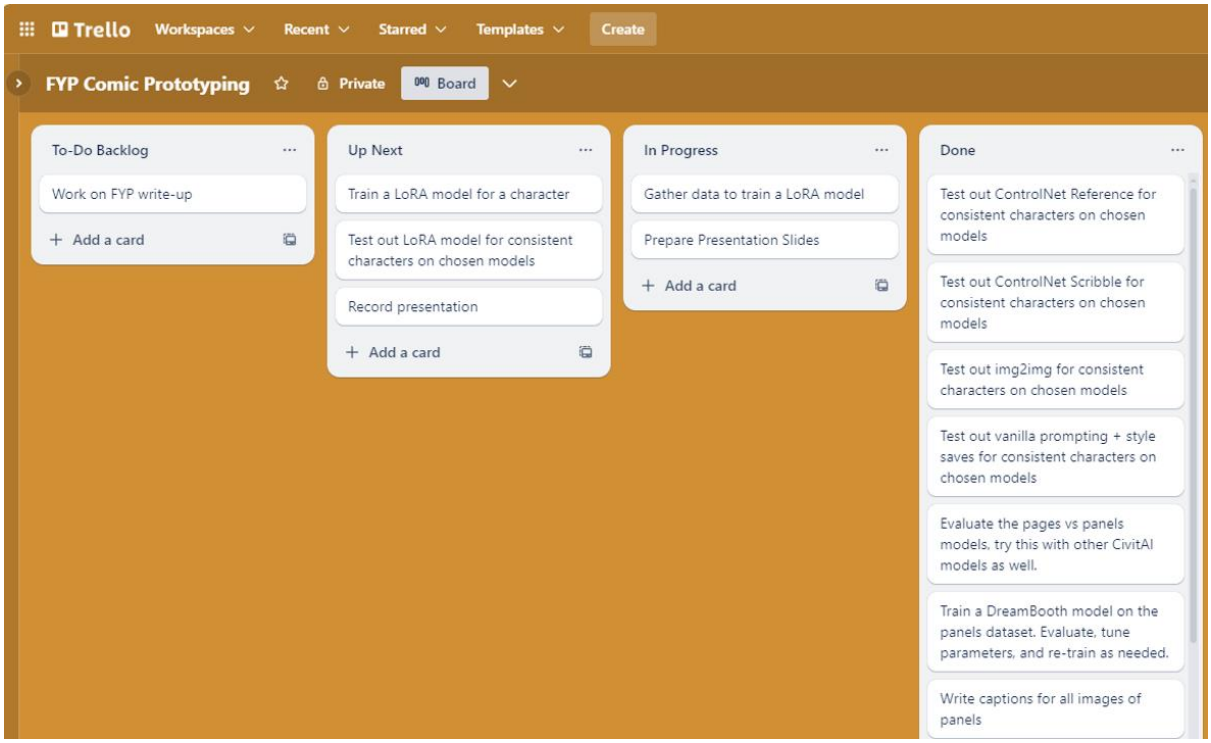
Chair of the College of Engineering, Design and Physical Sciences Research Ethics Committee

Brunel University London

C. Appendix – Project Management

C.1. Trello Board

The following screenshot was taken some time during the second fortnight of February, when I was working on the demonstration of my work for my second marker. The latest screenshot of the Board is not included as there are no task cards in the backlog and up-next categories, and the dissertation write-up is the only item in-progress.



C.2. Git

Link to GitHub:

https://github.com/TheAughat/fyp_comics

D. Appendix – Trained Checkpoint Models

Link to my Hugging Face profile:

TheAughat (Daniel Rodrigues) (huggingface.co)

The are four comic page models, one comic panel model, and a LoRA repository. The first version of the comic page model was unfortunately overwritten during the training attempts and did not get the chance to be uploaded or saved.

The results of all the pages except for “comicpage4” are of poor quality. “comicpage4” is the best version of the pages model, and “comicpanel” is the best version of the panels model.

Direct links to the best models:

Comic Panel:
https://huggingface.co/TheAughat/comicpanel/tree/main

Comic Page 4:
https://huggingface.co/TheAughat/comicpanel/tree/main

E. Appendix – Trained LoRA Models

The trained LoRA models have been pushed to Hugging Face and are available to view in the *output* folder from the given repository below. The LoRA training was done iteratively 6 times. The first two models were overwritten by the succeeding training attempt due to being of extremely poor quality. The third LoRA model trained was of acceptable quality, and thus was preserved. The remaining three models are also available in the repository. Model version 4 in the repository was the 6th and final version of the LoRA trained and is the best model of the lot.

https://huggingface.co/TheAughat/yellowjacketboylei/tree/main

F. Appendix – Gantt Chart