Master thesis : Generative AI methods to create comic strips

Auteur : Charles, Romain
Promoteur(s) : Geurts, Pierre; 22668
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil en science des données, à finalité spécialisée
Année académique : 2023-2024
URI/URL : http://hdl.handle.net/2268.2/19591

University of Liège
Faculty of Applied Sciences
Academic year 2023-2024

Generative AI methods to create comic strips

Graduation Studies conducted for obtaining the Master's degree in Data Science by **Romain Charles**

Supervisors :

Pierre Geurts - University of Liège
Joachim Roekens - Deuse
Maxime Deuse - Deuse

Abstract

The recent surge in interest surrounding generative AI, particularly stable diffusion, highlights their transformative potential in creating realistic content across various domains, from images to text and music. These advancements promise to revolutionize content generation, opening up new creative possibilities. However, challenges persist, notably in ensuring high image quality and consistency.

The challenges addressed include generating minimally pixelated, high-resolution images swiftly, and maintaining consistency across characters, scenes, and style within comic panels. Achieving 100% consistency in stable diffusion remains elusive due to the inherent randomness in AI models trained on diverse datasets.

The research aims to create a tool enabling individuals with limited drawing skills to produce comics using generative AI. Key findings are presented, starting with an exploration of generative AI and stable diffusion, comparing older models with newer SDXL1.0 models, and selecting ComfyUI as the ideal user interface. The study delves into workflows, testing image generation for text-to-image, text-to-image with ControlNet, inpainting, and maintaining consistency through effective prompts.

The research explores alternative solutions, focusing on LoRAs for fine-tuning models and achieving both consistency and flexibility. Tests reveal LoRAs' potential in altering character appearances, generating cartoon-style images, and providing conclusive results for prompt-driven modifications. The integration of LoRAs into ComfyUI is discussed.

In conclusion, the research successfully achieves its primary objectives, showcasing the tool's capability to generate consistent, high-quality comic panels. Despite challenges, the findings contribute to advancing generative AI applications. The implications extend to potential uses in the creative industry, emphasizing the tool's adaptability and user-friendly nature.

Acknowledgments

I want to express my deep appreciation to everyone who played a role in the success of my civil engineering thesis with a specialization in data sciences. This academic endeavor has been both challenging and rewarding, and I am sincerely grateful for the support and guidance I received.

First and foremost, I extend my profound thanks to my thesis advisor, Pierre Geurts, for his availability and invaluable feedback. In a broader sense, I am immensely thankful to the entire faculty for creating an intellectually stimulating environment that nurtured my academic and professional development. This research experience has broadened my perspective and enhanced my learning journey.

I extend a heartfelt thank you to the team at Deuse, the computer company where I completed my internship. The practical experience and exposure to real-world applications of data sciences in a corporate setting were enlightening, especially in the dynamic field of generative AI. I have learned a lot in a short period of time in an area that has been evolving rapidly in recent years. My gratitude goes particularly to Joachim Roekens, my internship supervisor, and Maxime Deuse for their guidance, advice, and encouragement. Their benevolent support and generous sharing of knowledge have been instrumental. Contributing to their ambitious project has been a source of pride, and this intensive internship has set a positive tone for my professional journey.

I also want to express my thanks to my family and friends for their unwavering support throughout my academic journey. Their belief in my abilities has been a constant motivator. Special thanks to Julien Simar for providing support and instilling the confidence I needed during both favorable and challenging times.

In summary, this thesis marks the culmination of years of dedication and hard work. I am thankful for the opportunities, mentorship, and support that have shaped my academic and professional growth. My sincere gratitude to all who have been part of this journey, contributing to the success of my civil engineering thesis with a focus on optional data sciences.

Sincerely,
Romain

Contents

Chapter 1

Introduction

Before developing the technical aspects and methods used in this practical work, a contextual background followed by a brief introduction to the implementation of the tool will be considered. Any challenges arising from this will then be discussed.

1.1 Context and motivation

In 2010, Julien and Maxime Deuse, 2 brothers, launched an international platform for Korean manga translated into English, which quickly established itself as the forerunner of Webtoon in Europe. The initiative was a huge success, with up to 100,000 readers a month, and it attracted a lot of interest. In fact, they ended up selling their platform to a major Silicon Valley entrepreneur following an offer they couldn't refuse. Thanks to this sale, they were able to launch their own company to put their passion for digital technology to work for project owners and, more recently, for their own projects.

Artificial intelligence is growing exponentially. Its evolution has been marked by significant progress in recent years, from rule-based systems to machine learning and deep learning techniques. These advances have been made possible by increased computing power, innovative algorithms and increasingly large datasets. As a result, the convincing results of using AI systems are making them popular. There is more and more talk about them, and this is giving ideas to companies, like Deuse, that want to innovate and have the opportunity to jump on the bandwagon.

Moreover, even more recently, generative AI, and in particular stable diffusion, have become hot topics because of their transformative potential. They have demonstrated remarkable capabilities in the creation of realistic content, from images to text and even music, pushing back the limits of creativity and automation. These advances are gaining in importance today as they promise to revolutionise content generation enabling new possibilities.

That's why creative companies like Deuse are jumping on the bandwagon and thinking up innovative products involving generative AI and image generation in particular. What's more, the Deuse brothers saw an opportunity to link their early experience and passion for Webtoon with the emergence of artificial intelligence. They seized it by wanting to solve a problem that some people wishing to create their own comic strip might have.

1.2 Idea

Today, when creating a comic strip, there are two important aspects to consider: the story and the graphics. As a general rule, a writer will be good at one or the other. Many authors are not good at drawing, but they can write good stories. Moreover, sketching art for comic strips can take long hours of work, it's quiet cumbersome and time consuming.

The idea of the project is therefore to enable authors with average or even mediocre drawing skills to produce strips with a certain graphic and aesthetic style. These people will express their creativity by using generative AI to generate high-quality images tailored to their story in just a few minutes using tools such as prompts, drawings and image retouching. They select their favourite images and then they can assemble them one below the other, adding the script and speech bubbles by hand on each one, to create their own unique, high-quality comic strip.

The ultimate aim is to produce a web and/or mobile solution tool with a professional finish and acceptable results. The aim is to ingeniously circumvent the limitations of this type of artificial intelligence, in order to achieve a convincing result.

1.3 Benefits of generative AI for comic strips

Considering the potential advantages of such a tool [1], there are various technical benefits and practical applications to explore.

- Democratizing artistic expression : A tool like this enables people with a wide range of artistic abilities to create captivating visual stories by expressing their ideas and emotions in a unique and accessible way. It opens up artistic avenues to people who might not otherwise have considered creating comics as a form of self-expression.
- Educational value : This tool could be a valuable educational resource, helping users, particularly students, to become familiar with storytelling, visual communication and the art of sequential storytelling. It would foster skills related to plot development, character construction and visual composition.
- Efficiency and time-saving : AI-generated comics can be considerably faster than manual creation, making it a practical tool for content creators. This efficiency allows users to focus more on storytelling and less on technical aspects.
- Entertainment : Beyond its educational and professional applications, the tool offers an entertaining and attractive hobby for users who simply want to have fun by creating and sharing comic content that suits their tastes.
- Accessibility : It makes the art of comic creation accessible to a wider audience, including people with physical limitations or disabilities that might hinder traditional drawing or illustration. It could therefore promote inclusion and diversity in the art world.
- Customization and personalization : Users can customise characters, settings and storylines to suit their preferences, fostering a sense of ownership and personalization of their creations. It's a versatile means of communication and storytelling.

- Collaboration : The tool facilitates collaboration, allowing multiple users to work together on comic book projects, making it ideal for group work, creative collaborations and team-building exercises in order to produce original content.

1.4 Setting up the tool - framework description

In this section, the configuration chosen to achieve our objective from front-end to back-end will be briefly described. Here below, we can see on Figure 1.1 the project configuration.

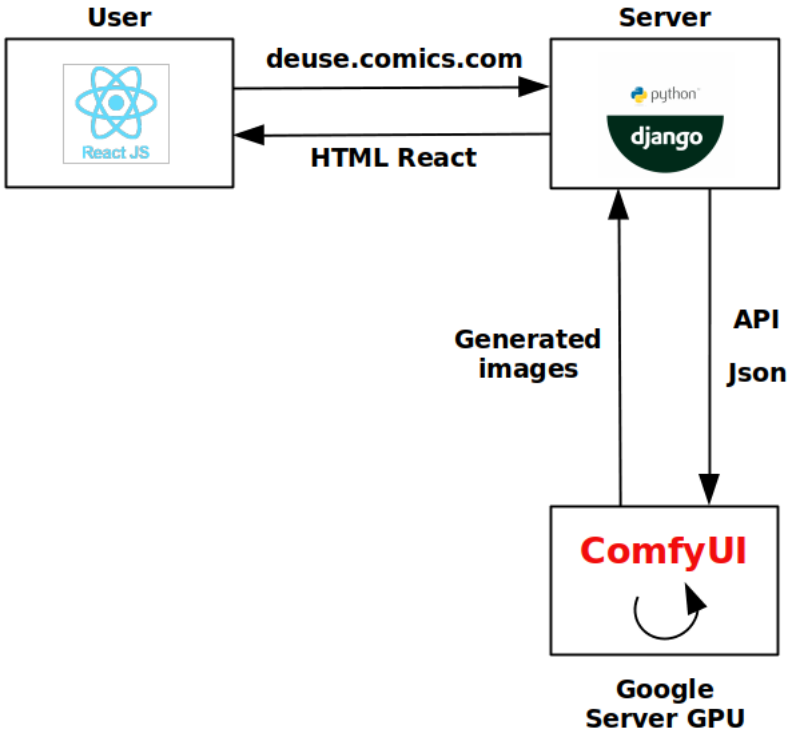


Figure 1.1: Project configuration

Firstly, an interactive web user interface has been created using the React JavaScript library. It is from this interface that users will be able to create their comic strip. This React interface (front-end) will interact with a Django server (back-end) using HTTP requests to send data to the server, as well as receiving data from the server in HTML format. Then, depending on the data received from Django, the user interface will be updated. This creates an interactive and responsive user experience while leveraging the power of React to build the interface and Django to manage the data and server-side logic.

For image generation, we'll be using stable diffusion which is an open-source generative AI implementation. For ease of use and to be able to control how the images are generated, we'll use ComfyUI [2, 3] which is an existing node-based graphic user interface (GUI) for stable diffusion. It was created in January 2023 by Comfyanonymous [4]. It's one of the most powerful and modular interfaces. In fact, even StabilityAI, the creators of stable diffusion, are testing Stable Diffusion internally using ComfyUI and have hired the Comfyanonymous collective to help them work together on in-house tools.

In a nutshell, the interface lets you build image production workflows by linking together different blocks (called nodes). Some commonly used blocks are loading a control point model, entering a prompt, specifying a sampler and so on. ComfyUI breaks down a workflow into rearrangeable elements so you can easily create your own. There's no need to code anything, everything is already pre-implemented in the backend. Workflows are saved as Json files. Another interesting feature is that SDXL models, the latest latent image generation models, are supported.

However, image generation requires a lot of computing power. So we needed a high-performance machine to generate good-quality images relatively quickly. That's why Google servers are being used for this project, at least temporarily. Google Cloud's Compute Engine [5] offers the option of adding graphics processors (NVIDIA GPUs) to virtual machine (VM) instances. These GPUs can therefore be used to accelerate specific heavy graphics workloads on the VM, such as deep learning.

Then the communication with ComfyUI is established via an Application Programming Interface (API), which sends data in Json format. This Json document defines the workflow with all the parameters for image generation. The API is a set of rules and protocols that enable different software applications to communicate with each other. They are essential for creating interconnected software and enabling different applications to work together coherently.

Finally, the results - the generated images - are sent to the Django server and then to the user interface. All the user has to do is select the one they prefer and incorporate it into their comic. This process occurs cyclically each time requests are sent by the user to the Django server.

1.5 Challenges

Any project and any initial idea brings, depending on the methods and techniques used, its share of advantages and disadvantages. These problems are often possible to circumvent, mitigate or even resolve following an understanding of the tool and in-depth research. In the present case, the use of stable diffusion offers many advantages but certain limitations are visible.

As a reminder, the goal is to offer users a qualitative experience both in terms of image generation, ease of use and the acceptable time necessary for them to create their comic strip. So, perhaps in the end we will have to make compromises between these different aspects in order to have the best possible output.

Here are some challenges :

- **Image quality :** The images generated must be the least pixelated images possible for relatively large dimensions 1024×1024 in order to provide a pleasant experience to readers. These high resolution images should be generated in line with the prompt relatively fast.
- **Image composition :** A character should be able to be generated in the desired position and it should also be possible to generate images with multiple characters on them. This is not easy to achieve at the moment.

- **Image consistency :** While progressing through the various panels, it is essential to uphold a consistent portrayal of both characters and scenes. This entails ensuring that the elements within the scene, as well as the appearance of the characters and their facial expressions, remain unchanged. The consistency between characters, character clothing and locations is complicated to obtain at the moment, as is a certain style. However, we know that we have to be realistic and therefore that achieving 100% consistency in stable diffusion is not possible. Indeed, AI models are trained on large image datasets. These datasets are diverse and contain a wide range of visual styles and/or objects and/or scenes. The inherent variability in training data can lead to some degree of randomness in the generated images.

We already know that there are different techniques to tackle these problems like good SDXL models, good prompts creation, LoRA training, ControlNet SD plugin, ... It is on these that the foundation of this research work rests.

1.6 Report structure and useful information

The structure of the report is as follows:

- Deuse SRL will be presented in Chapter 2. The analysis will focus on a number of interesting aspects : the company's history, its services and products, its target sectors of activity, its short- and long-term vision, its various business, management and organisational strategies and, finally, its quality.
- The deep learning concepts necessary to understand the use of ComfyUI, such as generative AI models and particularly latent diffusion models are developed in Chapter 3. We will also look at stable diffusion, explaining its various components and applications and comparing the different SD1.5 and SDXL models. Other interesting concepts will also be covered.
- ComfyUI will be present and the first workflows created using SDXL will be tested and analysed in Chapter 4.
- Several ideas and possible technical solutions to alleviate the limitations caused by the difficulty of having coherence between characters will be discussed in Chapter 5.
- Analyses on LoRAs and the different stages involved in training SDXL LoRA from data collection and captioning to training itself and testing will be seen in Chapter 6.
- The final result obtained and discussed is established in Chapter 7.
- Finally, the contributions made by this work and the directions for future work are finally listed in Chapter 8.

The image generation tests were carried out on an NVIDIA GeForce RTX 4060.

Chapter 2

Deuse SRL company analysis

This thesis is associated with an internship at Deuse. This chapter provides a non-technical overview of the company and its strategic elements. Much of the information in this chapter comes from the Deuse company website [6] as well as my personal experience in the company.

2.1 About

Deuse is a SRL company currently based in Liège, Brussels and Hasselt. It's an IT engineering company. Their team of around 40 people is made up composed of engineers and developers and business and marketing oriented profiles. They are driven by the desire to invest in innovative and daring projects and to become a trusted partner for their customers. Their partners' success is their priority. The company's website is available here.

2.2 Services (and products)

Deuse is first and foremost a consultancy specialising in the creation of digital solutions tailored to its customers' needs, such as mobile applications, web platforms and interfaces, management software (ERP, CRM, progiciel, etc.), IoT systems and other bespoke developments. They also have expertise in data science (databases).

However, they are also keen to develop interesting in-house projects such as Odevio and Stripik, the application for creating webtoon-style comics.

- **Odevio** [7] is a utility designed to assist developers, whether working independently or collaboratively across various operating systems such as Linux, Windows, and MacOS. Its purpose is to streamline the configuration and publication process of Flutter applications on iOS, significantly minimizing the time required for installation. Here is the website for this particular project.
- **Stripik** [8] is still a beta version of an application that will enable users to create their own comic strips in a reasonable time by generating panel images using artificial intelligence. The aim is to offer a tool that is easy to use, has a good quality-creation time ratio and enables consistent, yet diverse, high-resolution images to be obtained. It is this project that the thesis research focuses on. Here is the website for this particular project.

In order to carry out all these projects, they employ diverse technologies, such as the Django web framework [9] in Python for the backend, facilitating straightforward web application development through code reuse. The Flutter software development kit [10] is used for the front-end, enabling the creation of cross-platform applications compiled natively from a unified code base. Additionally, the React JS library [11] is employed to simplify the development of single-page web applications. These technologies operate within the Docker architecture [12], designed to assist developers in creating, sharing, and running containerized applications.

2.3 Activity sectors

They develop unique tools for their customers in a wide range of sectors : healthcare, events and the general public, construction, industry, human resources, marketing and sales, public services, etc.

2.4 Vision and mission

The team of skilled engineers is motivated by a commitment to invest in cutting-edge and bold initiatives, creating digital solutions tailored to address the intricate challenges faced by businesses. The objective is to provide companies with a high-quality tool that sets them apart from competitors, ultimately establishing a trusted partnership. The selection criteria prioritize innovation and the significant impact of a project.

The company envisions substantial growth in the future and is actively pursuing this goal. As part of this expansion strategy, new offices have been established in Hasselt, Flanders, while the current offices in Liège, now deemed too small, will be relocated to a larger three-story building. Over the past few months, the team has grown from 30 to 40 members. Additionally, there is a strategic ambition to launch more enterprise software projects (such as ERP and CRM) and initiatives involving artificial intelligence, seizing the existing opportunities in these domains.

Furthermore, the company aims to embrace sustainable development by pursuing a three-year sustainable entrepreneurship certification offered by the CCI Wallonia[1]. The objective is to align the company with the seventeen sustainable development goals outlined by the United Nations since 2015.

2.5 Strategy

The strategic elements of a company encompass factors that can provide it with a competitive edge. This segment elucidates the strategic measures adopted by Deuse from perspectives such as organization, human resources, research and development, and marketing.

[1]CCI Wallonia corresponds to "Chambre Wallonne de Commerce et d'Industrie". It manages the network of Walloon business services and provides assistance to its members' projects in the Walloon region.

2.5.1 Organizational

The staff consists of two CEOs, the Deuse brothers, along with product owners, salespersons, communication professionals, HR personnel, and R&D employees. All are more or less on an equal footing except for some which depend mainly on seniority. Departments are designated based on internal constraints, employing an input-based departmentalization model. Decision-making is centralized.

Most of the meetings are informal, although there are occasional more significant ones. They follow the SCRUM method's daily practice, wherein each morning, every product owner conducts a meeting with the employees working on their projects. During these meetings, everyone shares their plans for the day and sometimes discusses challenges encountered in the project. This fosters mutual support among the meeting participants. Consequently, work coordination within the company is achieved through mutual adjustment, without specific oversight from management. Team monitoring is carried out by 2-3 employees, and the findings are reported to the directors to assess whether everything is going well and if everyone is satisfied.

2.5.2 Human resources management

Deuse is an expanding firm boasting approximately 40 staff members. The cornerstone of its human resources management (HRM) policy lies in its values, which define the principles upheld by the company. Essentially, Deuse places a significant emphasis on fostering team spirit, open communication, and mutual support among its employees, all while maintaining a serious and professional demeanor. The company places great importance on cultivating a positive atmosphere and embracing the youthful energy of its workforce. To achieve this, regular team-building activities and group meals are organized. Deuse is committed to ensuring that every individual not only finds their place within the team but also experiences personal and professional well-being. The company offers in-house training opportunities and actively encourages participation in engaging and diverse projects, aiming to equip employees with a broad spectrum of skills and prevent monotony in their work.

2.5.3 Marketing management

Deuse primarily caters to businesses within its customer base. The company places considerable emphasis on maintaining strong customer relations, utilizing the SCRUM project management method for project implementation. This methodology fosters extensive collaboration with diverse stakeholders through multiple interactions and validation stages, ultimately aiming to develop an optimal product. The SCRUM approach operates on the principles of transparency, inspection, and adaptation, with significant dedication to its implementation to ensure customer satisfaction.

In terms of customer acquisition, Deuse follows a business-to-business (B2B) model. The company relies on its sales representatives to actively engage with clients, participating in conferences and forums to showcase its services. Moreover, Deuse strategically selects bold and innovative projects, collaborating with well-known events and companies like Nomics, Pairi Daiza, Radio Contact (Les étoiles), SPA GP, among others. These collaborations contribute to their visibility through articles, reports, and radio segments, fostering word-of-mouth referrals within the business community. Additionally, Deuse promotes its services through its website and various social media platforms such as Facebook and LinkedIn.

2.5.4 Research and development

Deuse has recently engaged in internal research and development initiatives. They involve a few employees in these projects, but not on a full-time basis, to have specialized experts in the research field. Additionally, they welcome interns each year to assist them in their tasks. This collaboration benefits both parties. On one hand, interns can acquire new skills while benefiting from the guidance and experience of several skilled engineers in their field. On the other hand, Deuse, aiming to be increasingly innovative and brimming with new ideas, lacks the time and personnel to bring them to fruition. Thus, the interns provide them with the opportunity to explore these new areas.

Chapter 3

Generative AI principles

Generating high-quality images based on text descriptions is a formidable challenge, requiring a deep understanding of the intrinsic meaning of the text and the ability to create an image in line with that meaning. Recently, diffusion models [13] have emerged as a powerful tool for solving this problem. These models are capable of generating a wide range of images from textual content. Among the diffusion models, the stable diffusion model is one of the most popular because it is an open-source tool, meaning that anyone can easily create fantastic illustrations from simple text. This is the tool we're going to use to generate the images that will be included in the comics.

So, in order to fully understand the mechanism of stable diffusion, which is important to do in order to be able to use the tool correctly, accurately and obtain the desired image generations. We will look at the different layers of the diagram shown in Figure 3.1 below. We'll start with deep learning models and then go deeper and deeper into the diagram by making things more complex until we reach the chapter we're interested in : stable diffusion. The idea behind each layer will be clearly explained, including the important components and sometimes mathematical principles. The advantages and disadvantages will also be briefly discussed where appropriate.

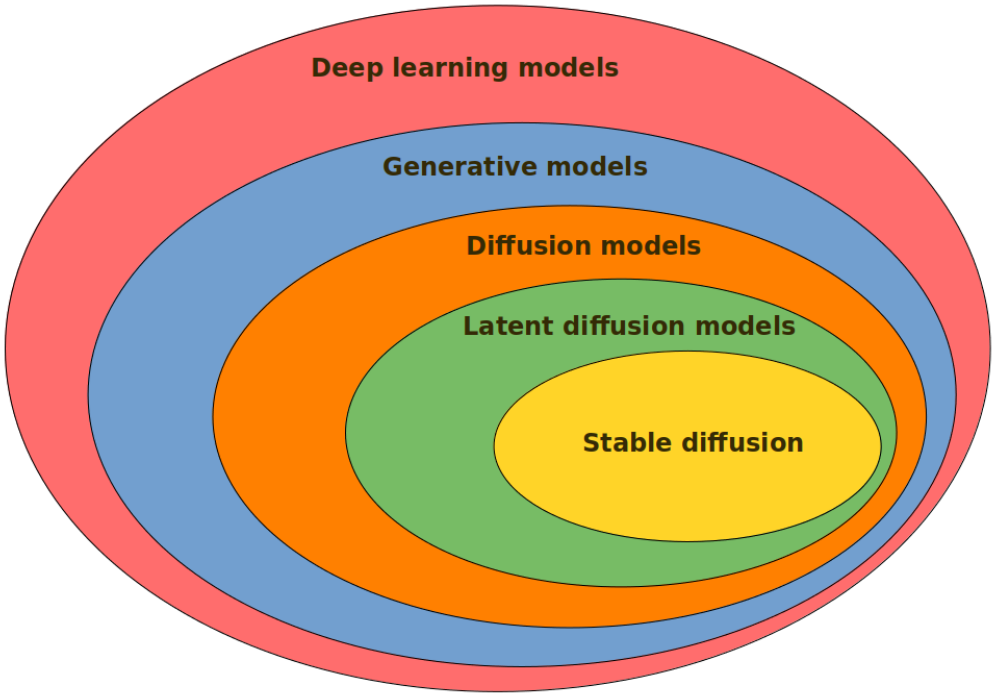


Figure 3.1: Generative AI schema [14]

3.1 Deep learning models

Deep learning models are a subset of machine learning models built on artificial neural networks, inspired by the structure and functioning of the human brain. This is the basis for understanding what happens next. These models have gained widespread popularity and demonstrated remarkable success across diverse domains due to their capacity to autonomously learn hierarchical representations from raw data. This intrinsic feature enables machines to execute tasks and make predictions or decisions without the need for explicit, task-specific programming. Deep learning harnesses the power of neural networks to learn functions that approximate the underlying data distribution.

Here is a technical explanation of some key aspects of deep learning models:

- **Neural network structure :** Neural networks are at the heart of deep learning models. Each neuron in a neural network receives raw inputs, performs a weighted sum of these inputs, then passes the result through an activation function to produce an output. These neurons are organised in layers, generally consisting of an input layer, one or more hidden layers and an output layer that produces the final output. The hidden layers are used to capture and model complex patterns and representations in the data. Here below, on Figure 3.2, we can visualize the general structure of a deep neural network highlighting what has just been said.

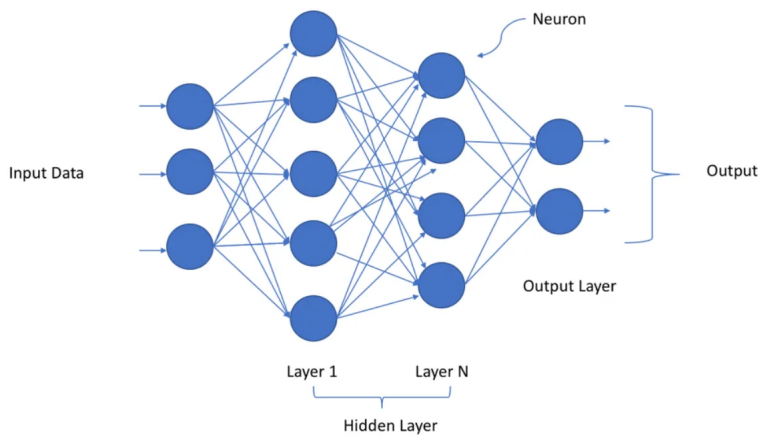


Figure 3.2: Deep neural network [15]

  As implied above, each connection between neurons is associated with a weight, which determines the strength of the connection. Neurons also have associated biases.

- **Activation functions :** Activation functions introduce non-linearity into the model. The most common activation functions are the sigmoid, the tanh and the rectified linear unit (ReLU). These functions enable neural networks to model complex relationships in data. They are essential for modelling the true distribution of data, as real-life problems are usually complex.

- **Training :** Deep neural network models are learned in two important phases : first the forward pass, then the backward pass.

  - *Forward propagation :* During forward propagation, input data is passed to the network layer by layer and calculations are performed to generate the predicted output. The output is then compared to the actual target values and the error is calculated using a chosen loss function.

– *Backpropagation :* Once the error has been calculated, backpropagation begins. This involves calculating the gradients of the error with respect to the weights and biases, starting at the output layer and working up the network. These gradients are computed using the chain rule of calculus. Then the weights and biases are adjusted in the opposite direction of the gradient using optimisation algorithms such as stochastic gradient descent, with the aim of minimizing the error in subsequent iterations. This process is typically repeated for multiple iterations (epochs) until the network converges to a state where the error is minimized.

There are a number of different deep learning architectures, including feedforward neural networks, convolutional neural networks (CNNs) for image data, recurrent neural networks (RNNs) for sequential data, and more advanced architectures such as transformers. Each architecture is adapted to specific types of data and tasks. So, while deep learning models can be used for various tasks, generative models, a subfield of deep learning, use deep learning techniques to create new data resembling the patterns in the training data.

3.2 Generative models

Generative models [16, 17, 18, 19] are a category of unsupervised machine learning and probabilistic models, p, specifically designed for data generation or synthesis. These models can take various forms, handling inputs and outputs such as text, images, sounds, animations, 3D models, and other data types.

The core objective of generative models lies in understanding and capturing the inherent patterns or distributions within a given large dataset. Once these patterns are grasped, the model gains the capacity to produce new synthetic but realistic high-dimensional data that exhibits similar characteristics to the original dataset albeit with some level of variation or noise, $\mathbf{x} \sim p_\theta(\mathbf{x})$. Indeed, the generated data are as close as possible from the unknown distribution $p(\mathbf{x})$ and therefore we have empirical samples. For instance, by training on a dataset filled with images of horses, a generative model can be used to generate entirely new images of horses, ones that may not exist in reality but still possess a convincingly realistic appearance. This feat is attainable because the model has assimilated the fundamental principles that define what a horse typically looks like.

Within the realm of generative models, notable examples include :

- **Generative adversarial networks (GANs) :** GANs consist of a generator and a discriminator that compete in a game. The generator tries to produce realistic data, while the discriminator tries to distinguish between real and generated data. This competition results in the generation of high-quality data samples.
- **Variational autoencoders (VAEs) :** VAEs learn to encode data in a lower-dimensional latent space and decode it to generate new data points. They aim to maximise the lower limit of the log-likelihood of the data. This architecture will be discussed in the next section. VAEs represent a key element in stable diffusion, and, as a result, a dedicated section will delve deeper into this aspect to enhance comprehension (see Section 3.3). This will facilitate a smoother association with the stable diffusion architecture.
- **Flow-based models (or normalizing flows) :** Flow-based models focus on modeling complex probability distributions by using invertible transformations to map simple distributions to more intricate ones.

While these models are proficient at creating high-quality images, they do come with certain limitations that render them less effective than **diffusion models**, which model data distributions by iteratively diffusing and denoising [1] samples to generate data from simple priors. It is this last model that is of particular interest to us, and which we will describe and attempt to understand in detail in Section 3.4. The overall structure of all these generative models is shown in Figure 3.3.

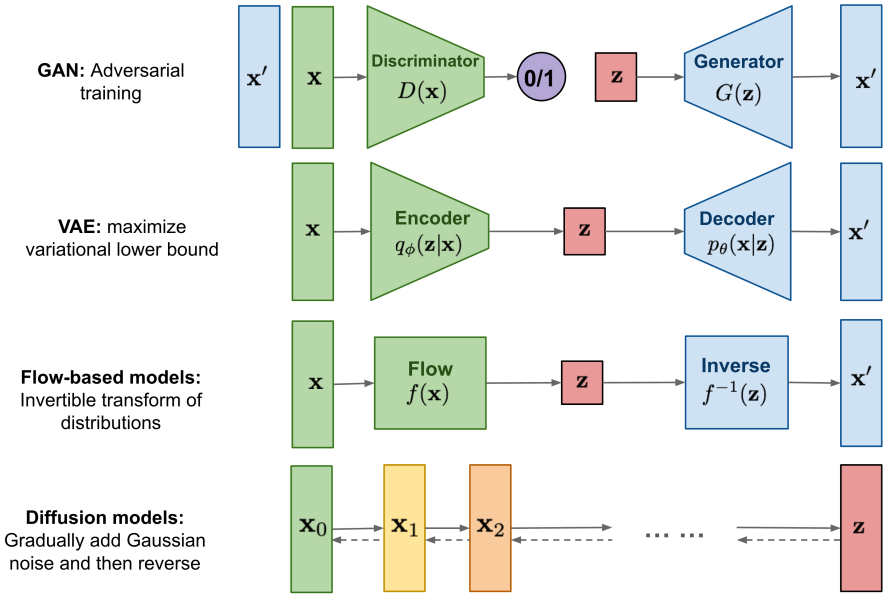


Figure 3.3: Generative models overview [20]

The success of a generative AI model hinges on meeting three essential criteria [16, 21] :

- **High-quality sampling :** In numerous applications, particularly those involving direct user interaction, there is a strong demand for high-quality generation. For instance, in the context of image generation, the objective is to produce results that closely resemble natural images, making it difficult to distinguish between the generated and real ones.
- **Modal coverage and sample diversity :** When the training data exhibits substantial complexity and diversity, an effective generative model should adeptly capture this diversity while maintaining the quality of its generated outputs. This contributes to the mitigation of unintended biases in the learned models.
- **Fast, computationally inexpensive sampling :** Numerous interactive applications demand swift generation capabilities, particularly in scenarios like real-time image editing, to facilitate their integration into content creation processes. This not only enhances user experience but also contributes to reducing the environmental impact of operating resource-intensive deep neural networks that serve as the foundation for generative models.

[1] Gaussian noise is commonly used, it's not a strict requirement. The type and characteristics of the noise can be adapted to suit the specific modeling task and the nature of the data.

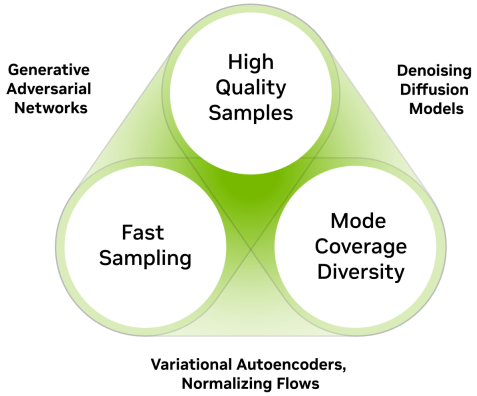


Figure 3.4: Generative learning trilemma [16]

While existing methods often involve trade-offs and may not fulfill all requirements simultaneously, diffusion models show promise in delivering both high-quality outputs and a wide diversity of samples. However, their main drawback is the relatively slow sampling speed compared to traditional GANs.

Fortunately, recent techniques [22] have been developed to address the issue of slow sampling in diffusion models. These methods include latent space diffusion models, critically damped Langevin diffusion, or denoising diffusion GANs. In our study, we focus on stable diffusion models, which consist on a latent diffusion model. It is therefore this type of model that will be the central focus of the remainder of this report.

3.3 Variational autoencoders (VAEs)

Now let's try to understand what VAEs are and how they work. This is important because it is one of the components of stable diffusion. We'll look at this later in Section 3.5.

VAEs [23, 24, 25] are a type of generative model used in machine learning, particularly for unsupervised learning and generative tasks. They combine the concepts of autoencoders and variational inference to model complex data distributions. The idea behind VAEs is that instead of compressing any inputs into a fixed latent vector, as in a normal autoencoder, we want to map inputs on to a distribution. The bottleneck vector, $\mathbf{z}$, is replaced by 2 separate vectors, one representing the mean of the distribution and the other one representing the standard deviation of that distribution.

At its core, see Figure 3.5, a VAE is a deep latent variable model that consists of primary components [23] :

- The prior $p(\mathbf{z})$ is prescribed, and usually chosen to be Gaussian.
- **Encoder :** This part of the VAE comprises a sequence of neural network layers that analyze the input image, extracting its features and transforming them into a lower-dimensional latent space representation of the probability distribution. The encoder is an *inference network*, NN_ϕ, which parameterizes the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ taking as input $\mathbf{x}$ and outputting parameters $\nu = NN_\phi(\mathbf{x})$ to the approximate posterior.

$$\mu, \sigma = NN_\phi(\mathbf{x}), \quad q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu, \sigma^2\mathbf{I})$$

- **Decoder :** This part of the VAE is composed of multiple neural network layers designed to reconstruct the original image from samples in the low-dimensional latent space. The decoder is a *generative network*, NN_θ, which parameterizes the likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ taking as input $\mathbf{z}$ and outputting parameters $\varphi = NN_\theta(\mathbf{z})$ to the data distribution.

$$\mu, \sigma = NN_\theta(\mathbf{z}), \quad p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu, \sigma^2\mathbf{I})$$

Variational inference : Variational inference is a statistical technique that approximates complex probability distributions with simpler ones. In the context of VAEs, it is used to approximate the true posterior distribution of the latent variables given the data. It relies on the Evidence Lower Bound (ELBO), which decomposes the intractable posterior into the sum of the reconstruction loss and the KL divergence loss.

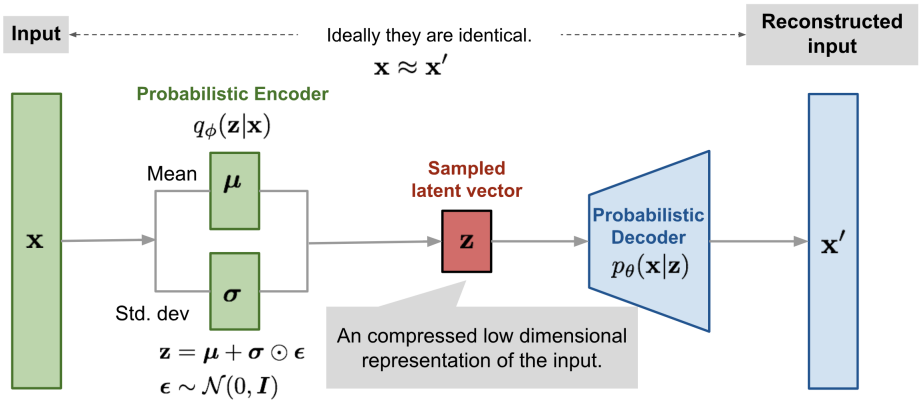


Figure 3.5: Variational autoencoder architecture with gaussian [25]

Loss function : The loss function in VAEs [23, 24] is expressed as the ELBO. It's a lower bound on the log-likelihood of the data. The foundation of this lower bound lies in the non-negativity of the KL divergence. So, by minimizing the loss, we simultaneously maximize the lower bound on the probability of generating novel samples.

- *Reconstruction loss*, the first term on the equation below (1), quantifies how well the model can generate data from latent space. Here we have an expectation operator because we are sampling from a distribution. It's often the negative log-likelihood of the data given the latent space.
- *KL divergence*, the second term on the equation below (2), measures the difference between the learned approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and the prior distribution $p(\mathbf{z})$. Usually, we want make sure that the latent distribution we are learning is not too far from a normally distributed gaussian $\mathcal{N}(0, 1)$.

$$\begin{aligned}\theta^*, \phi^* &= \underset{\theta,\phi}{\text{argmax}}\ \text{ELBO}(\mathbf{x}; \theta, \phi) \\ &= \underset{\theta,\phi}{\text{argmax}}\ \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\underbrace{[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{(1)} - \underbrace{\text{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{(2)}\end{aligned}$$

Reparameterization trick : In the computation graph that we have right now, on Figure 3.6, we have a problem. In the middle of that network, after the bottleneck we have a sampling operation. There is a node there that takes a sample from a distribution and then feeds that sample through the decoder but the problem is that we cannot run backpropagation. We cannot push gradients through a stochastic node.

Indeed, although unbiased gradients of the ELBO with respect to the generative model parameters θ, $\nabla_\theta \text{ELBO}(\mathbf{x}; \theta, \phi)$, are simple to obtain, gradients with respect to the inference model parameters ϕ, $\nabla_\phi \text{ELBO}(\mathbf{x}; \theta, \phi)$, are more difficult to obtain.

We have $\text{ELBO}(\mathbf{x}; \theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{x}, \mathbf{z}; \phi)]$ and so we cannot backpropagate through the stochastic node $\mathbf{z}$ to compute $\nabla_\phi f$. So, in order to run the gradients trough the entire network and train everything end-to-end, we will use the reparametrization trick [23, 24, 26].

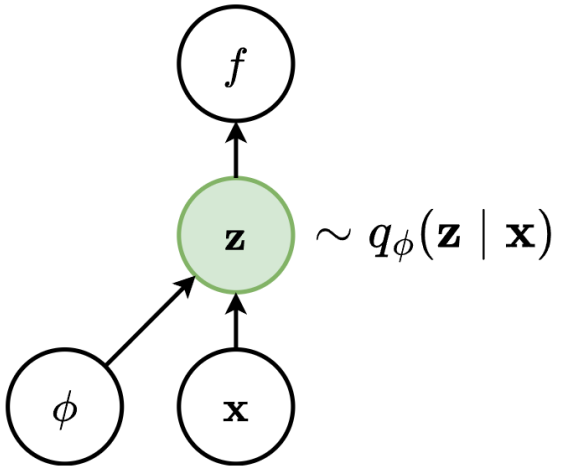


Figure 3.6: Computation graph - Original [23]

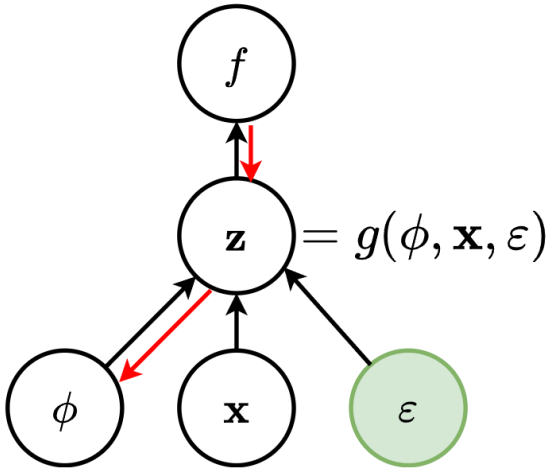


Figure 3.7: Computation graph - Reparametrization trick [23]

The reparameterization trick [24, 26], see Figure 3.7, consists in re-expressing the variable $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ as some differentiable and invertible transformation of another random variable ε given $\mathbf{x}$ and ϕ, $\mathbf{z} = g(\phi, \mathbf{x}, \varepsilon)$, such that the distribution of ε is independent of $\mathbf{x}$ or ϕ. Therefore, instead of directly sampling from the latent space and having a full stochastic node, $\mathbf{z}$, that is blocking all of the gradients because you cant do backpropagation trough it, we are going to split it up into a part where you can do backpropagation and then another part which is still stochastic, ε, but which we don't want to train because it's fixed.

For example, if $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu(\mathbf{x}; \phi), \sigma^2(\mathbf{x}; \phi))$, where $\mu(\mathbf{x}; \phi)$ and $\sigma^2(\mathbf{x}; \phi)$ are the outputs of the inference network NN_{phi}, then a common reparametrization [23] is :

$$p(\varepsilon) = \mathcal{N}(\varepsilon; \mathbf{0}, \mathbf{I})$$

$$\mathbf{z} = \mu(\mathbf{x}; \phi) + \sigma(\mathbf{x}; \phi) \odot \varepsilon$$

3.4 Diffusion models

Diffusion models [13, 27] belong to a category of probabilistic generative models that transform random noise into meaningful data representations. This chapter on the concepts of diffusion is essential to understanding stable diffusion. Through the utilization of diffusion models, we have the ability to create images in two distinct manners : unconditionally or conditionally. In this section, we will focus on the process of unconditional image generation and the process of conditional image generation will be discussed in Section 3.5.

- **Unconditional image generation** simply implies that the model converts noise into a completely arbitrary data representation. The generation process lacks control or guidance, allowing the model to produce images of diverse natures.
- **Conditional image generation** entails providing the model with additional information, whether through text (text2img) or class labels (similar to CGANs). This method guides or controls the image generation process.

Diffusion models entered the realm of deep learning with their initial introduction by Sohl-Dickstein and colleagues in the seminal 2015 paper titled "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". So, they are inspired by non-equilibrium thermodynamics. However, it wasn't until 2020 when Ho and their team published the widely embraced paper "Denoising Diffusion Probabilistic Models" [28], that the research and development of diffusion models gained significant momentum. Subsequently, rapid and substantial advancements have been achieved in this field in a relatively brief period [13].

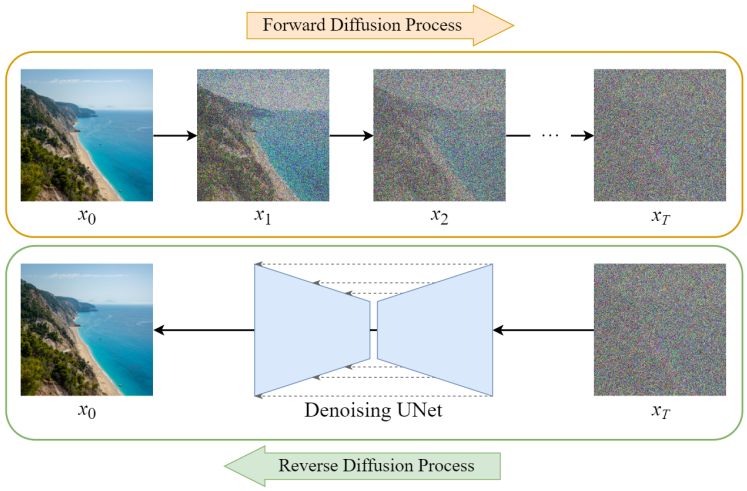


Figure 3.8: Diffusion process [29]

Globally this is how it works (see Figure 3.8) [13, 27], the structure (distribution) of the original image is progressively altered by adding noise following a Markovian chain of diffusion steps, and then restored using a neural network model, meaning that the noise is removed at each step. By repeating this process a sufficient number of times and with good quality data, the model eventually learns to estimate the distribution of the underlying (original) data. Afterward, it is possible to start with a noisy image and use the pre-trained neural network to generate a new image that represents the original training dataset. The forward and backward diffusion processes we've just explained are integral to every diffusion model. Let's delve a bit deeper into them.

Differing from VAEs or flow models, diffusion models are trained using a fixed procedure, and their latent variables exhibit high dimensionality, matching that of the original data [27].

3.4.1 Forward diffusion process

During a forward diffusion process [30], see Figure 3.9, noise is incrementally introduced to a training image, gradually transforming it into a non-representative noisy image. This process is done for some T time steps, i.e., $\mathbf{x}_T$. This forward process has the capability to randomly obscure the original features, making it increasingly challenging to distinguish the initial images. This aspect holds significant importance for training.

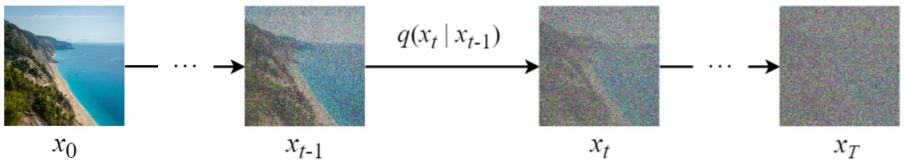


Figure 3.9: Forward diffusion process [29]

Given a data point sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, we can define a forward diffusion process in which a small amount of gaussian noise, $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, is added to the sample in T steps. This produces a sequence of noisy samples $\mathbf{x}_1, ..., \mathbf{x}_T$. A variance schedule, $\{\alpha_t \in (0,1)\}_{t=1}^{T}$, controls the step sizes. So, we have [27] :

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\varepsilon$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I})$$

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

One convenient feature of the aforementioned procedure is that we have the ability to sample at any chosen time step through a closed-form approach employing the reparameterization trick. With $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ and $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we have [28] :

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

With this formula, we are now able to efficiently sample $\mathbf{x}_t$ directly at any given time step, significantly enhancing the speed of the forward process. Note that all ε are standard i.i.d. normal random variables (independent and identically distributed).

Below, on Figure 3.10, we can observe how a noise is added to the image during the forward diffusion process, enabling to create a training example. So, it shows the relation between the equation and the image visualisation.

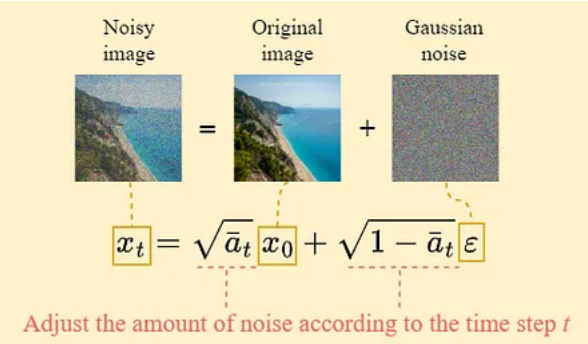


Figure 3.10: A step of the forward diffusion process [29]

After generating numerous training examples for refining the foundational element of our image generation model, we can use this dataset to train the noise predictor, resulting in an exceptional tool capable of generating images when executed under specific configurations. This is what we'll be looking at next.

3.4.2 Reverse diffusion process

Starting with a noisy and unintelligible image, reverse diffusion allows us to reconstruct an image (see Figure 3.11). That's the core concept. In this step, the task is to remove the noise added during the direct process, always iteratively (a Markov chain).

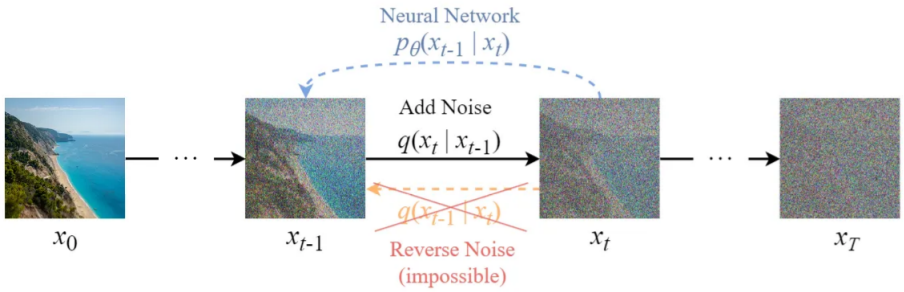


Figure 3.11: Reverse diffusion process [29]

Regrettably, a straightforward estimation of $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is challenging since it requires utilization of the entire dataset. Consequently, we must train a model p_θ, where θ are learnable parameters, to approximate these conditional probabilities to facilitate the execution of the reverse diffusion process. This is done using a neural network model known as a "noise predictor", which is a UNet model in the context of stable diffusion. The latter will attempt to predict the noise added to the image at each step. The denoising process is known as "sampling" in stable diffusion since a new sample is generated in each step. It's often a trade-off between speed and accuracy [27, 30, 31].

With $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we have [31] :

$$p(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_\theta^2(\mathbf{x}_t, t)\mathbf{I})$$

$$\mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \sigma_\theta(\mathbf{x}_t, t)\varepsilon$$

3.4.3 Loss function

For learning the parameters θ of the reverse process, we can form a variational lower bound on the log-likelihood of the data as [31, 29, 32] :

$$\mathbb{E}_{q(\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0)] \geq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] := L$$

This objective can be rewritten as :

$$\begin{aligned} L &= \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\ &= \mathbb{E}_{q(\mathbf{x}_0)}\left[L_0 - \sum_{t>1} L_{t-1} - L_T\right] \end{aligned}$$

where

- $L_0 = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)]$ is the reconstruction term of the last denoising step that can be ignored. One can approximate it using the identical neural network as in L_{t-1}. Doing this, it enhances sample quality and simplifies implementation.
- $L_{t-1} = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\mathrm{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}))$ is the stepwise denoising term. It compares the target denoising step q and the approximated denoising step p_θ. The transition $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ provides a learning signal for the reverse process, since it defines how to denoise the noisified input $\mathbf{x}_t$ with access to the original input $\mathbf{x}_0$.
- $L_T = \mathrm{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p_\theta(\mathbf{x}_T))$ shows how close the distribution of the final noisified input is to the standard Gaussian. It is a constant term that can be ignored since q has no trainable parameters and p is just a gaussian noise probability.

Parameterization of L_t for training loss : The distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is the tractable posterior distribution. You can see it on the computational graph of the reverse process, on Figure 3.12. So, we have :

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\ &= \mathcal{N}(\mathbf{x}_{t-1}; \mu_q(\mathbf{x}_t, \mathbf{x}_0, t), \sigma_t^2\mathbf{I}) \end{aligned}$$

where

$$\begin{aligned} \mu_q(\mathbf{x}_t, \mathbf{x}_0, t) &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}\mathbf{x}_0 \\ \sigma_t^2 &= \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} \end{aligned}$$

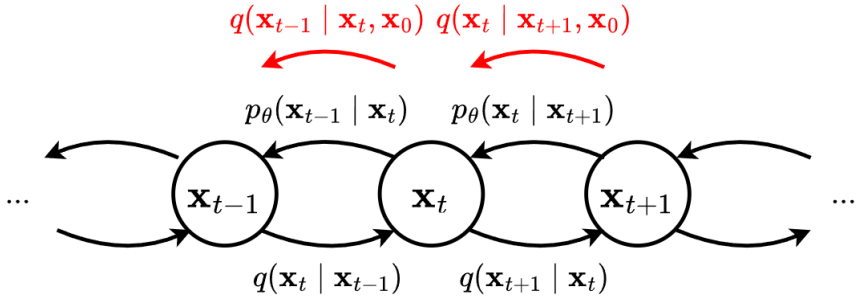


Figure 3.12: Reverse diffusion process with the tractable posterior distribution [31]

Now, using the reparameterization trick :

$$\mathbf{x}_0 = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\varepsilon}{\sqrt{\bar{\alpha}_t}},$$

we can obtain the mean of the tractable posterior :

$$\mu_q(\mathbf{x}_t, \mathbf{x}_0, t) = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{(1-\bar{\alpha}_t)\alpha_t}}\varepsilon$$

As a reminder, the goal is to approximate the conditional probability distributions in the reverse diffusion process, p_θ. To do that we need to learn a neural network, and therefore train μ_θ to predict μ_q. So, since $\mathbf{x}_t$ is available as input to the model, the mean of the reverse process can be parameterized with a *noise-prediction network* as :

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{(1-\bar{\alpha}_t)\alpha_t}}\varepsilon_\theta(\mathbf{x}_t, t)$$

where ε_θ is a function approximator intended to predict ε from $\mathbf{x}_t$.

Then, under this parameterization, we can assess the disparity between the target mean μ_q and the estimated mean μ_θ by employing the mean squared error (MSE). So, the minimization of the expected KL divergence L_{t-1} can be rewritten as :

$$L_t = \arg\min_\theta \mathbb{E}_{\mathcal{N}(\varepsilon;\mathbf{0},\mathbf{I})} \frac{1}{2\sigma_t^2} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \|\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon, t)\|_2^2$$

In their empirical study, Ho et al. (2020) [28, 27] observed that training their models using the true variational bound results in improved code lengths, as anticipated. However, they also discovered that achieving improved results in training the diffusion model is more effective when employing a simplified objective that excludes consideration of the weighting term since it produces the highest sample quality.

$$\begin{aligned} L_t^{\text{simple}}(\theta) &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\varepsilon}\left[\|\varepsilon - \varepsilon_\theta(\mathbf{x}_t, t)\|^2\right] \\ &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\varepsilon}\left[\|\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon, t)\|^2\right] \end{aligned}$$

3.4.4 Training

For the training of diffusion models [32], we have a dataset containing the random time step t selected for each training sample, which is converted in embeddings (vectors), and, the corresponding noisy images, obtained processing the forward diffusion process. These elements compose the dataset used for reverse diffusion process training. We can clearly see in Figure 3.13 an example of a representation of this dataset, taking N epochs for the training.

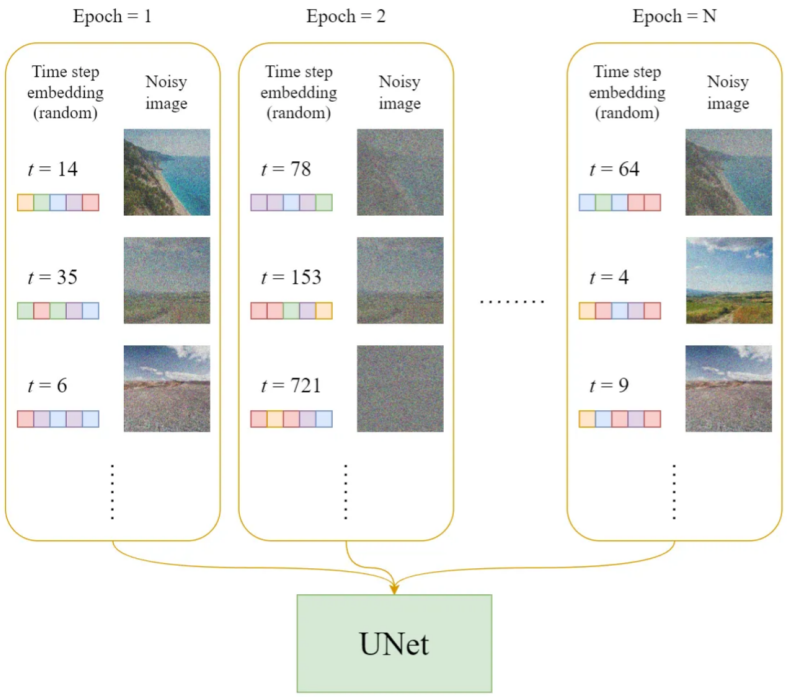


Figure 3.13: Dataset for diffusion training [32]

The UNet training process of a pure diffusion architecture (look at Figure 3.14) can be outlined as follows :

1. Begin with the selection, from the training dataset, of a noisy training image and its corresponding embedding.
2. Predict the random noise added to the image trough UNet.
3. Loss computation : we compare the predicted noise to the actual (labeled) noise.
4. Taking into account the calculated loss, that we tend to minimize, we update the noise predictor by adjusting its weights by backpropagation and gradient descent.

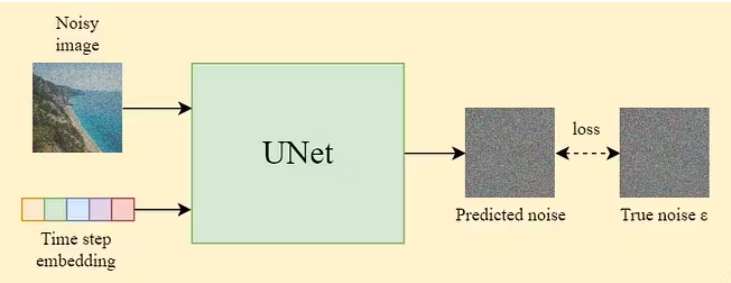


Figure 3.14: UNet training [32]

UNet architecture : A UNet [33], see Figure 3.15, is a symmetric convolutional neural network composed of a ResNet backbone that reduces the dimension of an image through downsampling (contracting path) and subsequently reconstructs it via upsampling (expansive path).

- The contracting part of the network consists of a series of convolutional and pooling layers. The convolutional layers are responsible for capturing features at different spatial resolutions, and the pooling layers reduce the spatial dimensions while increasing the depth.
- The bottleneck connects the contracting path to the expansive path. It typically contains a series of convolutional layers and serves as a bridge between the low-level and high-level features.
- The expansive path consists of a series of upsampling and convolutional layers. Upsampling builds upon the FCN architecture, is used to increase the spatial resolution. And convolutional layers help refine the segmentation output.

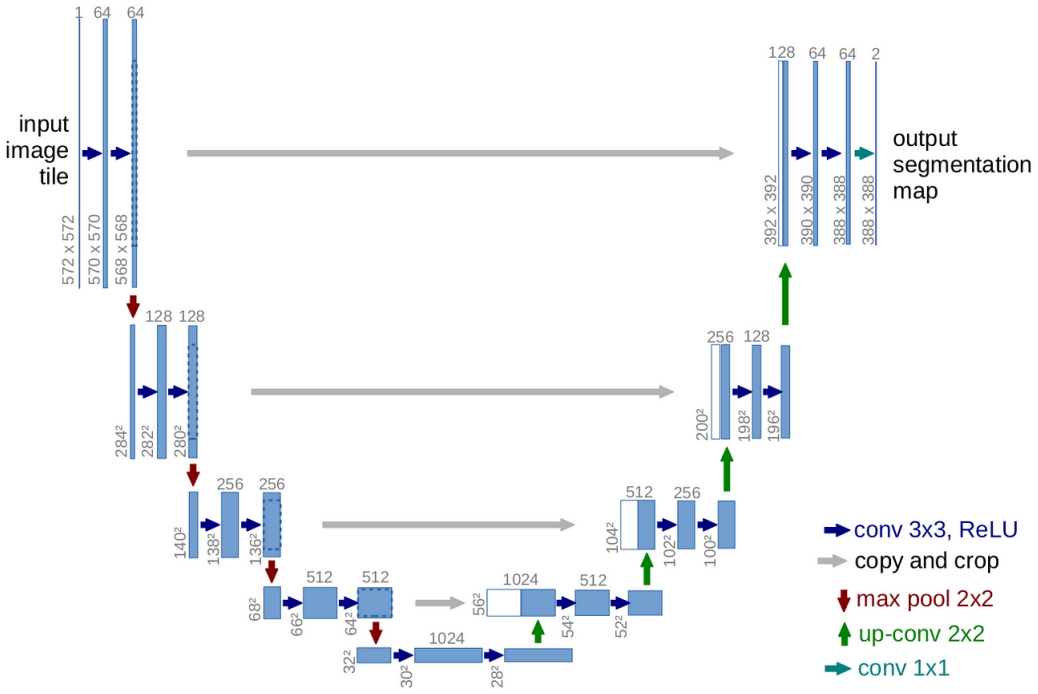


Figure 3.15: UNet architecture [33]

To enhance gradient flow, and, therefore, to address the vanishing gradient problem, skip connections are incorporated between the downsampling and upsampling layers bypassing certain layers. Moreover, these residual connections also help retain high-resolution information during the upsampling process in the decoder part of the network.

Vanishing gradient problem : The vanishing gradient problem [34] (Figure 3.16) emerges in deep neural networks during backpropagation, where gradients diminish rapidly as they propagate from output to earlier layers. This results in minimal updates to the weights of early layers, causing slow learning or stagnation in training. The issue is accentuated with activation functions like sigmoid or tanh, which compress input values into a limited range, leading to extremely small gradients, especially for values far from the origin. We can see an example on Figure 3.17 of the sigmoid that squeezes the input values between 0 and 1. Factors contributing to vanishing gradients include

the use of bounded activation functions, network depth (since each layer multiplies the gradients), improper weight initialization, or a high learning rate.

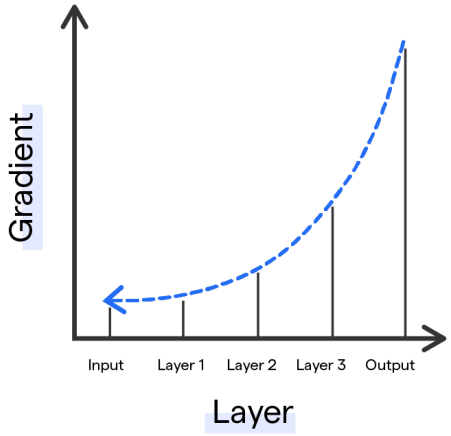


Figure 3.16: Vanishing gradient problem [35]

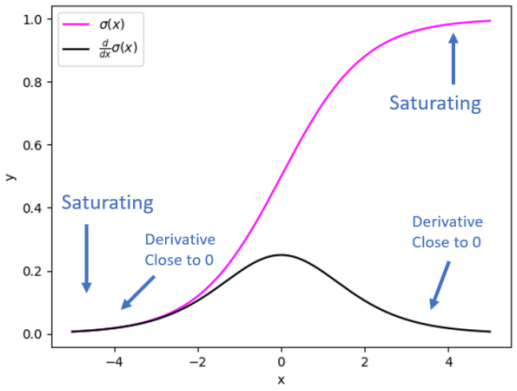


Figure 3.17: Graph of the sigmoid activation function and its derivative [34]

To solve this problem, we can use ReLU activation functions, a proper initialization of weights, batch normalization or also skip connections. Indeed, residual connections [36], look at Figure 3.18, allow gradients to bypass several layers and flow directly to earlier layers. This avoids the need for multiplicative gradient operations and this is useful for capturing long-term dependencies by directly liking the input and output of a layer.

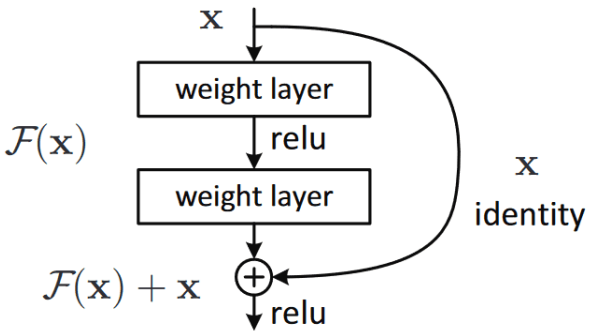


Figure 3.18: Skip connection [36]

3.4.5 Sampling

Upon completing the training, we now possess a noise predictor capable of accurately evaluating the introduced noise in an image. When utilizing this noise predictor, the following steps are followed (see Figure 3.19) [30, 29] :

1. Start with a completely random image, a Gaussian noise for example $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
2. Iteratively denoise the image, doing at each step :
   - Request of the noise predictor, UNet, to estimate the noise present.
   - Then, subtract this estimated noise from the original image.
3. Finally, we get the clean image, $\mathbf{x}_0$, after repeating multiple times as needed the denoising process.

To sample $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is to compute :

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(\mathbf{x}_t, t)\right) + \sqrt{\sigma_t}\varepsilon \quad \text{where} \quad \varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

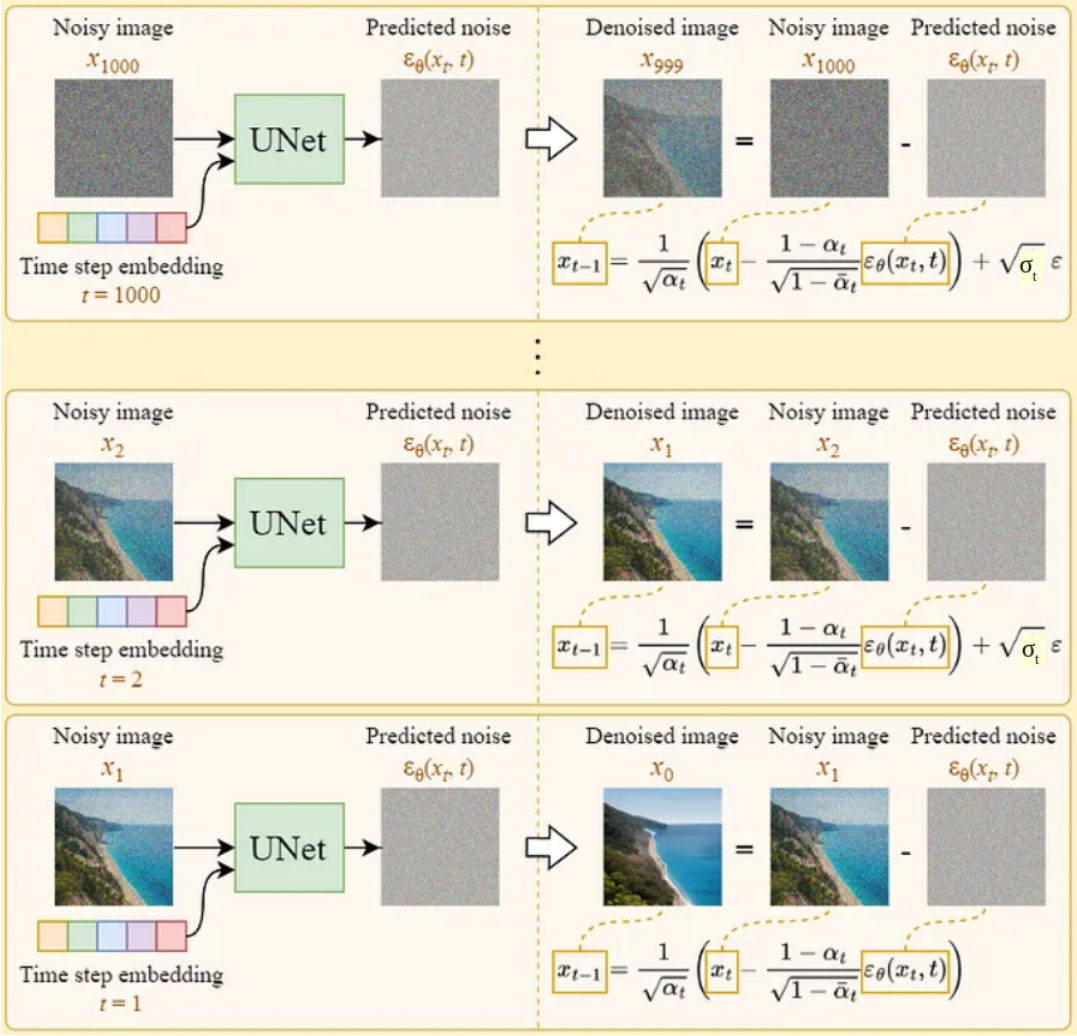


Figure 3.19: Sampling [29]

Please be aware that in the final step, we only produce the acquired mean $\mu_\theta(\mathbf{x_1}, 1)$ without incorporating any additional noise to it.

3.5 Latent diffusion models (LDMs)

One disadvantage of the diffusion models just described is that they are excessively slow on the IT front because of the enormous size of the image space. Indeed, think about a 1024×1024 image with three color channels (red, green, blue), we have here a $1024 \times 1024 \times 3 = 3.145.728$ dimensional space which is very heavy. This makes the pure diffusion model extremely slow.

Fortunately, Latent Diffusion Models (LDMs) offer a novel approach to enhance the speed of image generation. Introduced in the 2022 paper "High-resolution image synthesis with latent diffusion models" by Rombach & Blattmann, et al. [37], this method takes advantages of the perceptual power of VAEs, the detail-preserving capacity of diffusion models and the semantic capacity of transformers by merging all three.

LDMs generate diverse and detailed images while preserving semantic structure, exhibiting greater efficiency and lower memory requirements compared to other models. The key innovation lies in applying diffusion processes within the latent space instead of pixel space, integrating semantic feedback from transformers. The motivation for this approach recognizes that a significant portion of image information conveys perceptual details, even after robust compression [37, 38].

LDMs can be explained through two crucial stages : perceptual compression and semantic compression.

3.5.1 Perceptual compression - Reducing computational cost via latent space

LDMs, drawing inspiration from VAEs, employ perceptual compression by transforming high-dimensional pixel data into a latent space using an autoencoder. This structure involves encoding data into the latent space and decoding to reconstruct the image, capturing abstract representations while excluding high-frequency details. In contrast to diffusing processes in a high-dimensional image space, LDMs compress images into a smaller latent space, reducing data processing, lowering training costs, and, speeding up generation. The auto-encoder efficiently learns a perceptually equivalent space with significantly reduced computational complexity. In the realm of machine learning, this concept is known as the manifold hypothesis [30, 37, 39].

The encoder, $\mathcal{E}$, is used to compress the input image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$ to a smaller 2D latent vector $\mathbf{z} = \mathcal{E}(\mathbf{x}) \in \mathbb{R}^{h \times w \times c}$ where the downsampling rate $f = \frac{H}{h} = \frac{W}{w} = 2^m$, $m \in \mathbb{N}$. Then the decoder $\mathcal{D}$ reconstructs the images from the latent vector, $\tilde{\mathbf{x}} = \mathcal{D}(\mathbf{z})$ [27]. In the paper [37], two different types of regularisation are explored in autoencoder training to avoid latent spaces with arbitrarily high variance :

- *KL-reg :* A small Kullback-Leibler penalty is imposed towards a standard normal over the learned latent. It's similar to a VAE.
- *VQ-reg :* A Vector Quantization layer is used within the decoder. It can be interpreted as a VQGAN with the quantization layer absorbed by the decoder.

The diffusion and denoising processes happen on the latent vector $\mathbf{z}$, as you can see in Figure 3.20.

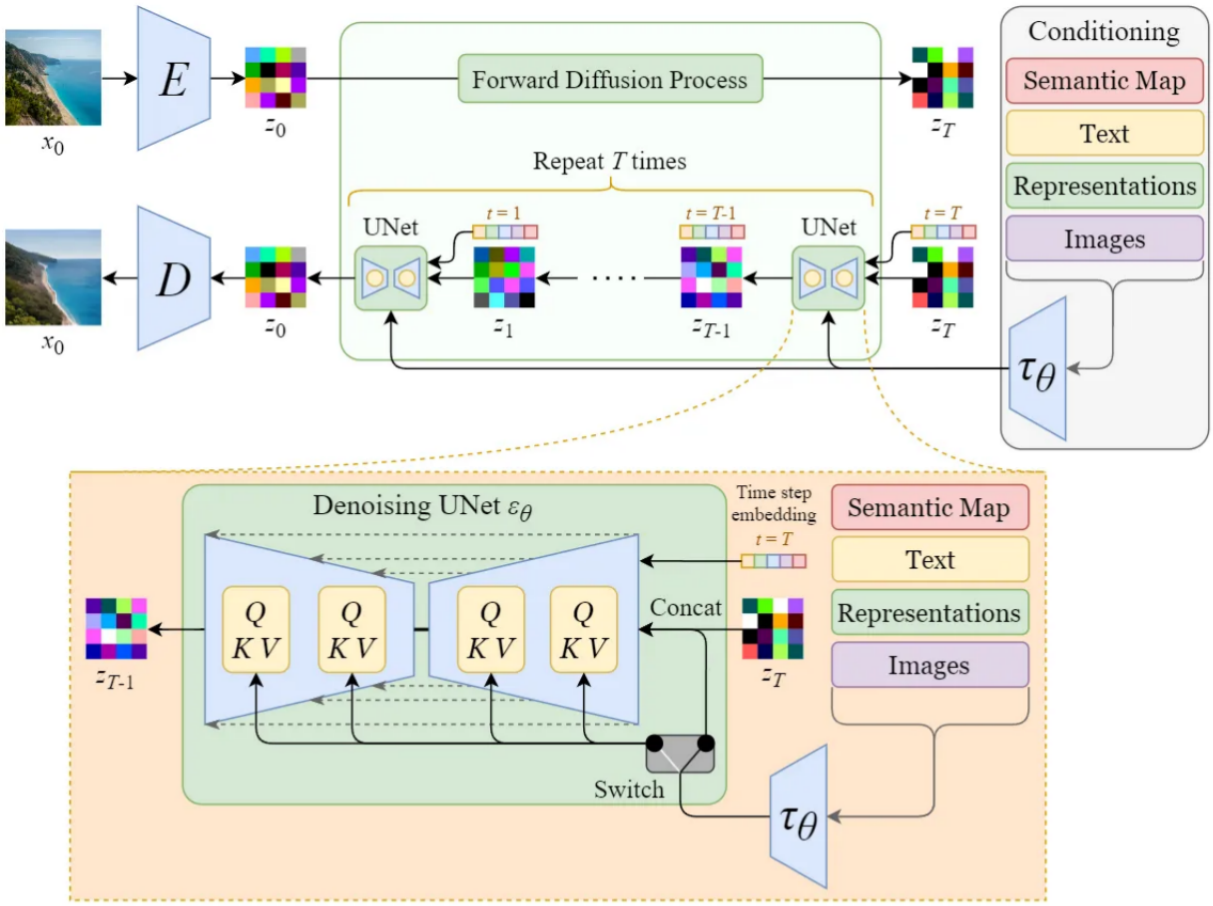


Figure 3.20: Architecture of latent diffusion models [29]

3.5.2 Semantic compression - Conditioning mechanisms

In the field of generative models, the complex interaction between textual and visual elements is an essential factor in shaping the result. So far, the diffusion processes described have focused on creating visually appealing images without relying on textual data. While this produces aesthetically pleasing results, it lacks the specificity needed to control whether the generated result represents a pyramid, a cat or any other subject. This is where the integration of conditioning becomes crucial.

The aim of conditioning is to guide the noise predictor in such a way that the predicted noise produces the desired result when subtracted from the image. In this way, the generative model not only learns semantic and conceptual composition from latent data, but also uses transformers to capture the semantic structure present in both guide textual prompts and/or guide images (images, drawings, semantic maps, human poses (stick man), etc) [40, 39, 30]. Some of guide images will be discussed in the Section 3.8. This integration makes it possible to regulate the specific nature of the images generated, guaranteeing a more nuanced and controlled result.

First, we'll start by looking at the learning technique that will enable us to use prompts to generate the images we want. Then we'll look at the technique that will allow us to integrate this type of control into the diffusion models.

3.5.2.1 Transformer language model mechanism

To enable language understanding from a prompt to a diffusion model, a transforming language model is first required [40, 41]. This is a component responsible for processing the text prompt and generating token embeddings. These now become one of the inputs to the UNet. The choice of language model is important to have high quality generated images. Here, the currently deployed robust delivery model relies on CLIP (Contrastive Language-Image Pre-Training), which is a neural network based on GPT architecture by OpenAI. This model is a combination of an image encoder and a text encoder.

No machine learning model inherently comprehends textual data. To enable a model to grasp the meaning of text, it is imperative to transform the textual information into numerical representations known as embeddings. This conversion process, essential for text understanding, can be delineated into two key components [30] :

- *Tokenizer :* First, the CLIP tokenizer is used to encode the text from the text prompt, i.e., the words are converted to numbers called token. This is done because that's the way the computer understands words.
- *Token To Embedding :* Every token possesses a distinctive embedding vector, representing a value vector. The embedding is determined by the CLIP model and remains constant throughout training. Embeddings are essential as they capture relationships between closely related words, allowing us to leverage this information.

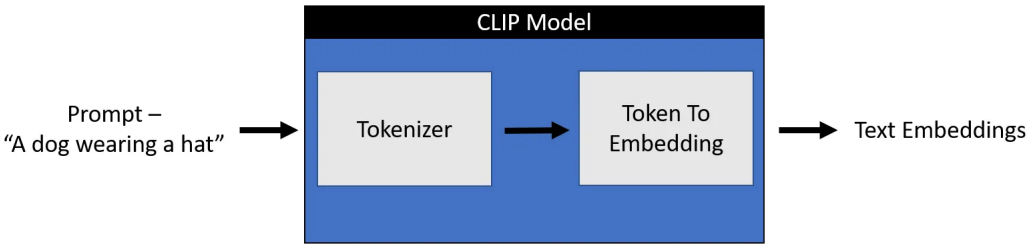


Figure 3.21: CLIP text encoder [39]

Pre-training : Let's delve into the process of CLIP training [40] (look at Figure 3.22). Initially, it undergoes training on an extensive dataset comprising billions of images paired with their respective captions to predict which images were paired with which texts in the dataset. So, it operates by comparing the resulting embeddings through cosine similarity. At the outset of the training, the similarity between embeddings is low, even if the text accurately describes the image. Subsequently, we iteratively update both models so that, the next time we embed them, the resulting embeddings exhibit increased similarity. This iterative process, applied across the entire dataset with large batch sizes, enables the encoders to generate embeddings that are similar in latent space. Notably, the training regimen incorporates negative examples—instances where images and captions don't align—and the model is trained to assign low similarity scores to such mismatches.

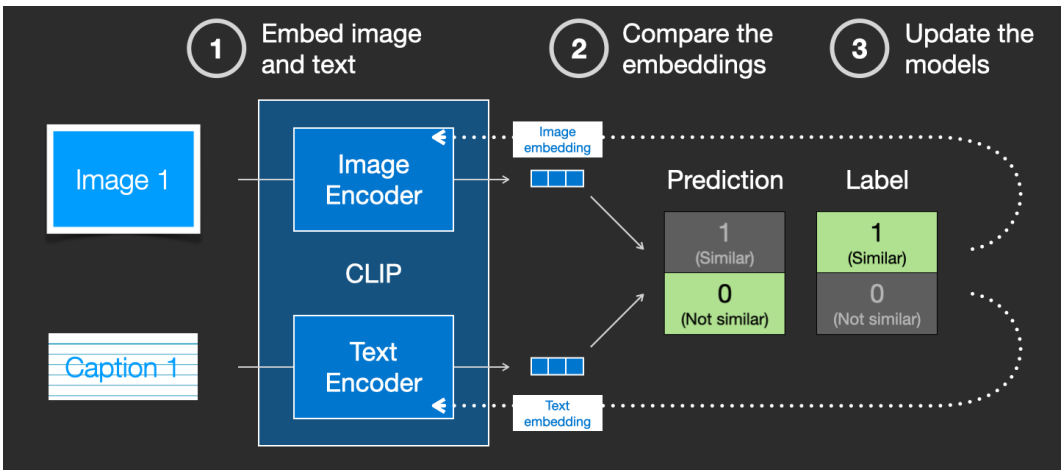


Figure 3.22: CLIP training [40]

Here below, on Figure 3.23, we have a clearer visualisation of the integration of the conditioning mechanism in the latent diffusion model. This integration will be discussed in more detail in the next section. In addition, we'll look at how to control the influence of the text prompt on the generation process in Section 3.6.

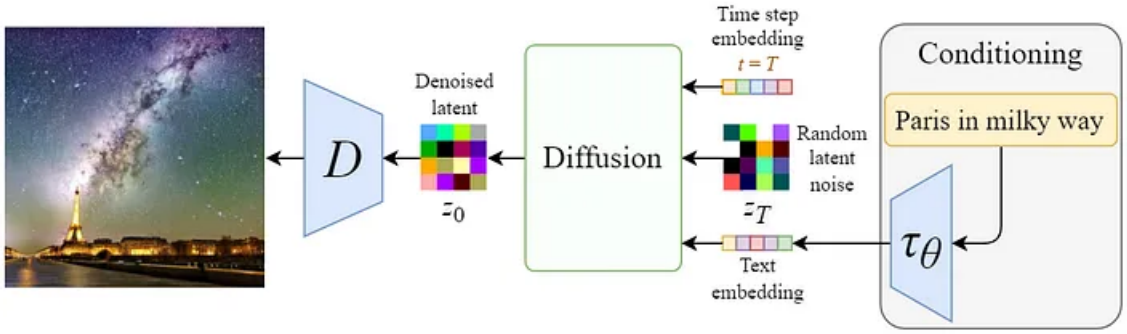


Figure 3.23: Overview of the conditioning mechanism [29]

3.5.2.2 Cross-attention mechanism

The denoising model of LDMs is a time-conditioned UNet, augmented with the cross-attention mechanism to condition transformers with arbitrary types of token-based conditioning mechanisms (text prompt, image, semantic maps, ...). Cross-attention layers are added to both the encoder and decoder part of the UNet, usually between ResNet blocks, as shown on Figure 3.24 describing the UNet architecture of LDMs. Each category of conditioning information is coupled with a domain-specific encoder τ_θ, a transformer (e.g. CLIP), facilitating the projection of conditioning input y into an intermediate latent representation, $\tau_\theta(y) \in \mathbb{R}^{M \times d_\tau}$, that can be seamlessly integrated into the cross-attention layer [37] :

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

where $\mathbf{Q} = \mathbf{W}_Q^{(i)} \cdot \varphi_i(\mathbf{z}_i)$, represents the query matrix which is the current noisy image, $\mathbf{K} = \mathbf{W}_K^{(i)} \cdot \tau_\theta(y)$, $\mathbf{V} = \mathbf{W}_V^{(i)} \cdot \tau_\theta(y)$, respectively are the key and the value matrices got from the conditioning features, and, $\mathbf{W}_Q^{(i)} \in \mathbb{R}^{d \times d_\epsilon^i}$, $\mathbf{W}_K^{(i)}, \mathbf{W}_V^{(i)} \in \mathbb{R}^{d \times d_\tau}$ are learnable projection matrices ; $\varphi_i(\mathbf{z}_i) \in \mathbb{R}^{N \times d_\epsilon^i}$, denotes a flattened intermediate representation of the UNet ; $\tau_\theta(y) \in \mathbb{R}^{M \times d_\tau}$.

The diagram presented above, see Figure 3.20, features a switch employed to regulate various conditioning inputs [29] :

- When dealing with text inputs, the initial step involves transforming them into embeddings (vectors) through a language model τ_θ (such as CLIP). Subsequently, these embeddings are integrated into the UNet via the (multi-head) Attention$(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ layer.
- Alternatively, for other inputs that are spatially aligned, such as semantic maps, images, or inpainting, conditioning is achieved through concatenation.

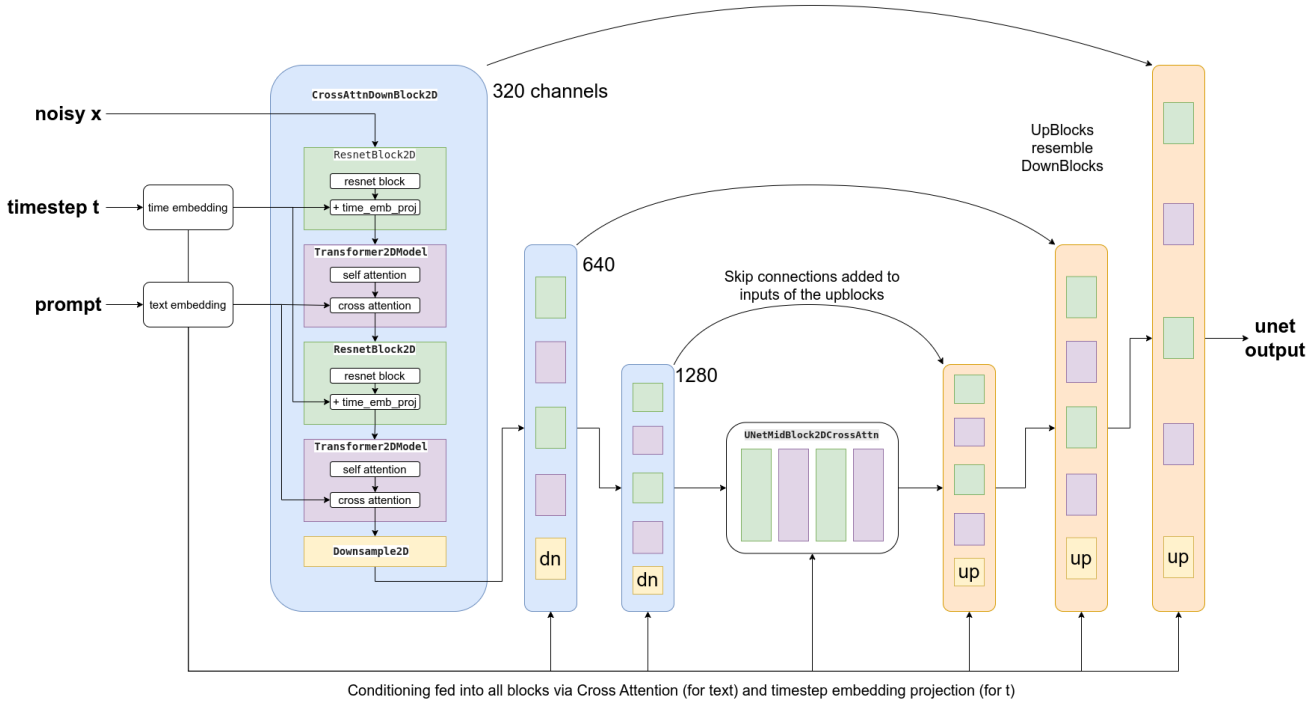


Figure 3.24: Architecture of the UNet in LDM [42]

By example, in the case of text conditioning via a prompt, we are choosing which parts of the text to pay attention to in the attention maps we see in Figure 3.25 below.

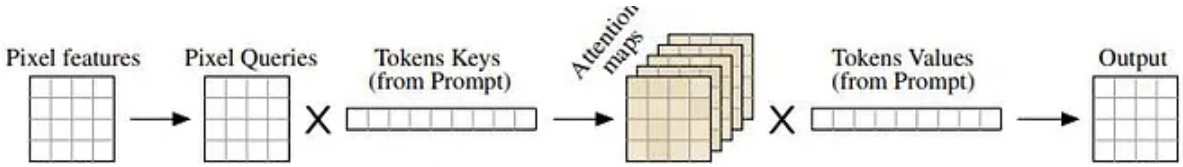


Figure 3.25: Text-to-image cross-attention [43]

Consider the instance of the message "A child with brown hair." In this scenario, the stable diffusion links the words "brown" and "hair" (self-attention within the prompt), resulting in the generation of an image featuring a person with brown eyes but not necessarily a person wearing a brown shirt. Subsequently, this generated information guides the reverse diffusion towards images specifically depicting individuals with brown hair (cross-attention between the prompt and the image).

3.5.3 Training

Hence, the ultimate training objective (loss function) [39] is contingent upon not just the latent space of the original image, $\mathbf{z}_t$, but also on the latent embeddings derived from the conditioning information, $\tau_\theta(y)$.

$$\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon \quad \text{where} \quad \mathbf{z}_0 = \mathcal{E}(\mathbf{x}_0)$$

$$L_{\text{LDM}} = \mathbb{E}_{t,\mathbf{z}_0,\varepsilon,y}\left[\|\varepsilon - \varepsilon_\theta(\mathbf{z}_t, t, \tau_\theta(y))\|^2\right]$$

A significant benefit of adopting this method is the ability to train the universal autoencoding step just once. Consequently, we can leverage it across various diffusion model training sessions or for delving into potentially divergent tasks. This facilitates the efficient exploration of numerous diffusion patterns applicable to a variety of image-to-image and text-to-image tasks [37].

3.5.4 Sampling

Once trained, for inference, image generation is facilitated by executing the reverse process, employing the VAE's decoder to generate the final image converting the latent matrix back into the pixel space. However, it is important to note that even if it has been trained on a large database of images and written descriptions, the model cannot create images that go beyond what it has learned during its training [37].

In the sampling step, the algorithm refines an image through guided conditional sampling and then refines it further with unguided unconditional sampling. The resulting diffusion tends to form recognizable images. The process involves iteratively computing the difference between conditional and unconditional samplings and is repeated for a specified number of steps [44].

3.6 Classifier-free guidance (CFG)

During the training of generative models on images with conditioning information, it is typical to produce samples based on class labels or a snippet of descriptive text. In this section, we'll look at a technique that gives you even more control over what is generated. The aim is to make the stable diffusion model understand how much it must rely on the prompt [27].

To seamlessly integrate class information into the diffusion process, Dhariwal and Nichol [45] developed a classifier, $f_\phi(y|\mathbf{x}_t, t)$, trained on noisy images $\mathbf{x}_t$. They employed gradients $\nabla_{\mathbf{x}_t} \log f_\phi(y|\mathbf{x}_t)$ to steer the diffusion sampling process towards the specified conditioning information y (such as a target class label) by modifying the noise prediction [27]. We call this method "*classifier guidance*".

Even in the absence of a standalone classifier f_ϕ, Ho and Salimans [46] demonstrated the feasibility of executing conditional diffusion steps by integrating scores from both a conditional diffusion model and an unconditional diffusion model. We call this new method "*classifier-free guidance*".

Take, for instance, an unconditional denoising diffusion model $p_\theta(\mathbf{x})$ characterized by a score estimator $\varepsilon_\theta(\mathbf{x}_t, t)$, alongside a conditional model $p_\theta(\mathbf{x}|y)$ also characterized by a score estimator $\varepsilon_\theta(\mathbf{x}_t, t, y)$. These two models can be acquired through the training of a unified neural network. Specifically, the conditional diffusion model $p_\theta(\mathbf{x}|y)$ undergoes training on paired data $(\mathbf{x}, y)$, where the conditioning information y is intermittently and randomly omitted, i.e. $\varepsilon_\theta(\mathbf{x}_t, t) = \varepsilon_\theta(\mathbf{x}_t, t, y = \varnothing)$. This process ensures that the model is adept at generating images unconditionally.

The gradient of a latent classifier can be expressed using estimators for both conditional and unconditional scores. After integrating with the classifier-informed adjusted score, the resulting score is independent of any external classifier. So, we execute sampling by employing the subsequent linear combination of the estimates for both conditional and unconditional scores [46] (see Figure 3.26) :

$$\bar{\varepsilon}_\theta(\mathbf{x}_t, t, y) = (\omega + 1)\varepsilon_\theta(\mathbf{x}_t, t, y) - \omega\varepsilon_\theta(\mathbf{x}_t, t)$$

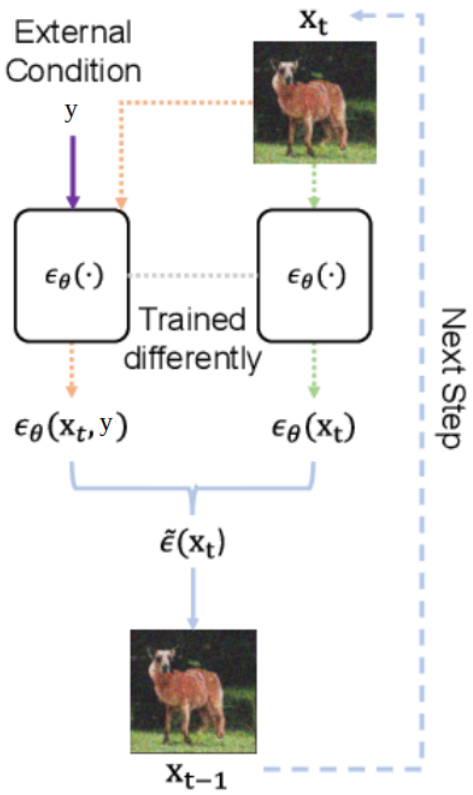


Figure 3.26: Classifier-free guidance [47]

Their studies demonstrated that employing CFG can strike a favorable equilibrium between FID (Fréchet inception distance) which is a metric discerning between synthetic and generated images and IS (Inception score) which is also a metric assessing both quality and diversity [27].

This will allow us to play with the guidance scale ω which is a parameter that influences the proximity of the generated image to the textual prompt. Guidance entails a trade-off : it significantly enhances adherence to the conditioning signal and overall sample quality, but at the cost of diversity [48, 49].

- When $\omega = 0$, we recover an unconditional model.
- When $\omega = 1$, we get the standard conditional model.
- When $\omega > 1$, we employ CFR with the objective of generating a high quality image belonging to class y. The concept is that the class-informed model will produce an output related to the desired class, but the influence of the class signal may vary in strength.

To boost the signal extracted from class information, we have the option to filter out the signal generated by the model lacking class information, resulting in an image that is essentially random. As the parameter ω is augmented, we successively filter out a greater number of "null" images. Theoretically, the removal of information linked to the null class enables a more distinct comprehension of the target class.

Nevertheless, an excessively elevated value for ω removes too much signal from the image, resulting in the generation of essentially random noise. This is due to an excessive removal of signal, which compromises the coherence of the generated output.

3.7 Stable diffusion

At present, there are a large number of diffusion models available (see Section 3.7.4), but most of the recent AI artworks found on the internet are generated using the stable diffusion model. As it is an open-source tool, anyone can easily create fantastic illustrations from simple text, provided you have a graphics card (GPU) with at least 4 GB of video memory (VRAM). Stable diffusion [50] is a kink of diffusion model, called latent diffusion model. Its code and model weights have been released in August 2022 by a group of researchers called CompVis from the university of Munich (LMU Munich) and Runway, with a compute donation by Stability AI and training data from non-profit organizations [51]. The architecture of the stable diffusion models consists of a UNet, a VAE and a text encoder. These models were trained using the LAION database. To simplify the utilization of stable diffusion, you can opt for a graphical interface, enabling convenient browser-based access instead of relying on command line instructions. There are several, including *AUTOMATIC1111*, ComfyUI, ... The one we've decided to use and which we'll talk about in the Chapter 4 is ComfyUI.

3.7.1 Components of stable diffusion

Stable Diffusion [40] is a multi-component system comprised of various models. It is not a single, monolithic model. We can see the composition mentioned below in the Figure 3.27.

1. **Text-understanding component :** To begin with, there is a text encoder component responsible for converting textual information into a digital representation that encapsulates the concepts within the text. This text encoder is a specialized transformer language model, specifically the text encoder used in a CLIP model.

   Input : A text.
   Output : A list of numbers representing each word in the text, each representing a word or token in the text.

2. **Image generator component :** The information coming from the text encoder is fed into the image generator, which undergoes a two-stage process:

   - **Image information creator :** The most crucial component in the system, responsible for a significant performance enhancement compared to earlier models, is the image information generator. It operates through multiple steps to create image data, as defined by the "steps" parameter in stable diffusion interfaces and libraries. This generator exclusively operates in the latent space through the VAE's encoder, distinguishing it from prior diffusion models that functioned in pixel space. Technically, it comprises a UNet neural network and a scheduler algorithm that gradually process/diffuse information in the information array.

     Input : Text embeddings from CLIP and a random starting multi-dimensional image information array (the latents) made up of noise.
     Output : A processed information array. More precisely, it's the predicted noise residual that the input noisy latent contains.

   - **Image decoder :** It's a VAE and its role is to construct an image, in the pixel space, from the data provided by the image information generator. This process occurs once, at the conclusion of the workflow, resulting in the production of the final pixel image.

     Input : A processed information array.
     Output : The resulting image.

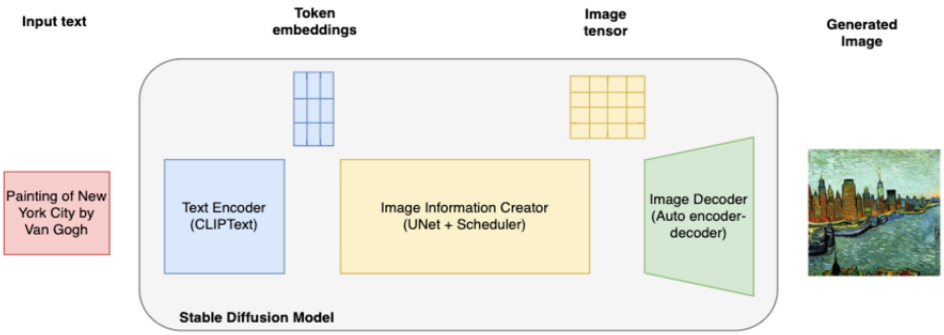


Figure 3.27: Stable diffusion components [52]

3.7.2 Comparison between SD1.5 & SDXL

In this section, we'll start with a technological comparison of the 2 most popular stable diffusion models. Then, we'll make up our own minds by generating and comparing different examples from these 2 models.

The first most popular Runway model before Stable Diffusion XL was SD1.5 released in October 2022. Then, Stability AI unveiled stable diffusion XL (SDXL1.0) in July 2023, an enhanced, larger model than SD1.5 designed for the enterprise. SDXL stands out for its exceptional photorealism, surpassing its predecessors in producing highly detailed images with superior composition. Notably, it can generate descriptive images with concise prompts, create readable text within images, and enhance image composition and face generation. The outcome is a visually striking and authentically realistic aesthetic. The introduction of stable diffusion marks a revolutionary advancement in the realm of text-to-image transformation [53, 54].

As shown on Figure 3.28, the SDXL model consists of 2 models : the base model and the refiner model [53]. The basic model is used to generate noisy latents and describes the general structure. In the following, the refiner model, essentially working as an image-to-image model, is specialized for the final denoising steps and incorporates more complex details. If desired, the base model can be executed independently.

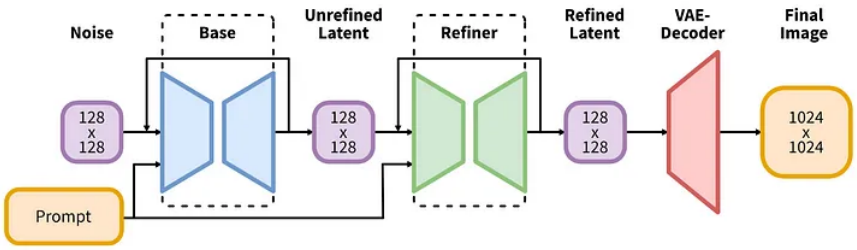


Figure 3.28: SDXL model [55]

As per the research paper titled "SDXL: Enhanced Latent Diffusion Models for Synthesizing High-Resolution Images" authored by Podell and team [55], SDXL demonstrates superior performance across all aspects compared to the SD1.5 model. This improved performance is the result of a number of improvements to the architecture. Here's a detailed list of the differences between the 2 types of model.

Image size : By default, SDXL adopts an image size of 1024×1024, representing a fourfold increase compared to the SD1.5 model, which maintains a size of 512×512 [53].

Number of UNet parameters : The crucial UNet component in the diffusion model has been increased threefold, going from around 860 million UNet parameters to a total of 2.6 billion for SDXL. Leveraging this enlarged linguistic model, SDXL generates top-notch images closely aligned with the provided prompt. The rise in model parameters primarily stems from an expansion in attention blocks and an increased cross-attention context, driven by the utilization of a second text encoder in SDXL [53, 56].

Number of text encoder parameters : The main difference between the different versions of stable diffusion is their use of the CLIP machine learning model [40, 41, 56]. The first stable diffusion models, SD1, are connected to the pre-trained ClipText model published by OpenAI. The SD2 models are connected to the OpenCLIP variants of CLIP, which are much larger but known to have difficulty understanding instructions. This batch includes text models with up to 354 million parameters, compared with 63 million parameters for ClipText. The most recent SDXL models incorporate two CLIP models, which consist of one of the largest OpenCLIP models trained to date, namely OpenClip ViT-bigG-14, with 694.7 million parameters. These models are employed to effectively incorporate textual information into the process of generating images. Additionally, there is an extra text encoder, CLIP ViT-L with 123.65 million parameters, that appends its output, enhancing the conditioning process by introducing supplementary text features. Therefore, the total number of parameters for the SDXL text encoders is more than 800 million parameters. This increases its ability to create realistic images with greater depth and a higher resolution of 1024×1024. Nevertheless, employing numerous extensive text encoders, though appealing, brings about added intricacies that may be disadvantageous.

Total number of parameters : The SDXL model stands out for its significant size, boasting a base model with 3.5 billion parameters and an ensemble model pipeline with a total of 6.6 billion parameters (achieved by combining results from two models). This extensive parameter count enables advanced image generation capabilities. In comparison, the SD1.5 ensemble model pipeline has a comparatively modest 0.98 billion parameters [53].

Complexity : However, although a greater number of parameters may initially seem advantageous, it is essential to weigh up the trade-off between complexity and quality [53, 56]. As the number of parameters increases, so do the training and system generation requirements. So, although SDXL provides better image quality, generation is slower than for previous models and this requires the use of more powerful systems. Indeed, SDXL should run efficiently on consumer GPUs with 8GB of VRAM or on easily accessible cloud instances. 4 GB of VRAM may still work, but it's likely to be a little tight.

Now, we will compare for ourselves the 2 SD models generating images with the same prompts. This will give us a good idea of some of the advantages and disadvantages and, more generally, which model gives the best results. We make sure to compare base models, and in particular the very first base models released. Other models giving better results could be created from these basic models using a fine-tuning method. For all the tests below, the original SD1.5 base model used is "v1-5-pruned.safetensors", the other SD1.5 base model used is "dreamshaper_8.safetensors" and finally the original SDXL1.0 base model used is "sdXL_v10VAEFix.safetensors".

To ensure clarity in the upcoming information, a *positive prompt* involves entering text that outlines the desired elements in the image, while a *negative prompt* involves entering text that specifies undesired elements. We'll discuss how prompts work and how to use them later in a dedicated section, which you can find in Section 5.1.

Test 1 - Real person (full body) :

- *Positive prompt :* realistic photography, a young woman standing in a grassy field, full body, looking at the sky, brown hair, brown eyes, happy, purple dress, sunset, intricate detail, masterpiece, high quality.
- *Negative prompt :* ugly, disfigured, bad hands, bad eyes, poor face, deformed, blurry, long limbs, long arms, poor anatomy, text.

Figure 3.29: Test 1 - SD1.5 vs SDXL1.0 - Real person generation (full body)

We can see that the second basic SD1.5 model chosen, DreamShaper, delivers better quality images than its predecessor. However, the SDXL model still outperforms them. Looking at the first test in Figure 3.29, we can see that the woman's face in the images is not perfect with any of the models. However, it is less disfigured with the SDXL model. It's also clear that the image is more detailed with the new models. This analysis will work for all the tests. The prompt is also a little better respected, as women are more likely to look up at the sky. This is less the case with the SD1.5 models.

Test 2 - Legible text :

- *Positive prompt :* a classroom on the moon with "Happy holidays!" written on the blackboard.
- *Negative prompt :* ugly, disfigured, bad hands, bad eyes, poor face, deformed, blurry.

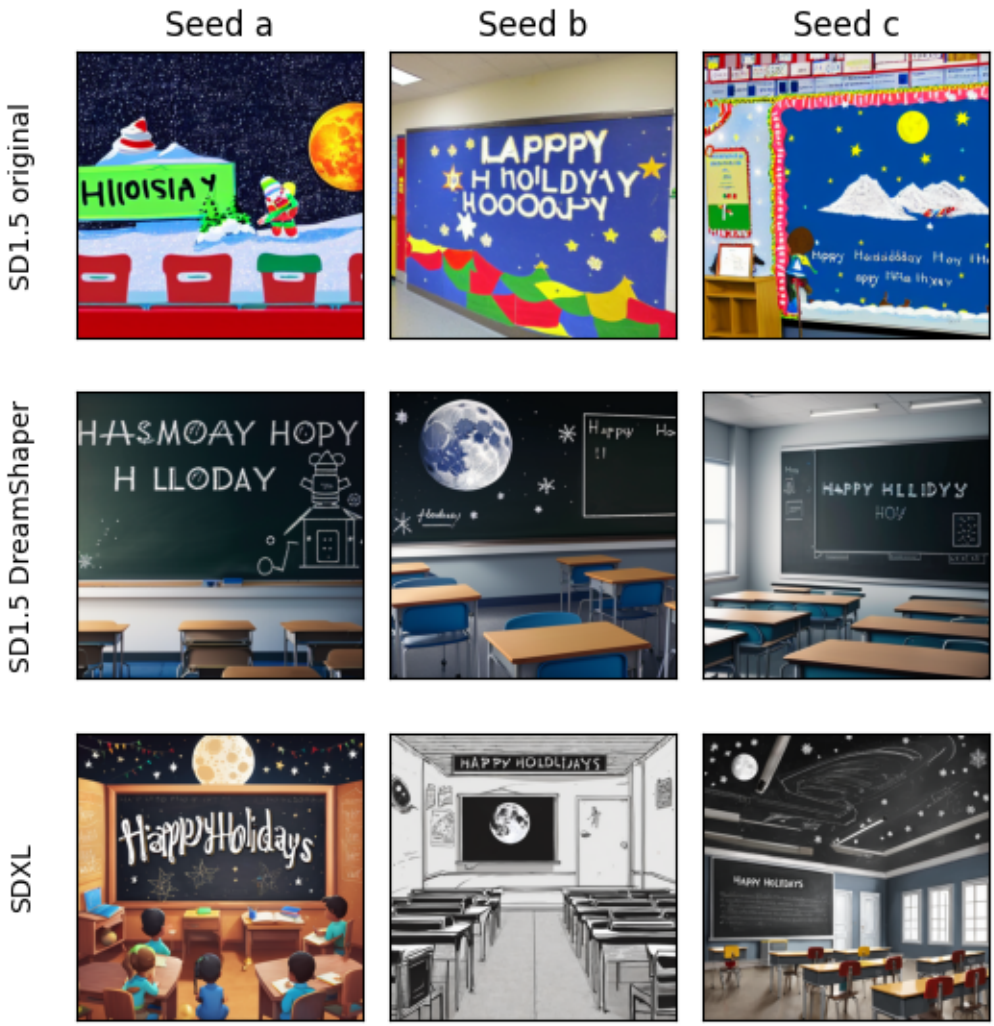


Figure 3.30: Test 2 - SD1.5 vs SDXL1.0 - Legible text generation

In the second test, see Figure 3.30, we try to incorporate text into the image. It's clear that the original SD1.5 base model can't do it. In the end, the image generated doesn't make much sense. As for the second SD1.5 model, it's better, the image makes sense. It's easier to understand what we wanted to generate. However, it's clear that in terms of creativity, detail and, quite simply, quality, the SDXL model surpasses the SD1.5 models completely. In this case, you can read on the board the sentence written in the prompt that you wanted to have on your image.

Test 3 - Simple prompt :

- *Positive prompt :* fruit basket painted in the style of Dali.
- *Negative prompt :* ugly, bad quality, text.

Figure 3.31: Test 3 - SD1.5 vs SDXL1.0 - Simple prompt generation

Here, in Figure 3.31, we can see that the generations obtained with a simple prompt are consistent. However, we can clearly see the difference in quality between the different models. The further down you go, the more beautiful and appealing the image becomes. There are more and more details, colours and elements. The SDXL model produces more sophisticated images, while the prompt is very simple.

Test 4 - Sophisticated prompt :

- *Positive prompt :* a landscape of floating islands with lush gardens drifting in a colourful sky, magical waterfalls cascading from the edges of the islands, fantastic creatures flying between the islands, intricate detail, mastepiece, high quality.
- *Negative prompt :* ugly, disfigured, bad hands, bad eyes, poor face, deformed, blurry.

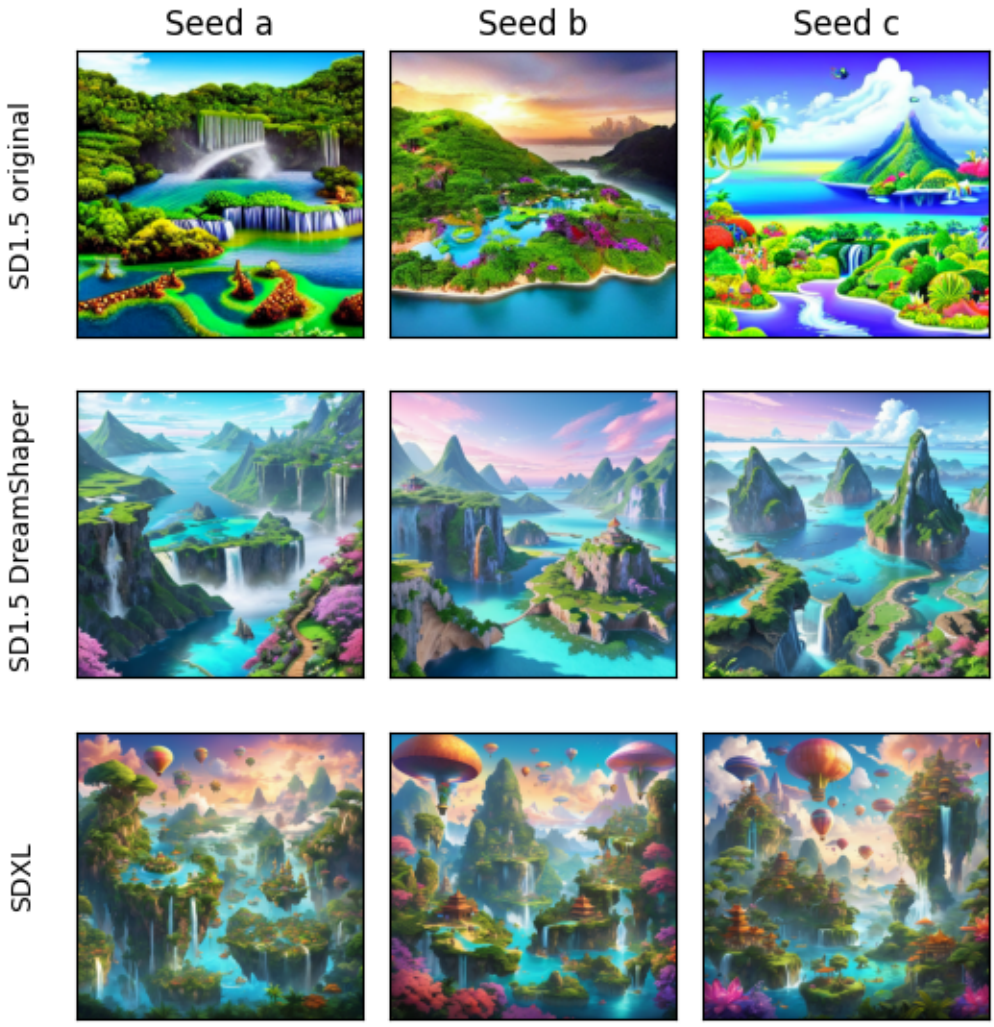


Figure 3.32: Test 4 - SD1.5 vs SDXL1.0 - Sophisticated prompt generation

In Figure 3.32, the result here is unequivocal. With a slightly evasive prompt, the SDXL model comes off well. This confirms previous analyses. Tests 5 and 6, corresponding respectively to generations of close-up real people (Figure 1) and generations of cartoon characters (Figure 2), can be found in the appendix. They teach us nothing more.

Conclusion : Examining the generated images reveals that, overall, the SDXL models provide superior image quality compared to the SD1.5 models. They exhibit a closer adherence to the given prompt, greater complexity in terms of colors, depth, composition, ..., larger image dimensions which improves quality, and the capability to produce images with consistent, legible text and darker aesthetics [53]. These tests have shown that the SDXL models provide high-quality images. That's why we're going to continue working with these brand new models in this project. And, logically, SDXL models excel at image generation. By learning more data and parameters, these models can acquire more complex hierarchical representations of the data, resulting in improved performance.

However, from this fact, the major disadvantage of SDXL is that it requires even greater computing power, which considerably increases image production time. Nevertheless, the models are also brand new and improvements will be made in the future. There are already a few, such as SDXL Turbo [57] or LCM-LoRA [58], which increase the speed of generation. It's not perfect yet, but it shows that there are possibilities to be explored and that it's worth taking the time to understand and use SDXL models now.

Note that there are techniques that can be applied to the generated images to improve their quality. In addition, SDXL models followed by a refiner can be used to refine details, but do not always bring a flagrant improvement to the image.

3.7.3 Various possible applications

Given the open-source nature of the code base, a community of developers and researchers has innovatively explored stable diffusion. This section delves into some noteworthy applications [13] and instances of diffusion models.

3.7.3.1 Text to image generation

Text to image is the simplest way to use diffusion stable image generation models. Just simply add the text description and then start generating the image.

- *Positive prompt :* cartoon style, boy stroking a dolphin on the edge of a wooden pontoon on the sea, short blond hair, blue eyes, rectangular glasses, sunset.
- *Negative prompt :* ugly, bad face disfigured, text, watermark.

Figure 3.33: Example of text-to-image

3.7.3.2 Image to image generation

Image to Image is a method employed to convert a given image into a desired target domain based on a provided text prompt. By offering a textual description of the intended transformation, the streaming model can generate a fresh image maintaining the original color and composition, incorporating the specified modifications.

- *Positive prompt :* a man is skiing in the foreground by night, black suit, pink flowers, realistic.
- *Negative prompt :* ugly, bad face, disfigured, text, watermark.

Figure 3.34: Original image

Figure 3.35: Image obtained after applying image-to-image

Figure 3.36: Example of image-to-image

One practical application of Img2Img is evident in style transfer. This involves utilizing two distinct images—one for content and another for style reference. The outcome is a newly generated image, combining the content of the first image with the stylistic elements of the second.

3.7.3.3 Inpainting and outpainting generation

Image inpainting serves as a technique for restoring images, aiming to eliminate undesired objects or substitute them entirely with another object, texture, or design. The user initiates the inpainting process by outlining a mask around the targeted object or pixels, indicating areas for modification. Subsequently, the model is guided on how to alter the specified masked pixels based on user input. Here below on Figure 3.40 is an example of its utilization.

- *Positive prompt :* anime, lemon tree in the desert.
- *Negative prompt :* ugly, bad quality, (blurred image).

Figure 3.37: Original image

Figure 3.38: Mask

Figure 3.39: Image obtained after applying inpainting

Figure 3.40: Example of inpainting

On the other hand, outpainting involves the diffusion model incorporating additional details outside the boundaries of the original image. This extension of the original image can be achieved either by utilizing segments of the original image or newly generated pixels as reference points. Alternatively, new textures and concepts can be introduced through the incorporation of a text prompt.

- *Positive prompt :* outdoors nature, winter landscape.

Figure 3.41: Original image

Figure 3.42: Image obtained after applying outpainting

Figure 3.43: Example of outpainting

3.7.4 Types of well-known diffusion-based image generation models (Quick View)

Let's examine notable diffusion-based image generation models that gained popularity in recent months. Given the abundance of diffusion models and their variations in the field, we'll provide a brief overview by sampling some of the more renowned ones. We examine these models to determine the most suitable one for our tool and to demonstrate the availability of alternative solutions. So, in this discussion, we won't delve deeply into the mentioned architectures. Instead, we'll offer an overview of each model, exploring a few prompts and examining the images produced by Dall-E 3, Stable Diffusion XL, and MidJourney 5.2.

- **Dall-E 3 :** OpenAI introduced the initial image generator, DALL-E 1, via a blog post in January 2021, utilizing GANs and a modified GPT-3 for image production. However, a new iteration, DALL-E 3, was released in September 2023, featuring a text-conditioned UNet latent diffusion model with three stages and integrating ChatGPT for smooth text-to-image communication [59, 60]. Described in "Enhancing Image Generation Using Improved Captions" [61], DALL-E 3 enhances coherence and image quality, specifically addressing challenges related to unclear captions in training images and improving prompt adherence.
- **MidJourney 5.2 :** Midjourney [13, 62], a diffusion-based image generation model developed by the independent research lab led by David Holz, gained popularity for its expressive style and early availability to the public in February 2022. The latest version, model version 5.2, released in June 2023, boasts improved image quality, higher resolution, and enhanced understanding of natural language prompts. Despite its closed-source nature and self-funded status, Midjourney distinguishes itself from competitors with its powerful generative AI functionalities. The absence of a related paper makes it challenging to understand the underlying mechanisms, but users can access its image generation capabilities through an official Discord bot.

Now that we know a little more about the characteristics of these different diffusion-based models, we're going to compare a few outputs based on different criteria using the same prompts. It should be noted that the examples shown are neither personal nor numerous. They merely give an idea of what would seem to be best achieved with a particular model, i.e. no firm conclusions can be drawn from them.

Test 1 - Coherence
Prompt : "An astronaut riding a steel horse on the moon. The astronaut is wearing a medieval armor with a party hat and a green sword"

DALL-E 3 MidJourney 5.2 SDXL

Figure 3.44: Coherence comparison between DALL-E 3, MidJourney 5.2 and SDXL [59]

Here, in this example, the coherence of the image in relation to the statement seems to be one of the strong points of DALL-E 3. Although a unicorn appears in the image when a horse is required, the rest of the surreal composition of the statement is respected. In comparison, SDXL and MidJourney 5.2 failed to incorporate everything the message asked for - the hat was missing, as was the sword. In this case, it's best if the cartoon creator can easily integrate the elements they want into the image. SDXL doesn't do that perfectly here, which is annoying.

Test 2 - Emotion :
Prompt : "A portrait of a woman holding a wilted rose, her expression one of profound sadness and longing"

DALL-E 3 MidJourney 5.2 SDXL

Figure 3.45: Emotion comparison between DALL-E 3, MidJourney 5.2 and SDXL [59]

In this case, the test was to see whether these models were capable of understanding feelings and emotions so as to be able to transcribe them onto the images generated. The quality of appearance is essential, but so are facial expressions. In the case of a comic strip, the text doesn't represent much compared to the images. So the reader has to be able to understand what's going on just by looking at the image. SDXL seems to manage this better.

Test 3 - Composition :
Prompt : "Design an eco-friendly futuristic city floating on the ocean, with advanced technology and harmony between nature and artificial structures"

DALL-E 3 MidJourney 5.2 SDXL

Figure 3.46: Architecture comparison between DALL-E 3, MidJourney 5.2 and SDXL [59]

Prompt : "A dreamy landscape where clouds are made of cotton candy and rivers flow with liquid gold"

DALL-E 3 MidJourney 5.2 SDXL

Figure 3.47: Surreal landscape comparison between DALL-E 3, MidJourney 5.2 and SDXL [59]

Now we want to compare the quality of the images in terms of composition and originality. Overall, the outputs are cool although we could give a bonus to DALL-E 3 and MidJourney 5.2 because the image produced is more complex.

Conclusion : Overall, the quality of the images generated is impressive. Dall-E 3 seems to follow instructions better and generate images that correspond to specific descriptions. MidJourney 5.2 reportedly produces images that are often stunning and creative, but sometimes lack consistency. SDXL is a good AI image generator, but it seems to lag slightly behind MidJourney and Dall-E 3 in terms of visual aesthetics on the examples provided here. Its great strength is that it's open source and you have greater control over what you can produce (ControlNet, inpainting, outpainting, different styles, etc.). It's mainly for this last reason, and the fact that the image quality is still very good, that stable diffusion XL was chosen for the application. Once again, these are just a few examples. We don't have enough here to be convinced of the superior generation quality of one model or the other but it can give brief overview.

3.8 ControlNet

The most basic form of using stable diffusion models is text-to-image conversion. It uses text prompts as a condition for directing image generation to generate images that match the text prompt. However, using only the prompt can quickly limit the production of the desired spatial composition of the image. Indeed, articulating intricate arrangements, poses, shapes, and forms accurately through text prompts alone can be challenging [63].

Fortunately, Lvmin Zhang and his colleagues came up with ControlNet, see the article "Adding Conditional Control to Text-to-Image Diffusion Models" [64]. ControlNet [65] is an end-to-end neural network architecture that controls image generation in stable diffusion by adding one or more additional conditionals on top of the text prompt, thereby improving performance. This conditioning can take a number of forms, including squiggles, depth maps, human pose skeletons and segmentation maps.

3.8.1 ControlNet architecture

ControlNet introduces extra conditions into the components of a neural network [64, 66], see Figure 3.48. In a standard scenario, a trained neural network block[2] $\mathcal{F}(\cdot;\Theta)$, with parameters Θ, receives a feature map $\mathbf{x} \in \mathbb{R}^{h\times w\times c}$ as input, with $\{h, w, c\}$ as the height, width, and number of channels in the map, respectively, and generates a corresponding feature map $\mathbf{y} \in \mathbb{R}^{h\times w\times c}$, as depicted in Figure 3.48a.

$$\mathbf{y} = \mathcal{F}(\mathbf{x};\Theta)$$

When incorporating a ControlNet into this pre-trained block, we freeze the parameters Θ of the original block and generate a trainable duplicate Θ_c which receives an external conditioning vector $\mathbf{c}$ as its input. When implementing this framework in extensive models like stable diffusion, the fixed parameters safeguard the production-ready model trained on billions of images. Simultaneously, the trainable copy leverages this pre-trained model extensively, constructing a profound, resilient backbone capable of handling diverse input conditions on a large scale. The locked collection of initial weights enables ControlNet to generate consistent outcomes without requiring extensive conditional data, thereby averting the risk of overfitting.

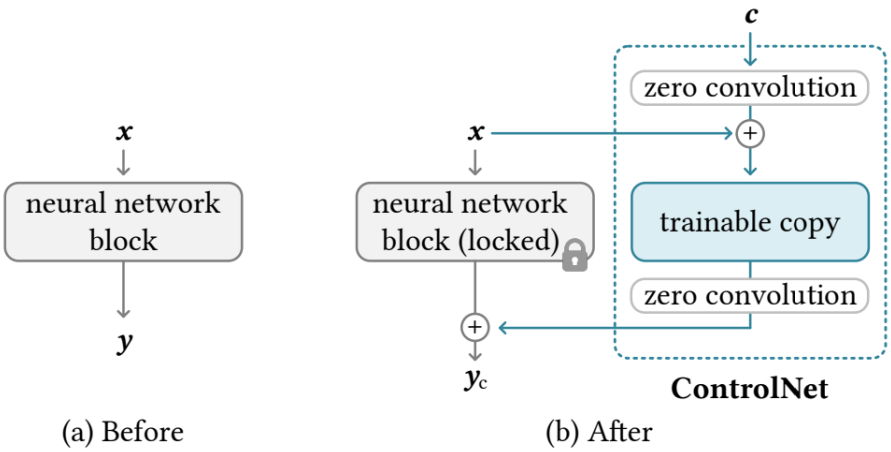


Figure 3.48: ControlNet architecture [64]

[2]Refers to a set of neural layers usually joined together to form a single unit of a neural network, e.g., resnet block, conv-bn-relu block, multi-head attention block, transformer block, etc.

These two blocks are then linked using *zero convolution* layers denoted $\mathcal{Z}(\cdot;\cdot)$, specifically a 1×1 convolution with weight and bias set to zero (see Figure 3.48b). To create a ControlNet, we employ two instances of zero convolutions with parameters Θ_{z1} and Θ_{z2}, respectively. Here below, on Figure 3.49, is the entire feedforward ControlNet with all the computations to reach $\mathbf{y}_c$, the output of the ControlNet block [64].

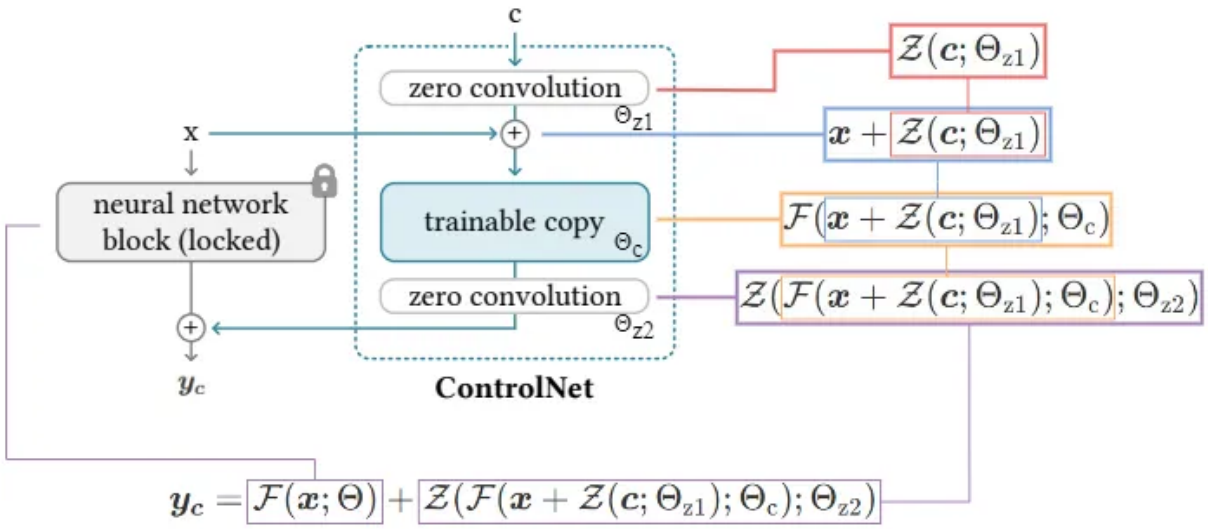


Figure 3.49: ControlNet feedforward [66]

Zero-convolution layers : In the initial training step, when the weight and bias of a zero convolution layer are set to zero, both of the $\mathcal{Z}(\cdot;\cdot)$ terms are evaluated to 0, and,

$$\mathbf{y}_c = \mathbf{y}$$

This ensures that detrimental noise cannot impact the hidden states of the neural network layers within the trainable copy at the commencement of training. Additionally, given that $\mathcal{Z}(\mathbf{c};\Theta_{z1}) = 0$, and the trainable copy is fed the input image x, it remains fully operational, preserving the capacities of the extensive pre-trained model. This enables it to function as a robust foundation for subsequent learning. The incorporation of zero convolutions serves to safeguard this foundational structure by eliminating random noise as gradients during the initial training phases [64].

3.8.2 ControlNet for stable diffusion

As a reminder, stable diffusion can be described as a UNet comprising an encoder, a central block, and a skip-connected decoder. It doesn't go through any gradient update and is thus "locked in". The encoding of text prompts involves the utilization of the CLIP text encoder, while the encoding of diffusion time steps incorporates a time encoder with positional encoding.

The ControlNet architecture is implemented across every encoder level of the UNet [64], as illustrated in Figure 3.50. Specifically, ControlNet is utilized to construct a trainable duplicate of both the encoding blocks and an intermediate block within stable diffusion. The encoding blocks exist at four different resolutions, each replicated three times. The resulting outputs are incorporated into both the skip connections and the central block of the UNet.

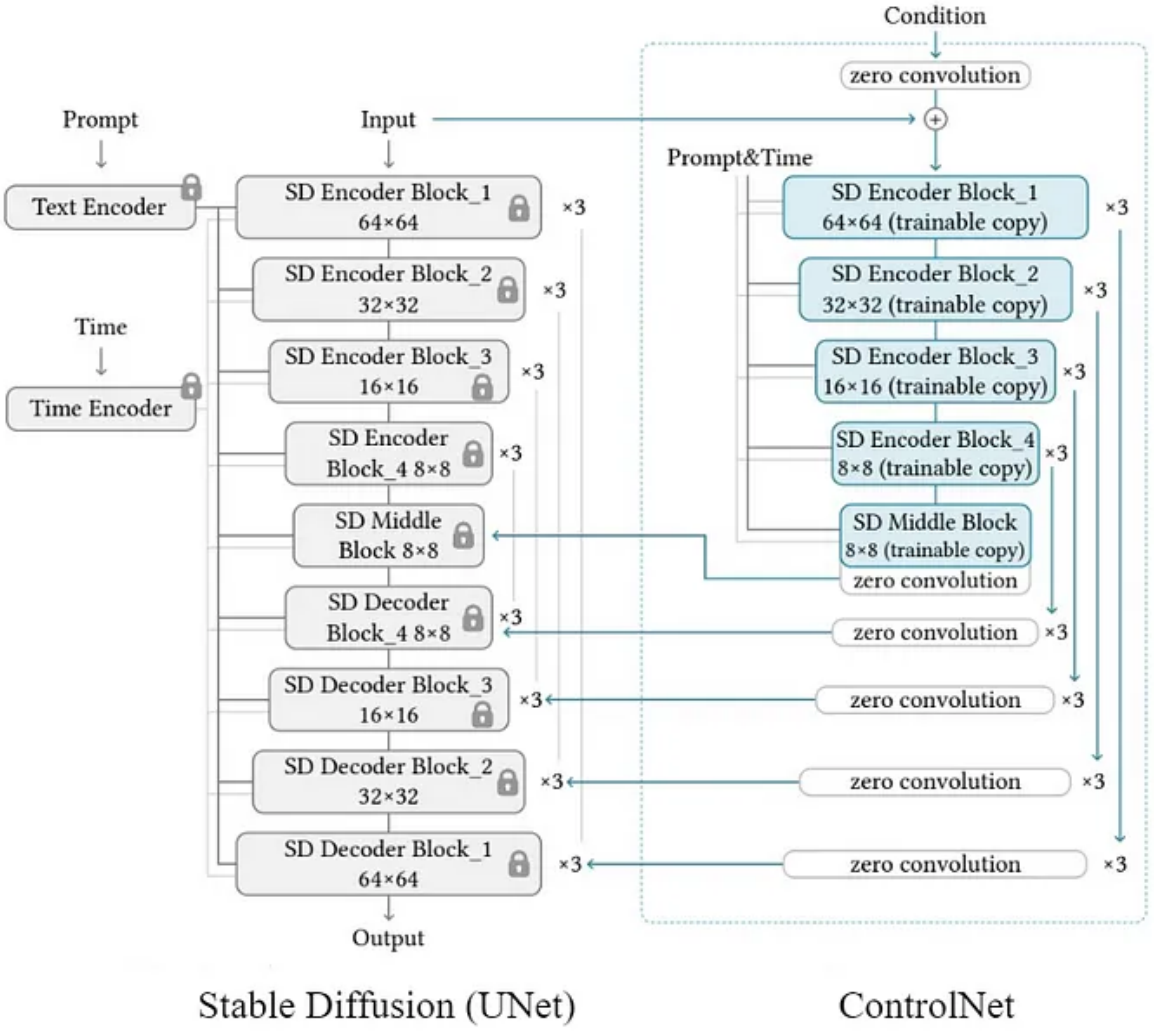


Figure 3.50: UNet and ControlNet architectures [66]

Stable diffusion employs latent images during training. Thus, to incorporate ControlNet into stable diffusion, the initial step involves transforming each input conditioning image (such as edges, poses, depth, etc.) from a pixel-sized input to a feature space vector that aligns with the dimensions of stable diffusion. Specifically, we utilize a small network $\mathcal{E}(\cdot)$ comprising four convolution layers to transform an image space condition $\mathbf{c}_i$ into a feature space conditioning vector $\mathbf{c}_f$, as follows [64, 66]:

$$\mathbf{c}_f = \mathcal{E}(\mathbf{c}_i)$$

The vector $\mathbf{c}_f$, representing the conditioning information, is sent to the ControlNet. Indeed, we can observe this integration of ControlNet in the modified reverse diffusion process at Figure 3.51 below.

3.8.3 Training

Throughout the training process, ControlNet duplicates the weights of neural network blocks into a "locked" copy, maintaining the model, and a "trainable" copy, which learns the given condition. As a result, as zero convolutions don't introduce noise to the network, the model should retain the capability to predict high-quality images.

Loss function : The loss function of ControlNet [66] resembles that of stable diffusion, but it incorporates the text condition $\tau_\theta(y)$ and latent condition $\mathbf{c}_f$ to enhance the consistency of the output with specified conditions :

$$L = \mathbb{E}_{t,\mathbf{z}_0,\varepsilon,y,\mathbf{c}_f}\left[\|\varepsilon - \varepsilon_\theta(\mathbf{z}_t, t, \tau_\theta(y), \mathbf{c}_f)\|^2\right]$$

As part of the training process, 50% of the text prompts $\tau_\theta(y)$ are randomly substituted with empty strings. Eliminating the prompts compels the encoder to depend more on the information contained in the control maps, this enables ControlNet to directly discern semantics in the input conditioning images maps, such as edges, poses, depth, and so on.

The model doesn't progressively grasp the control conditions but swiftly adapts to the input conditioning image, typically achieving this in fewer than 10,000 optimization steps. This occurrence is referred to as the "sudden convergence phenomenon" [64].

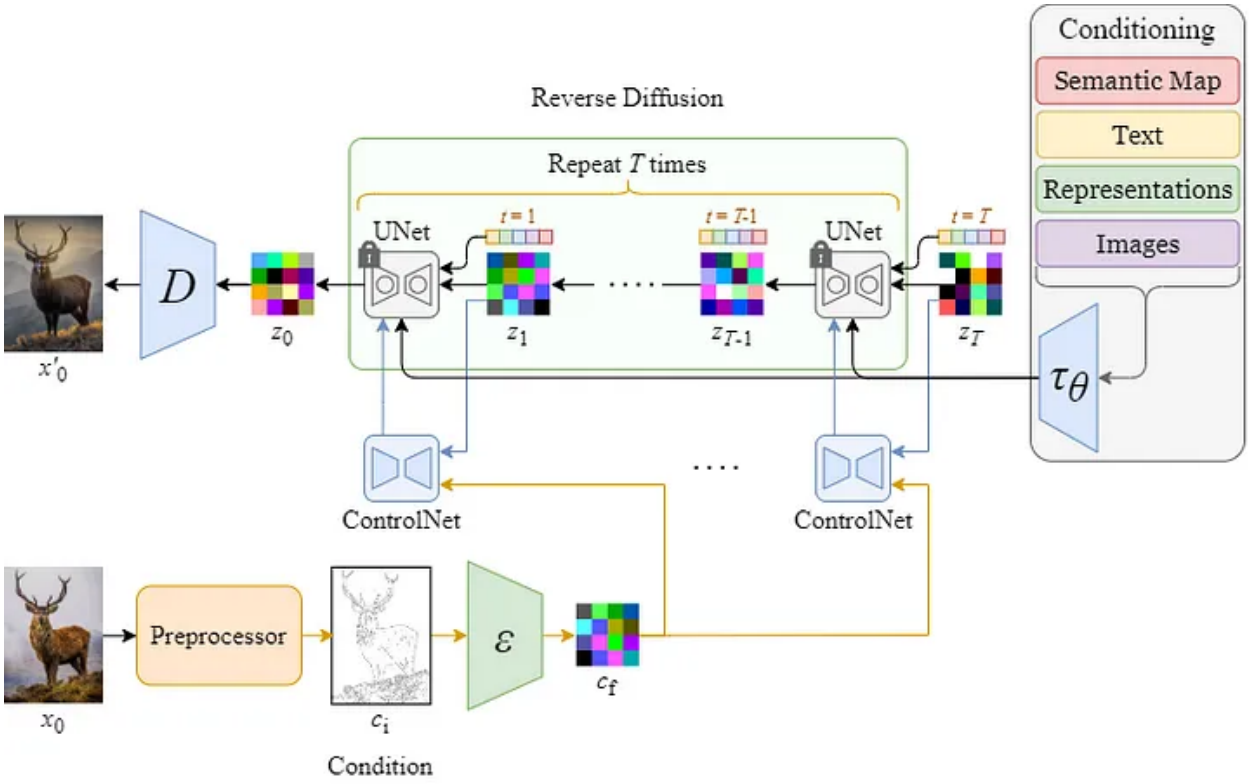


Figure 3.51: Reverse diffusion process of LDM with ControlNet [66]

3.8.4 Inference

We have the capability to influence the impact of additional ControlNet conditions on the denoising diffusion process through various means.

3.8.4.1 Classifier-free guidance resolution weighting

As said before, stable diffusion relies on a method known as Classifier-free guidance (CFG) to produce high-quality images. To reiterate, CFG is expressed as :

$$\bar{\varepsilon}_\theta(\mathbf{x}_t, t, y) = (\omega + 1)\varepsilon_\theta(\mathbf{x}_t, t, y) - \omega\varepsilon_\theta(\mathbf{x}_t, t)$$

When incorporating a conditioning image through ControlNet, it can be introduced to both the unconditional part and the conditional part, or exclusively to the conditional part. In challenging scenarios, such as when no prompt is provided, including the image in both the unconditional part and the conditional part eliminates CFG guidance entirely, while utilizing only the conditional part intensifies the guidance significantly.

The proposed solution in the document [64] involves initially adding the conditioning image to the conditional part, followed by multiplying a weight w_i at each connection between stable diffusion and ControlNet based on the resolution of each block, where $wi = \text{resolution}/h_i$, and h_i represents the size of the ith block. This approach reduces the guiding force of CFG, resulting in favorable outcomes, termed as CFG resolution weighting.

3.8.4.2 Composing multiple ControlNets

To incorporate various conditional images into a single instance of stable diffusion, one can simply combine the outputs of the respective ControlNets with the stable diffusion model. There is no need for extra weighting or linear interpolation in this compositional process [64].

3.8.5 Different types of conditioning

ControlNet is a versatile tool that enables the utilization of stable diffusion with various types of conditional inputs. We will go through several examples of conditional input types taking me as reference image and using the SD1.5 model and an upscaler.

Figure 3.52: Reference image to preprocess using ControlNet

- *Positive prompt :* (man smilling), happy face, short blond hair, beautiful brown eyes, black shirt, blue denim, silver watch, (intricate detail, best quality, masterpiece).
- *Negative prompt :* woman, (worst quality, low quality, letterboxed), ugly, disfigured, bad quality, bad eyes, bad nose, bad face, bad hands, text, watermark, blurred, blurry, tiling, deformed, mutated.

3.8.5.1 Canny edge

As depicted in the following illustration, ControlNet incorporates an extra input image and identifies its contours through the application of the Canny edge detector [65]. The resulting image, which includes the identified edges, is stored as a control map. This control map is subsequently introduced into the ControlNet model as supplementary conditioning along with the text prompt. Canny edge is valuable for preserving the original image's composition.

Figure 3.53: Canny edge image

Figure 3.54: Generated image

3.8.5.2 OpenPose

OpenPose [65] identifies key points on the human body, including the head, shoulders, and hands, enabling the replication of human poses while excluding specific details like clothing, hairstyles, and backgrounds.

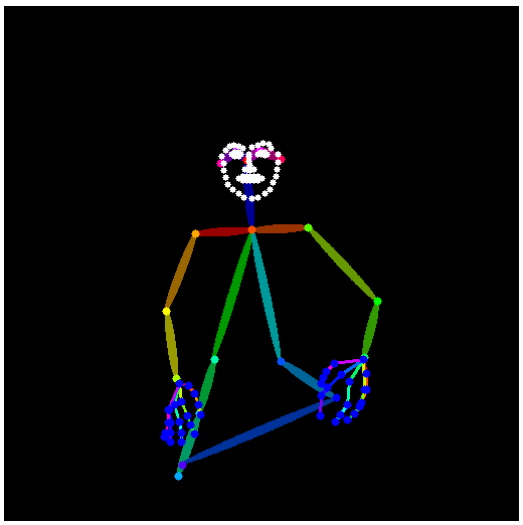


Figure 3.55: Open pose image

Figure 3.56: Generated image

3.8.5.3 Depth

The depth preprocessor estimates depth information based on the reference image.

Figure 3.57: Depth image

Figure 3.58: Generated image

3.8.5.4 Line art

Line art translates the image into an outlined representation, aiming to simplify it into a basic drawing.

Figure 3.59: Line art image

Figure 3.60: Generated image

3.8.5.5 Scribble

Scribble takes a drawn-by-hand image and process it to generate a great picture. It's an interesting tool for the application we build since it allows users to generate exactly what they want to have on their panel without having to be a great artist. Therefore, in what follows, we will go into more detail about this particular image formation process. Some examples can be found at Section 4.4.2.

3.8.5.6 Semantic segmentation

Segmentation preprocessors categorize objects present in the reference image.

Figure 3.61: Semantic segmentation image

Figure 3.62: Generated image

Chapter 4

ComfyUI

ComfyUI [30, 67] offers a user-friendly graphical user interface (GUI) for stable diffusion that features a highly adaptable, node-based design. It was created by Comfyanonymous in January 2023. This allows users to visually place and connect the constituent elements of the stable diffusion model with intuitive ease. Briefly, the interface enables the construction of image production workflows by connecting various blocks, referred to as nodes. These blocks include actions such as loading a control point model, inputting a prompt, specifying a sampler, and more. ComfyUI breaks down workflows into customizable elements, allowing users to easily design their own without the need for coding—everything is pre-implemented in the backend. Workflows can be saved as Json files, and an additional noteworthy feature is the support for SDXL templates, the latest templates for latent image generation [56].

ComfyUI's brief introduction sheds a little more light on the chapter that interests us at the moment. In what follows, we will outline the benefits of this GUI compared with another very popular one, and see how it relates to the technical components of stable diffusion through different workflows of interest to the application. Finally, tests and analyses will be carried out, using an SDXL model, to see how the generation of these different workflows behaves, and links will be established with the theoretical aspects previously mentioned.

4.1 Benefits of ComfyUI

It exists other well-known GUI as the de facto GUI for stable diffusion : *AUTOMATIC1111*. However, compared with *AUTOMATIC1111*, ComfyUI offers users several advantages [30, 67]. It is lightweight, ensuring fast performance, and boasts flexibility through extensive configurability. This implies that ComfyUI iterations can be enhanced in a manner not achievable with *AUTOMATIC1111* iterations. The pace at which the iterations occur is quicker with ComfyUI compared to *AUTOMATIC1111*. The transparency of the data flow is a notable benefit, as it remains visible throughout the process. So, by gaining proficiency in ComfyUI, you'll acquire an understanding of the true mechanics behind stable diffusion. Sharing is made easy, with each file serving as a reproducible workflow. ComfyUI stores all the details of the generation stream inside the generated PNG. ComfyUI is particularly advantageous for prototyping, allowing users to prototype using a graphic interface rather than relying on coding.

However, there are some drawbacks associated with ComfyUI. The interface may be inconsistent, as the placement of nodes can vary between workflows, requiring users to identify necessary adjustments. The level of detail provided may be excessive for average users who may not need insight

into the intricate workings beneath the surface—a key expectation when utilizing a GUI. Additionally, ComfyUI lacks inpainting tools, necessitating the use of an external program for inpainting tasks.

4.2 How does it work?

Every node performs a code by receiving inputs (values provided to the instructions) and producing outputs (values generated by the instructions) [67]. We can think of them as functions.

Users have the ability to :

- Generate new nodes
- Adjust node parameters, such as variables
- Establish connections between nodes by linking their inputs and outputs

For a clear understanding of how ComfyUI works, we will construct step-by-step a simple default text-to-image workflow made up of the most important nodes. Figure 4.1 shows what the ComfyUI interface looks like, as well as the final flow of the text-to-image process.

Hence, nodes have inputs on the left and outputs on the right, with parameters represented in the center of the block. The connections between nodes are depicted as wires linking outputs and inputs. Another interesting thing is that we know the compatibility between nodes when inputs and outputs have the same color [67].

When we select "Queue Prompt", the sequence progresses through the interconnected nodes in the specified order up to the generated image. It begins with loaders that lack inputs and only generate outputs [67].

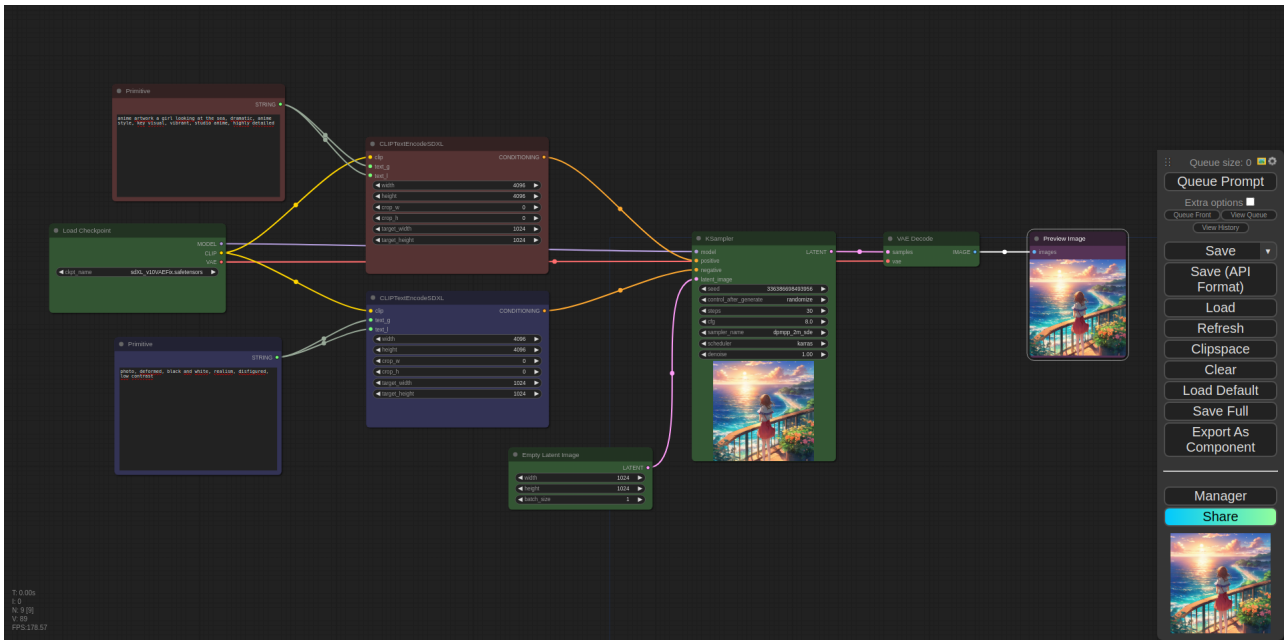


Figure 4.1: Complete text-to-image workflow

4.2.1 Load checkpoint

Before starting the workflow creation process, it is essential to choose a checkpoint model that captures our interest and will be utilized for generating images. Numerous checkpoint models can be found on CivitAI, HuggingFace, and TensorArt websites. Simply download the chosen model and place it in the designated directory.

Models, referred to as checkpoints, consist of predetermined diffusion weights established after undergoing stable diffusion training with specific images. These models can be customized for a specific style, genre, or subject, while also having generic models that can produce diverse images. The model's capabilities and the keywords it can identify are shaped by the images and related texts utilized in the training process. There are 2 types of models [68] :

- **Base models :** These represent the primary models employed by stable diffusion, derived from an extensive collection of images, serving as the foundation for the ability to create images. Given the substantial quantity of images needed for their creation, the variety of these templates is limited. The conventional ones are those released by the company responsible for developing stable diffusion : Stability AI.
- **Fine-tune models :** Fine-tuned models employ a standard machine learning approach called fine-tuning. This process entails taking a model that has been trained on a comprehensive dataset and providing additional training on a more specific dataset. As a result, the model becomes inclined to generate images resembling those from the supplementary training, all while preserving the adaptability of the original model. There are different methods for fine-tuning.

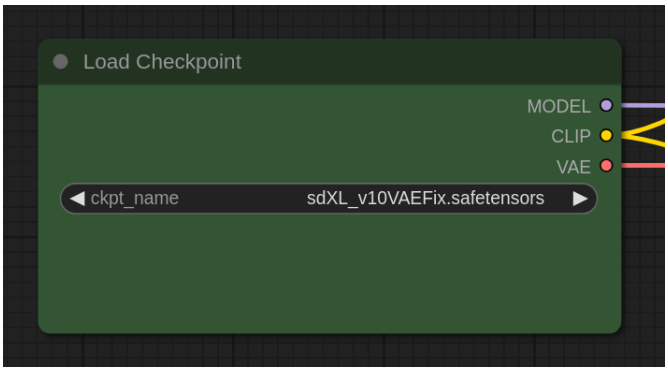


Figure 4.2: Checkpoint model loader

As shown on Figure 4.2 and explained, we utilize the Load checkpoint node for model selection. The stable diffusion models employed for image generation consist of three primary elements [30, 69] :

- **MODEL :** This corresponds to the latent space noise prediction model (UNet), facilitating the step-by-step execution of the diffusion process. This process involves denoising the latents, leading to the generation of the image.
- **CLIP :** This component handles the preprocessing of positive and negative prompts for the language model. It enables the encoding of text into a format comprehensible by UNet.
- **VAE :** The VAE is responsible for decoding images from latent space to pixel space. Additionally, it can encode a regular image from pixel space to latent space in cases where image-to-image encoding is required.

4.2.2 CLIP text encode SDXL

The output from the Load Checkpoint node in CLIP connects to the CLIP text encode SDXL nodes [30, 70, 71] (see Figure 4.3). These nodes retrieve positive and negative prompts and input them into a CLIP language model. Subsequently, this model transforms the text into embeddings. The resulting embeddings play a crucial role in directing the diffusion model towards the generation of specific images. Primitive nodes are only used to write prompts.

For image generation guidance, SDXL employs two text prompts : OpenCLIP ViT-bigG-14 and CLIP ViT-L. In ComfyUI, these input parameters are represented as follows :

- text_G corresponds to the text_encoder, CLIP_G (CLIPTextModel).
- text_L corresponds to text_encoder_2, CLIP_L (CLIPTextModelWithProjection).

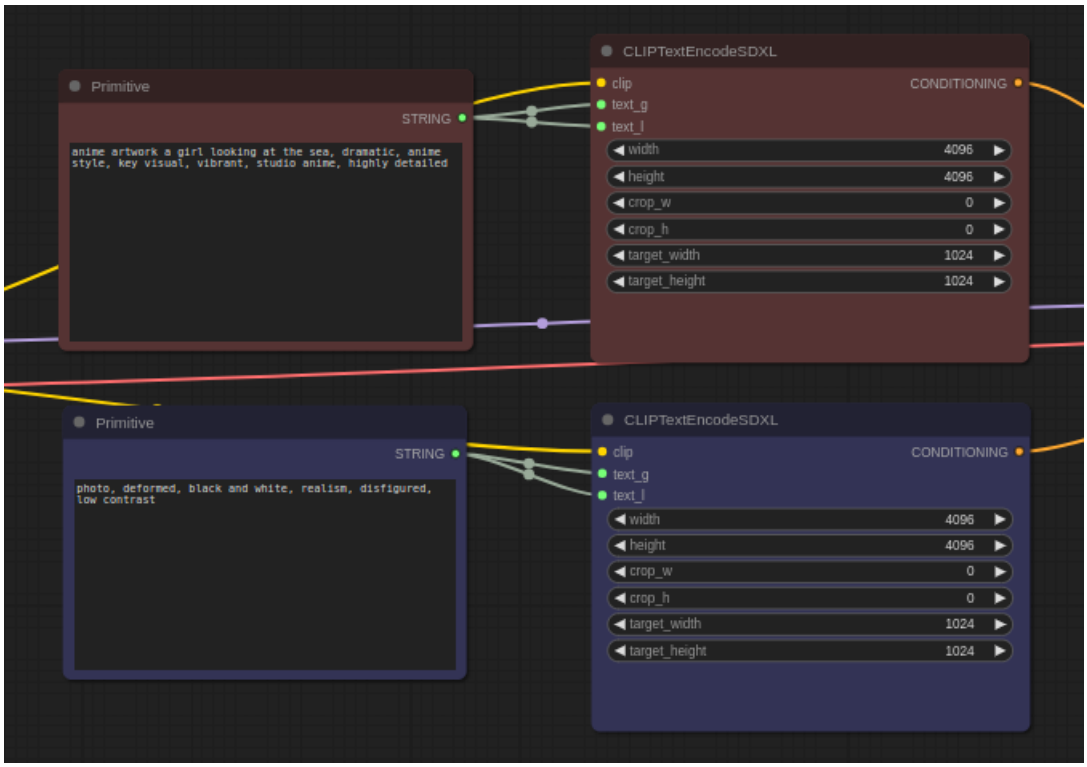


Figure 4.3: Clip text encode SDXL

4.2.3 Empty latent image

The process of converting text to images initiates with a randomly selected image in latent space. The dimensions of this latent image are directly proportional to the corresponding image in pixel space. This enables you to specify the image size in terms of height and width, along with the batch size, indicating the number of images generated in each iteration. The Empty latent image node [30, 72] generates a fresh set of vacant latent images. These latent images must undergo a process of introducing noise and subsequent denoising using a sampler node. This node is displayed in Figure 4.4.

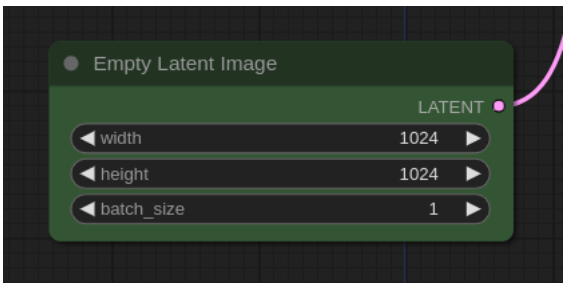


Figure 4.4: Empty latent image

4.2.4 KSampler

In stable diffusion, images are generated through a process known as sampling. In ComfyUI, this process occurs within the KSampler node (Figure 4.5).

KSampler [30, 73, 74] takes inputs of embeddings corresponding to positive and negative prompts from CLIP models, as well as an image in latent space (in our case, an empty one) that needs to be denoised. To achieve this, KSampler first introduces random noise to the latent image. Subsequently, it utilizes the MODEL output from the Load checkpoint, allowing the reverse diffusion process to take place, using the embeddings as a guide. At the end of this process, the denoised latent image is produced, hopefully meeting the desired specifications.

Several important parameters are integrated into the KSampler node and allow you to find the best way to obtain the desired final image :

- **Seed :** The initial randomness in the latent image and, consequently, the final image composition are governed by the random seed value.
- **Control_after_generate :** This parameter dictates how the seed should change after each generation. It could be a randomized value (randomize), an increase of 1 (increment), a decrease of 1 (decrement), or a constant value (fixed).
- **Steps :** The denoising process utilizes a specified number of steps. A higher step count results in fewer artifacts during the processing, which results in better quality. But each step requires calculation time. Attention that too many steps can ruin the image.
- **CFG :** The classifier's free guidance scale (CFG) is a parameter that regulates the extent to which the image generation process aligns with the text prompt. Increased scales prompt the image to closely adhere to the prompt, but overly high scaling can negatively impact image quality. Conversely, lower values result in greater creative deviation of the image from the text input. A study was carried out at the Section 4.4 to confirm the effects announced and to determine the most interesting values.
- **Sampler_name :** This configuration enables you to select the sampling algorithm responsible for executing denoising procedures. The algorithms can exhibit differences in speed, quality, and the variety of the image generation process.
- **Scheduler :** It governs the pace and advancement of the sampling process. It dictates the frequency and intervals at which the sampling process is iterated. Various Schedulers yield distinct impacts on both the quality and variety observed in the image generation process.

- **Denoise :** This parameter ranges from 0 to 1. A setting of 1 implies the input image is entirely substituted with noise, producing an output image unrelated to the original. Conversely, a setting of 0 signifies no noise addition, yielding an output identical to the input. Any value between 0 and 1 introduces some noise, allowing the output image to be influenced by the input but not entirely determined by it. This represents the amount to be removed to obtain the final image.

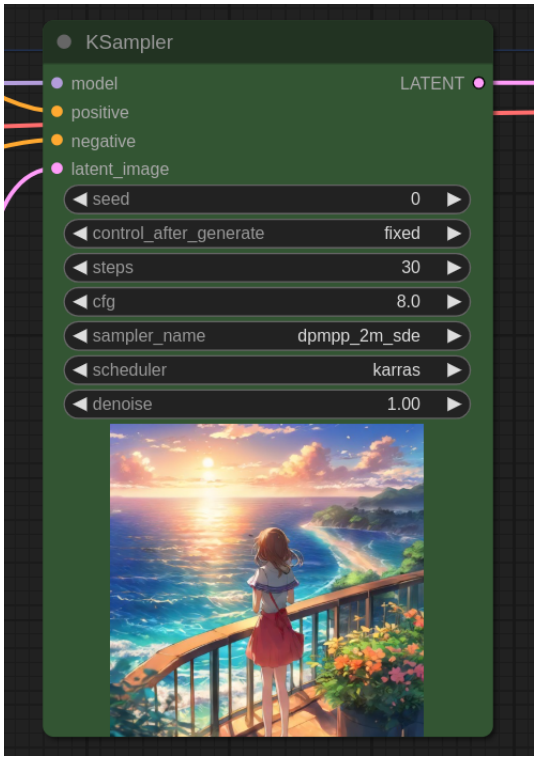


Figure 4.5: KSampler

4.2.5 VAE decode

The VAE Decode node [30], see Figure 4.6, accepts inputs such as the VAE from our checkpoint model or any other VAE, along with the denoised latent space image generated by our KSampler. The VAE is employed to convert an image from latent space to pixel space, specifically utilizing the decoder component of the autoencoder.

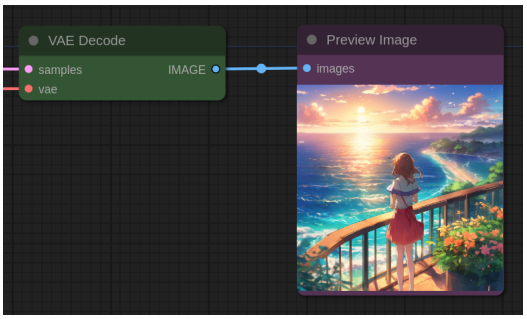


Figure 4.6: VAE Decode and Preview image

It sends the ultimate pixel image to the Preview image node for display purposes. Additionally, the Save image node is available for downloading the image.

4.3 Interesting custom nodes

A large number of nodes have been created and put online in open-source packages. These are added to ComfyUI. We won't go through many of them, just the ones that really interest us for this work. There are dozens of them, with a variety of objectives. The advantage of ComfyUI and its workflows made up of different nodes is that you can apply other tools to improve image generation that at first glance have nothing to do with the stable diffusion architecture. By using them appropriately, we can obtain or get closer to the desired result.

4.3.1 ComfyUI Manager

ComfyUI manager [2, 75] is an extension crafted to enhance the user experience of ComfyUI. It serves as a specialized node with the purpose of simplifying the installation, removal, and updating of extra custom nodes through the ComfyUI interface. Furthermore, it offers a hub function and user-friendly features to access various information within the ComfyUI system. The ComfyUI manager menu is shown below on Figure 4.7.

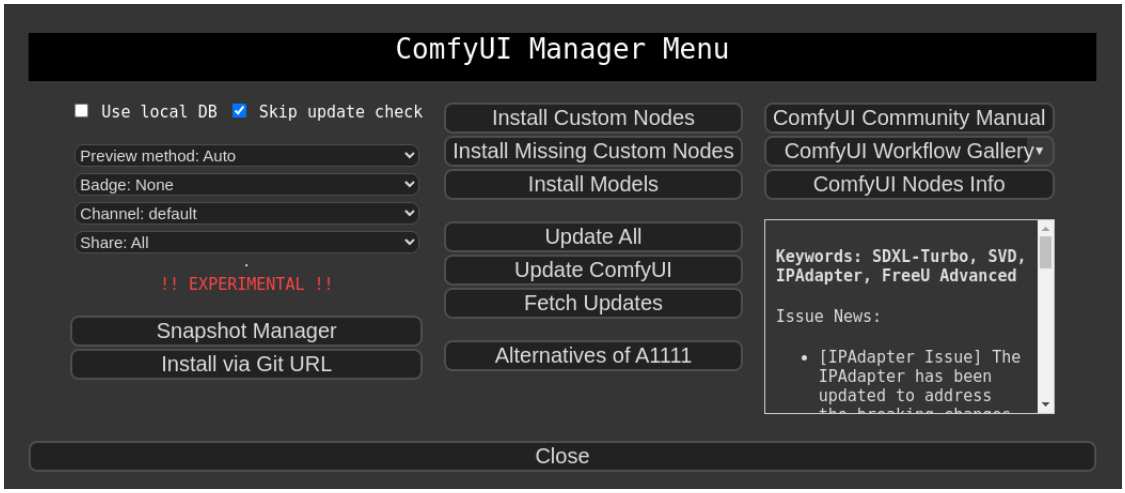


Figure 4.7: ComfyUI manager menu

4.3.2 ComfyUI Impact Pack

This collection of specialized nodes designed for ComfyUI simplifies the process of improving images using features such as detector, detailer, upscaler, pipe, and others.

Face Detailer : Included in this set is the FaceDetailer node [76], see Figure 4.10, which stands out as particularly intriguing. It merges two nodes – the Detector node for face detection and the Detailer node for restoring details. This combination simplifies the process of identifying faces and improving their quality. It proves valuable, especially in cases where stable diffusion struggles to generate high-quality facial images. Since this node shares functionality with KSampler's image enhancement, there is some overlap in options.

The detector identifies specific regions based on the model, and there are three types : SEGM, BBOX, or SAM [76].

- **BBOX :** BBOX, short for Bounding Box, enables the capture of detection zones in the form of rectangular regions. For instance, by employing the bbox/face_yolov8m.pt model, one can obtain masks corresponding to the rectangular areas of faces. These masks are accessible through BBOXs acquired via the UltralyticsDetectorProvider. The UltralyticsDetectorProvider node loads the Ultralytics detection models.
- **SEGM :** SEGM, which stands for Segmentation, captures detection zones as silhouettes. Using the segm/person_yolov8n-seg.pt model, for instance, one can obtain silhouette masks for human shapes. These masks can be acquired through SEGMs obtained via the UltralyticsDetectorProvider.

YOLO (You Only Look Once) [77, 78], a popular model for real-time object detection and image segmentation, has gained popularity for its speed and accuracy. It operates by dividing an image into a grid and predicting bounding boxes and class probabilities for each grid cell. YOLO is renowned for its real-time processing capabilities, as it processes the entire image in a single forward pass through the neural network. The network predicts multiple bounding boxes with associated class probabilities for each box using intersection over union. These predictions are refined using non-maximum suppression to filter out redundant or overlapping boxes, yielding a final set of accurate and distinct object detections. You can see a visualisation summarising what has just been explained in the Figure 4.8.

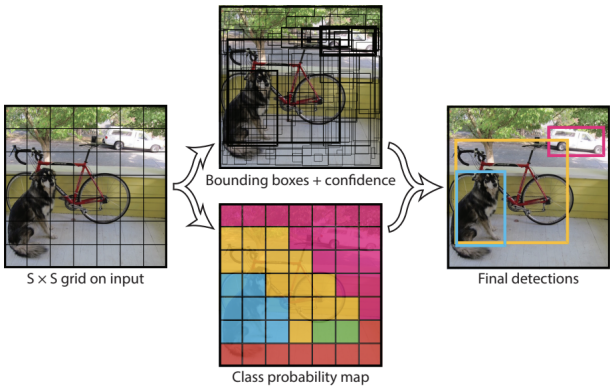


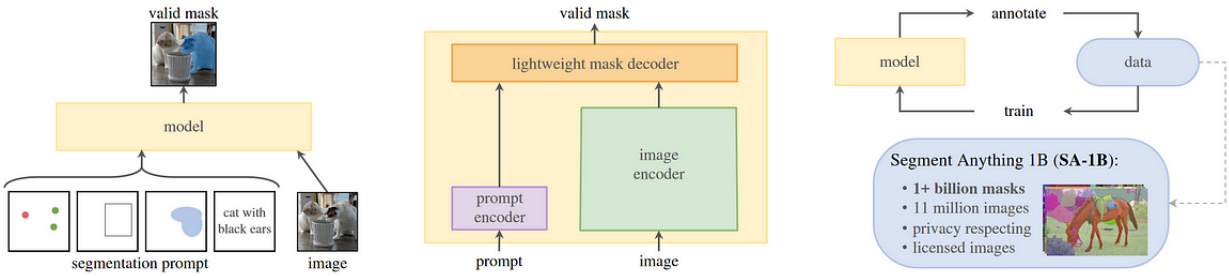
Figure 4.8: YOLO principle [78]

YOLOv8, developed by Ultralytics [79], represents the most recent iteration of YOLO, offering comprehensive support for various vision AI tasks. It leverages the accomplishments of its predecessors while introducing novel features and enhancements to boost overall performance, flexibility, and efficiency.

- **SAM :** SAM [80] produces silhouette masks using the Segment Anything technique. While it cannot be used independently, when used in conjunction with a BBOX model to specify the target for detection, it can generate finely detailed silhouette masks for the detected objects.

SAM is a promptable segmentation system designed for zero-shot generalization without requiring additional training on unfamiliar objects or images. Its architecture comprises an image encoder, a prompt encoder, and a lightweight mask decoder. The image encoder generates one-time image embeddings, while the prompt encoder embeds prompts such as points, boxes, or text in real-time. The lightweight mask decoder predicts segmentation masks based on embeddings from both encoders, utilizing prompt self-attention and cross-attention. SAM updates model weights using annotated

masks, allowing for continual learning and flexibility. The model is trained on the Segment Anything 1 Billion Mask (SA-1B) dataset, the largest labeled segmentation dataset to date, known for its diversity, size, and high-quality annotations [81, 82]. You can get a better idea of what was just said by looking at Figure 4.9.



(a) **Task**: promptable segmentation (b) **Model**: Segment Anything Model (**SAM**) (c) **Data**: data engine (top) & dataset (bottom)

Figure 4.9: SAM principle [83]

Figure 4.10: Face detailer

An example of its use with BBOX is shown below in figure 4.13. We can clearly see that the man's

face in the image on the right is of high quality, unlike the image on the left. We will therefore use and test this node later in our workflows. It solves some interesting problems, as you can see from this example.

Figure 4.11: Image obtained after one generation

Figure 4.12: Image obtained after applying Face detailer

Figure 4.13: Example of the effectiveness of the Face detailer

4.4 SDXL workflows - tests

Now that we have understood the tool, i.e. we have seen and understood what generative AI is, how stable diffusion works, the SDXL models, the ComfyUI interface, etc., we can now create streams in ComfyUI using SDXL. We'll then test these workflows and try to understand how images are generated using certain parameters. We'll keep the workflows and parameters that give the best image quality, depending on what we want to do. The 3 workflows we'll be testing are : text-to-image, text-to-image with Scribble and inpainting. We will apply these tests to character generation and scene generation. The influence of the Face detailer will also be tested.

The goal is to generate images of a comic strip by taking as an example some panels from an existing comic strip. The choice is : "To the Stars and Back - episode 1" by Peglo, found on webtoon.com. In this way, we will recreate the idea of the images, trying to ensure that the images we generate resemble the comic strip and that the images have as much consistency and quality as possible. In addition, other tests are carried out. The tests set out below will be used to demonstrate the consistency that can be achieved with prompts alone. To do this, go to Section 5.1 in the next chapter.

Figure 4.14: Panel 1 [84]

Figure 4.15: Panel 2 [84]

Figure 4.16: Panel 3 [84]

Figure 4.17: Panel 4 [84]

4.4.1 Text-to-image workflow

The easiest workflow to create is the one that was precisely explained in the previous section (back to Section 4.2), the text-to-image workflow. In this part, we will explain the choice of certain parameters and analyze the rendering of images, in order to see what is good and what has to be improved.

Now, we're now going to generate images corresponding to the chosen panels and inspect them, to see what can already be done and what the challenges are in terms of consistency and image quality. We'll start by generating the panels for the scene, then move on to generating the characters.

We used the following model, *dynavisionXLAllInOneStylized*, which is a 3D model, and, the same negative prompt for all generations : bad face, ugly, deformed face, bad quality, beard, earring. The "denoise" parameter of the KSampler is set to 1 because we need to generate a completely new image. The image to be denoised must consist of 100% random noisy image. The other parameters chosen are "sampler"=dpmpp_2m, "scheduler"=karras and "steps"=20. According to the literature [85], this is a good and rather popular choice when you want fast convergence, decent generation quality and something new. There are plenty of other samplers and schedulers out there, with slightly different characteristics but just as good.

Regarding the "cfg" parameter, as mentioned earlier in Section 3.6, we will conduct a brief investigation to observe its impact on the images. Here below are the different prompt scenarios devised for this test and it is important to note that the same seed was used for each CFG scale tested from a particular prompt.

Model : sdXL_v10VAEFix.safetensors

- **Prompt 1 :**
  - *Positive prompt :* bike resting against a wooden fence in front of a heavenly beach, (boat in the sea:1.2), sun shining, intricate detail, masterpiece, high quality.
  - *Negative prompt :* houses, ugly, deformed, blurry, text, painting, anime, cartoon.
- **Prompt 2 :**
  - *Positive prompt :* 4x4 car parked in the savannah, group of lions, lionesses posed near a baobab, sun shining, intricate detail, masterpiece, high quality.
  - *Negative prompt :* ugly, deformed, blurry, text, painting, anime, cartoon.
- **Prompt 3 :**
  - *Positive prompt :* bustling futuristic cityscape with advanced technology, soaring skyscrapers, and flying vehicles intricate detail, masterpiece, high quality.
  - *Negative prompt :* ugly, deformed, blurry, text, painting, anime, cartoon.
- **Prompt 4 :**
  - *Positive prompt :* anime, young brunette girl going down a slide in a green playground, close-up face, happy, smiling, rainy weather, yellow jacket, she raises her arms to the sky, intricate detail, masterpiece, high quality.
  - *Negative prompt :* ugly, deformed, disfigured, extra limbs, blurry, text.

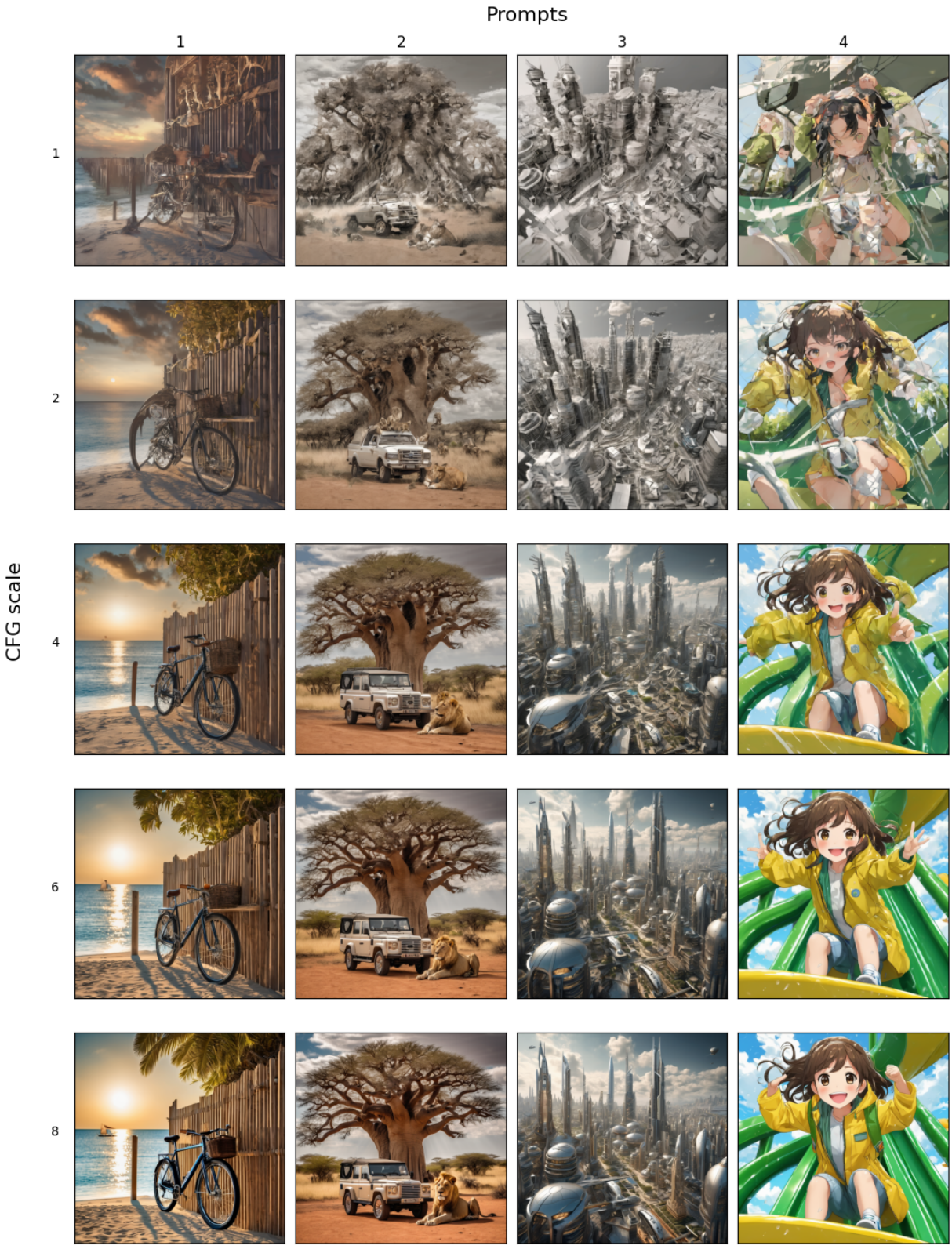


Figure 4.18: CFG test - First part

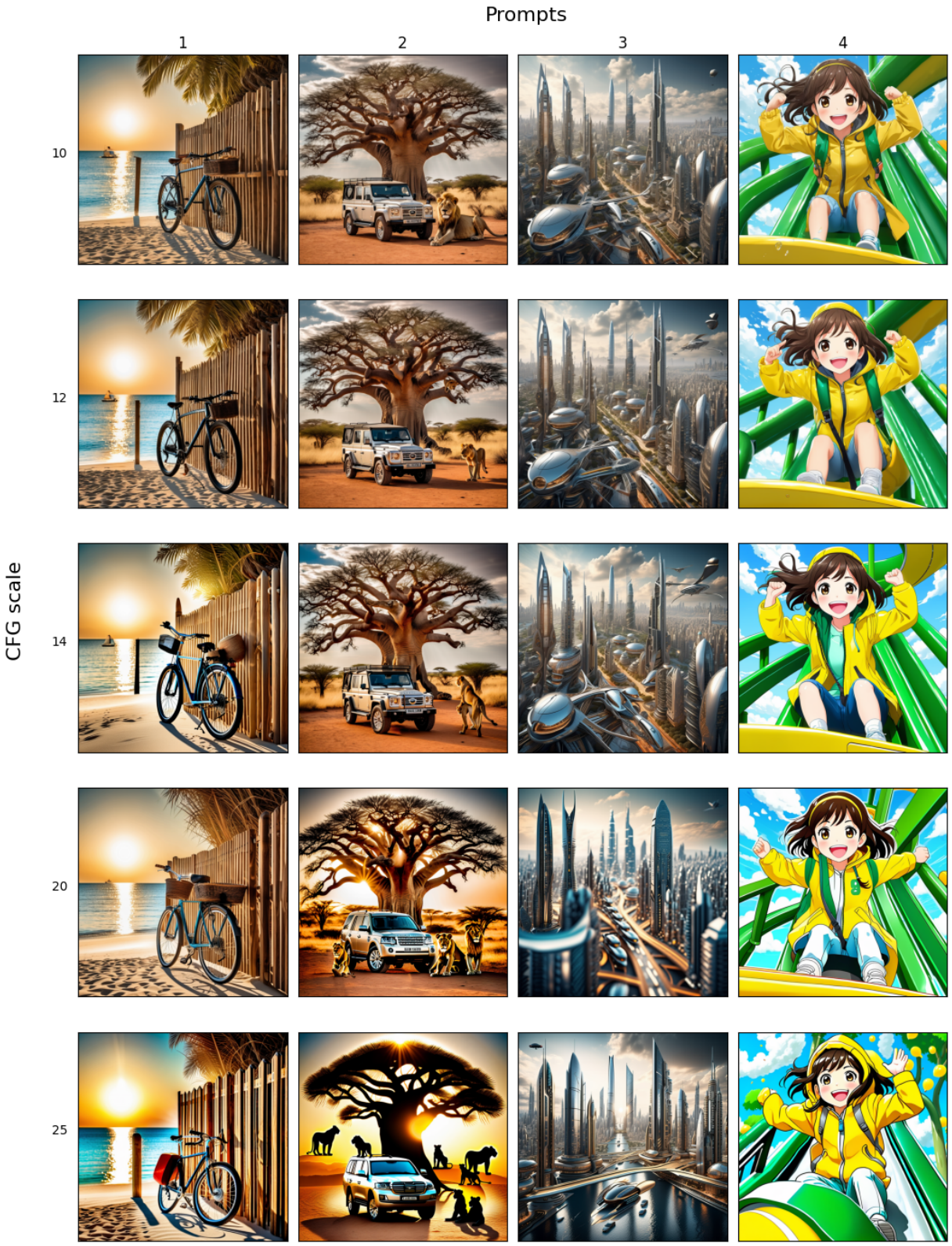


Figure 4.19: CFG test - Second part

As a reminder, we previously mentioned that the CFG scale serves as a parameter to regulate how closely the stable diffusion model aligns with the text prompt. A higher CFG value, ω regarding to the theoretical part in Section 3.6, implies a stronger adherence to the prompt, and vice versa. The tests will confirm this hypothesis. It is logical in view of the equation which combines the estimates for the conditional and unconditional scores.

Upon observing Figure 4.18, it becomes evident that for lower CFG values (1, 2, 4), the model deviates from the prompt and we have a poor image quality. Conversely, as the CFG value increases, a noticeable enhancement occurs. Examining Figures 4.19, it's apparent that prompt adherence improves progressively. However, there's a simultaneous rise in color saturation and contrast with increasing CFG values, especially beyond 14. Beyond a certain CFG threshold, roughly 20-25, the image output starts to lose detail, becoming blurrier.

Consequently, outputs with a more creative and artistic appearance tend to emerge within the CFG range of 6 to 12. Although using a scale up to 14, and possibly a bit more, still yields results with minimal artifacts. Hence, selecting a CFG value of 8 appears to be a well-balanced choice based on our comprehensive testing.

Now that we have given and discussed the important parameters of the KSampler, we can first examine and analyse what we obtain by generating images of scenes using only the text-image method. Later, in Section 4.4.2, we will compare these results with those obtained with the help of a drawing.

- *Positive prompt :* ((cartoon style)), intersection of 2 small streets, buildings stuck together, crosswalk with a blond woman walking on it, small store, overhead electric cables carried by a pole and linking the houses, one tree is behind an electric pole, yellowish image, yellow-orange sky, some clouds, high quality.

Figure 4.20: Text-to-image - Panel 1

- *Positive prompt :* ((cartoon style)), a large yellow apartment building with balconies, white clouds in the yellow-orange sky, yellow-orange image, one tree at the bottom left of the building, high quality.

Figure 4.21: Text-to-image - Panel 2

We can already see here that it's difficult with a single prompt to capture all the elements you want to incorporate into the scene and position them where you want them to be positioned.

Another difficulty, as can be seen in Figure 4.20, the faces don't look great in a big scene without a Face detailer. The key to creating webtoon-style comic panels, particularly in manga, anime, and cartoons, is to focus on simplicity. It's advisable to generate straightforward scenes that don't require numerous elements. Unlike other genres, there's usually no necessity for elaborate, complex scenes in this style of comics. Here, we need to concentrate above all on the characters.

The second series of images, Figure 4.21, requires fewer elements and already looks more like what we want. Overall, the quality of the images is acceptable and more or less resembles what we want to achieve, although a few elements are missing.

To achieve a cartoon style, it's important to specify in the prompt : "cartoon style". And, to make sure you get the cartoon effect you can use brackets around these keywords. This will give more weight to the words. A full explanation of this principle can be found in Section 5.1.

Now let's do the same thing for character generation.

- *Positive prompt :* ((cartoon style)), young adult man, 25, is reading a book sitting on the floor against a wall in his bedroom where there is a desk with a computer leaning, black color eyes, black color eyebrows, medium-length black hair, rectangular black glasses, wearing a solid-colored cyan t-shirt, solid-colored black pants and white socks, legs crossed, high quality.

Figure 4.22: Text-to-image - Panel 3

- *Positive prompt :* ((cartoon style)), young adult man, 25, concentrated face, black eyes, black eyebrows, medium-length black hair, rectangular black glasses wearing a solid cyan t-shirt, is reading a book held in one hand and scratching the back of his head with his other hand, beige background, high quality.

Figure 4.23: Text-to-image - Panel 4 - First order of the prompt

- *Positive prompt :* ((cartoon style)), young adult man, 25, is reading a book held in one hand and scratching the back of his head with his other hand, beige background, black eyes, black eyebrows, concentrated face, medium-length black hair, rectangular black glasses wearing a solid cyan t-shirt, high quality.

Figure 4.24: Text-to-image - Panel 4 - Second order of the prompt

Here are the analyses that can be made of images generated for characters using text-to-image. First of all, when comparing the images in Figure 4.22 with those in Figures 4.23 and 4.24, it should be noted that the quality of the faces is not as good when they are not close-up faces. We can also see that the haircuts, although sometimes similar, vary slightly from one image to another. It should be noted that it was difficult to get the character to look at the book. Also, as the colour of the book was not specified, the book took one of the colours from the prompt that was associated with something else. Nevertheless, it's sometimes complicated to use several colours in the same image. Even if they are specified for each element of the image, they can sometimes blend together. In the same way, when you indicate that the character is reading a book, there are lots of books in the room. Once again, we see that it's complicated to capture everything in the prompt. Finally, it's worth noting that it was difficult to get the character to look at the book or to make sure that he was in the expected pose, i.e. with his hand on the back of his neck. In fact, especially when it comes to the pose, we can see by comparing Figures 4.23 and 4.24 that the order of the words in the prompt matters.

This means that the way in which the prompt is written is very important and that by using only the prompt you can achieve a more or less coherent character with a very good quality image. These tests also showed that certain words give better results and that the order of the words in the prompt is important. We'll see why later in a chapter reserved for the prompt (go to Section 5.1). So these tests are already encouraging in terms of consistency and quality.

With Face detailer : We can now generate and analyse images of characters and then apply the Face detailer to obtain a better quality face. We use for the positive prompt, "beautiful young man face, 25, best quality, cartoon style", and, for the negative prompt, "bad face, ugly face, disfigured". As concern the parameters, a good choice is to use "denoise"=0.45, "sampler"=dpmpp_2m, "scheduler"=karras, "steps"=20 and "cfg"=8. The parameters are the same as before, except for the "denoise". The choice of a value of 0.45, around 0.5, will be discussed below following the tests carried out.

- *Positive prompt :* ((cartoon style)), young adult man, 25, is reading a book sitting on the floor against a wall in his bedroom where there is a desk with a computer leaning, black color eyes, black color eyebrows, medium-length black hair, rectangular black glasses, wearing a solid-colored cyan t-shirt, solid-colored black pants and white socks, legs crossed, high quality.

Figure 4.25: Text-to-image with Face detailer - Panel 4 - "denoise"=0.45

- *Positive prompt :* ((cartoon style)), young adult man, 25, is reading a book sitting on the floor against a wall in his bedroom where there is a desk with a computer leaning, black color eyes, black color eyebrows, medium-length black hair, rectangular black glasses, wearing a solid-colored cyan t-shirt, solid-colored black pants and white socks, legs crossed, high quality.

Figure 4.26: Text-to-image with Face detailer - Panel 4 - "denoise"=0.9

We can see that it works quite well, but it's difficult to correct the character's face if you want them to look at their book. Indeed, the Face detailer only regenerates part of the image and no longer has the context of the book you need to look at. Moreover, we can note that sometimes it is important to specify the character's gender in the Face detailer prompt so as not to change the character's gender. This depends on the models, which are often based on a larger number of female characters.

Finally, here it is important to note that the "denoise" parameter plays a vital role. When the "denoise" is high, see Figure 4.26, the original head is no longer respected and the face changes a lot. A lower denoise gives much better results in terms of consistency, see Figure 4.25. This is logical because, as we explained earlier, the more noise we decide to add, the more we're going to ask the model to generate an image that has no basis, no pre-generated image, which means that the final image will be completely different from the base image we were expecting. The "denoise" value depends on the image, but generally around 0.5 is a good compromise for flexibility and consistency. This could be a parameter to be supplied to the user of the application.

4.4.2 ControlNet (Scribble) worklfow

An interesting opportunity to integrate into the comic creator is the ability for the user to generate the image they want based on a sketch. This sketch should only give the direction to be followed during the denoising process. The drawing does not need to be of high quality for the tool to be accessible to everyone. So, below, Figure 4.27 shows the workflow, the assembly of nodes, chosen to achieve this.

Figure 4.27: Scribble SDXL workflow

Let's take a closer look at the part added to the text-to-image workflow. This part of the workflow, shown in Figure 4.28, is the one that enables ControlNet to be used. You can see that several nodes are involved. We will briefly discuss them.

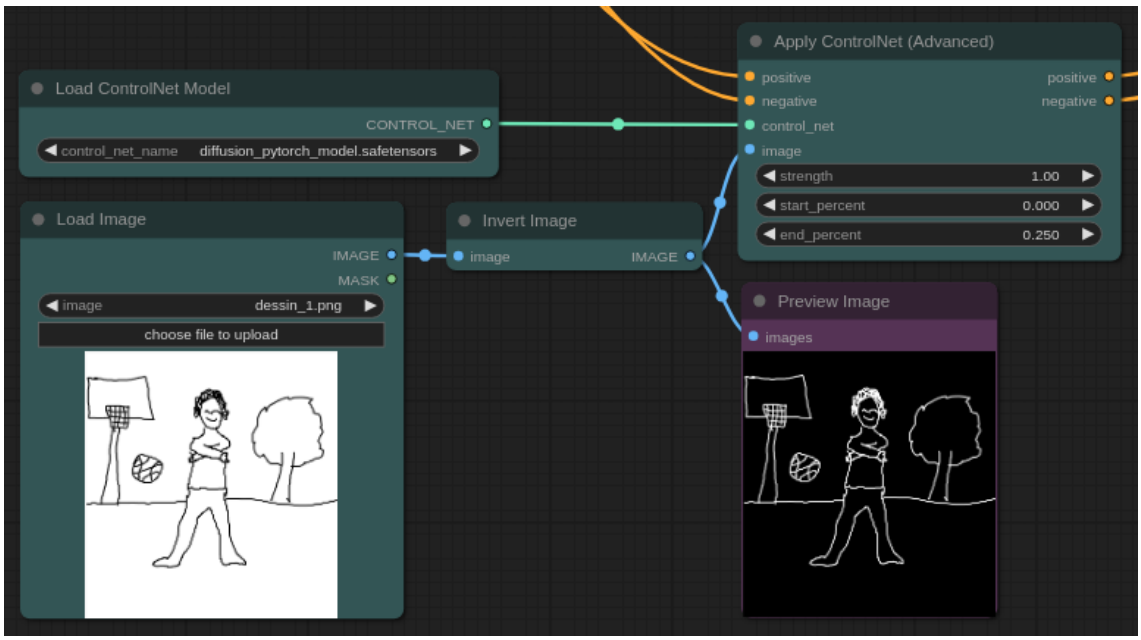


Figure 4.28: ControlNet workflow

- **Load image :** This node loads an image [86]. It therefore takes as input the name of the image to be used, in our case a black-on-white drawing, and returns the image in pixels and the mask, i.e. the alpha channel of the image.
- **Invert image :** This is used to invert the colours of an image [87]. It therefore takes as input the pixelated image to be inverted and returns the inverted pixelated image. So the black-on-white design becomes a white-on-black design. This is beneficial because the model we are using requires a hand-drawn image on a black background with white outlines.
- **Load ControlNet model :** This node is used to load the chosen ControlNet model [88]. This model is used to give visual indications to the diffusion model. As input, this node takes the name of the ControlNet model and returns the loaded model. In our case, the chosen model is TencentARC/t2i-adapter-sketch-sdxl-1.0[1] trained with PidiNet edge detection. This specific control point offers conditioning directly on the sketch for the SDXL control point. This collaboration is a joint effort involving Tencent ARC and HuggingFace.
- **Apply ControlNet (advanced) :** Ultimately, the primary node serves as the visual reference for the diffusion model [90], any model. It receives input from the positive and negative prompts, the ControlNet model guiding the diffusion model with specific image data, and the image serving as a visual guide. The output includes the conditioning incorporating both the ControlNet model and the visual guide. Additionally, three parameters require careful selection [91] : "strength", "start_percent," and "end_percent". They ascertain the timing and manner in which control is exerted over the generation process.
  - **strength :** The potency of the ControlNet network is defined by the strength parameter, indicating the extent of its impact. A greater weight ensures a closer resemblance between the generated image and the original reference image. The ControlNet outputs are multiplied by the "strength" factor, amplifying their influence during the integration with the UNet of the stable diffusion model. This ensures heightened attention to the contributions of ControlNet in the merging process.

[1]Similar to ControlNet, the T2I adapter [89] functions as a network, offering supplementary conditioning for stable diffusion.

- **start_percent :** This corresponds to the proportion of the generative process at which the impact of ControlNet initiates.
- **end_percent :** This marks the phase in the generation process where the ControlNet ceases to exert influence. To illustrate, suppose we opt to generate an image over 20 steps. By setting "start_percent" to 0 and "end_percent" to 0.75, the generation initiates with the ControlNet model and concludes after 15 steps, allowing the model some creative latitude.

To understand how the scribble and its parameters work and to see how it reacts, we're going to generate several images based on a hand-drawn picture. The same model, the same prompts and the same seed are used for each generation, so that we can compare and contrast the effects. All the parameters remain the same apart from the ControlNet parameters, which we will vary. We only fix the "start_percent" parameter to 0. Below you'll find the model and prompts used, as well as the reference image, which is my drawing on Figure 4.29.

Model : dynavisionXLAllInOneStylized

- *Positive Prompt :* cartoon style, man standing up arms crossed, white skin, blue eyes, black hairs, red t-shirt, white short, white socks, black baskets, happy, smiling, basketball hoop, bright sunshine, beautiful face, quality.
- *Negative Prompt :* ugly, disfigured, bad quality, bad eyes, bad nose, bad face, bad hands, text, watermark, blurred.
- *Positive Prompt of the Face detailer :* beautiful man face, blue eyes, cartoon style.
- *Negative Prompt of the Face detailer :* bad face, ugly, disfigured.

Figure 4.29: Hand-made drawing

Consequently, in order to decide on a relevant way to configure the sketch tool that will be proposed to the user, we will observe the influence of the parameters. A comparison grid will allow us to visualize the images generated with different combinations of parameters (see Figure 4.30). The goal is to decide on a slider to offer to the user so that the generated image resembles more or less their original sketch, while sparing them all the technical aspects because the creation tool should be accessible to as many people as possible.

Figure 4.30: Visualisation to determine the influence of Scribble ControlNet parameters

Analyzing this, we can extract some interesting information. Indeed, we see that the "strength" parameter plays a rather important role, having the expected effect. The higher its value, the more significance the sketch's weight carries. It can be observed that only from a "strength" of 0.75 does the model comprehend the sketch in its entirety. However, when the "strength" parameter is set to 2, the sketch is overly emphasized, preventing the generation of a high-quality image. This hinders the model's ability to be creative. As for the "end_percent" parameter, there is less noticeable difference between the values. However, upon closer inspection, some quality differences among the images become apparent. Whether for the "strength" or "end_percent" parameter, when set to 0, the sketch is not considered.

In conclusion, one can imagine introducing into the image generation tool a slider that would operate diagonally in the table, i.e., with similar values for both parameters ranging from 0 to 1.

We can now return to our tests for the comic book "To the Stars and Back - episode 1". However, we're now going to test the generation of images from a hand-drawn sketch. We'll start by generating scenes, as we did earlier for the tests with the text-to-image workflow. Below are the 2 drawings I made. The one based on panel 1 is in Figure 4.31 and the one based on panel 2 is in Figure 4.32. We will analyse the images generated and compare them with those obtained using text-to-image.

Figure 4.31: Drawing from panel 1

Figure 4.32: Drawing from panel 2

- *Positive prompt :* ((cartoon style)), intersection of 2 small streets, buildings stuck together, crosswalk with a blond woman walking on it, small store, overhead electric cables carried by a pole and linking the houses, one tree is behind an electric pole, yellowish image, yellow-orange sky, some clouds, high quality.

Figure 4.33: Scribble - Panel 1 with a strength of 1

- *Positive prompt :* ((cartoon style)), a large yellow apartment building with balconies, white clouds in the yellow-orange sky, yellow-orange image, one tree at the bottom left of the building, high quality.

Figure 4.34: Scribble - Panel 2 with a strength of 0.8

If we analyse the generations in Figures 4.33 and 4.34, we can see that the images are much closer to what we want to achieve than they would have been without the scribble, i.e. only with the text-to-image workflow (see Figures 4.20 and 4.21). To get a better match between the drawing and the generated image, a high level of strength is required. In this case, the elements are positioned in the right places. However, the characters are sometimes less well realised and it is difficult to

make out which character is which.[2] What's more, it's always complicated to bring together all the characteristics of the prompt. All in all, scribbling is very useful and allows you to specify what you want. The key is to adjust the ControlNet parameters using the tests described above.

Next, we'll examine the creation of character images derived from hand-drawn sketches. I've provided illustrations I crafted, with the depiction from panel 1 shown in Figure 4.35 and the one from panel 2 in Figure 4.36. We'll assess the generated images and draw comparisons with those produced using the text-image approach.

Figure 4.35: Drawing from panel 3

Figure 4.36: Drawing from panel 4

- *Positive prompt :* ((cartoon style)), young adult man, 25, is reading a book sitting on the floor against a wall in his bedroom where there is a desk with a computer leaning, black color eyes, black color eyebrows, medium-length black hair, rectangular black glasses, wearing a solid-colored cyan t-shirt, solid-colored black pants and white socks, legs crossed, high quality.

Figure 4.37: Scribble - Panel 3

[2] As a reminder, the multiplicity of characters in the images is not the subject of the research in this work and may be the subject of a future work.

- *Positive prompt :* ((cartoon style)), young adult man, 25, is reading a book held in one hand and scratching the back of his head with his other hand, beige background, black eyes, black eyebrows, concentrated face, medium-length black hair, rectangular black glasses wearing a solid cyan t-shirt, high quality.

Figure 4.38: Scribble - Panel 4 - with a strength of 1

- *Positive prompt :* ((cartoon style)), young adult man, 25, is reading a book held in one hand and scratching the back of his head with his other hand, beige background, black eyes, black eyebrows, concentrated face, medium-length black hair, rectangular black glasses wearing a solid cyan t-shirt, high quality.

Figure 4.39: Scribble - Panel 4 - with a strength of 0.4

Using the scribble, we can see that the model is trying to generate images in line with the sketch. However, he has difficulty understanding the depth of the drawing (see Figure 4.37), especially when the quality is poor. We can also see that with a high strength the character looks younger and older, and respects the prompt, see Figure 4.38. In fact, the pose is completely respected, as is the slightly odd shape of the head. With a lower strength, as in Figure 4.39, the pose is also taken into account, while leaving the model free to be creative. These results confirm our previous tests.

4.4.3 Inpainting workflow

A great feature to possibly integrate into the application is the ability for the user to regenerate selected parts of an already generated image. This would allow improvement by adding or removing components in the image. The user only needs to create a mask on the part of the image they want to modify. Below, Figure 4.40 illustrates the chosen workflow to achieve this.

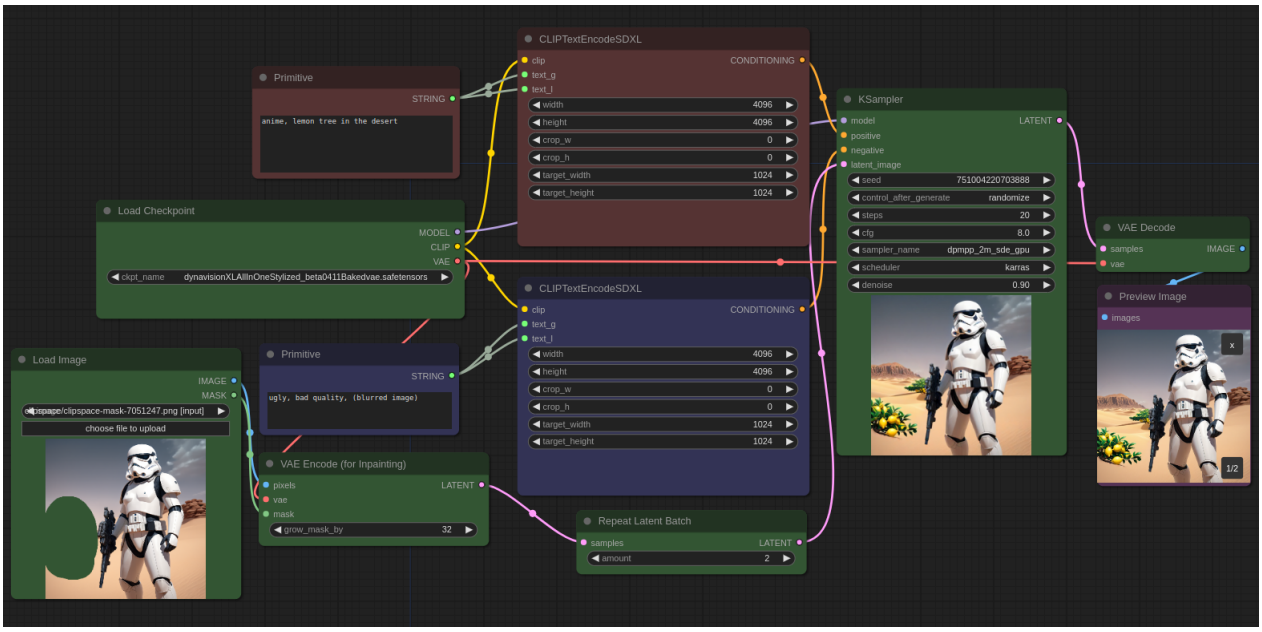


Figure 4.40: Inpainting SDXL workflow

Let's take a closer look at the part added to the text-image workflow. This portion of the workflow, illustrated in Figure 4.28, is the one that enables the use of inpainting. You can see that several nodes are involved. We will discuss them briefly.

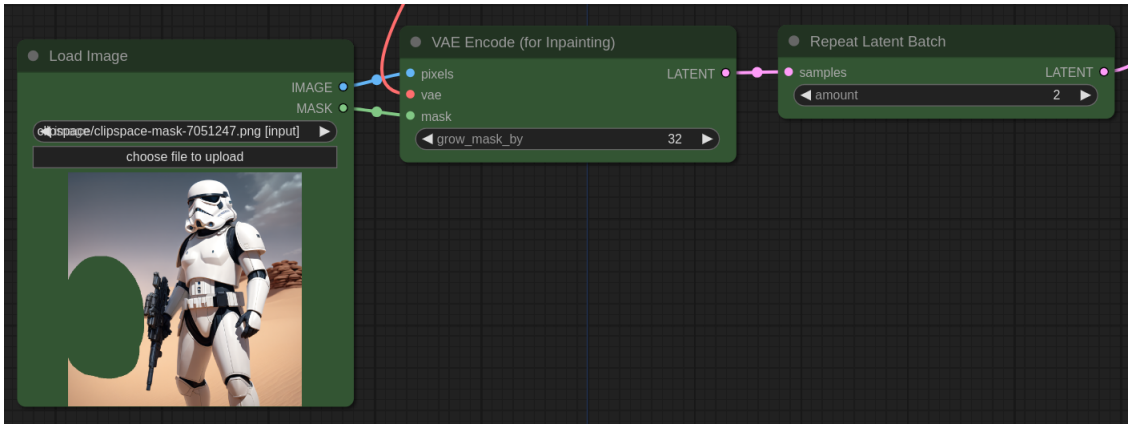


Figure 4.41: Inpainting workflow

- **VAE encode (for inpainting) :** This node uses a VAE to encode images from pixel space to latent space [92]. It takes as input the images to be encoded, a VAE, and the mask indicating where to denoise in the sampler, and returns the masked and encoded latent images. Finally, the "grow_mask_by" parameter, set to 32 here which seems good, indicates by how many pixels the surface of the given mask should be increased.

- **Repeat latent batch :** This node allows repeating a batch of latent images [93]. It is used here to have 2 generations of inpainting, allowing the selection of the best version. It takes as input the batch of latent images to be repeated and the number of repetitions. It returns a new batch of latent images repeated x times.

The generations obtained following the inpainting as described here are not always well-executed. This is an aspect to be improved.

Chapter 5

Consistency between characters

Ensuring consistency in AI-generated character images is a complex challenge, given the variability in training data and the intricacies of human appearance. This involves maintaining uniform facial features and clothing styles across diverse poses and backgrounds. AI models face challenges in preserving consistent details like expressions, poses, and anatomical features, with the potential risk of overfitting. Generative AIs encounter a significant challenge in creating consistent appearances across sequential contexts, such as in a comic series [94, 95].

Stability AI's stable diffusion provides a platform for achieving a high level of consistency, necessitating mastery of its tools and features for success. While reaching absolute uniformity remains a challenge, consistent characters, even if not at 100%, offer advantages for storytelling, branding, and user experience design, contributing to a cohesive narrative, enhancing viewer immersion, and establishing a strong brand image across diverse contexts [96].

With that out of the way, let's take a look at the current possibilities for approaching our goal, which is to enable users to create fairly coherent comic strips. We will see that there are several techniques such as correct use of prompts and fine-tuning methods. There are others which will be mentioned in the conclusion, see Chapter 8, as we do not have the possibility of covering everything. In the meantime, we're going to explain the techniques involved in this work and list the advantages and disadvantages of each in order to make a choice.

5.1 Prompts

A first idea for trying to create coherence between the characters is to use prompts intelligently. In this section, we'll look at how prompts should be used with stable diffusion and particularly with SDXL models. There are various techniques and subjects that can be covered in a prompt in order to achieve consistency. We'll take a look at them through a number of examples, which will give us an interesting first approach to analyse.

5.1.1 Essentials of an effective prompt

First of all, it's important to know what an effective prompt is [94, 97]. Intuitively, it must be complete and precise. A recommended approach is to review a list of keyword categories and determine if any of them align with your intended use.

- **Overall aesthetic of the image :** It's link to the style of the image. This pertains to the artistic expression reflected in the image, encompassing various forms such as comic, cartoon, realistic photograph, impressionist, surrealist, pop art, and others.
- **Character portrayal :** The physical appearance of the character is described in terms of age, hair colour and cut, eye colour, muscles, etc. Everything you want the character to look like.
- **Attire worn by a character :** The character's appearance is described through his or her top, bottoms, shoes, the shape and colours of all these components, etc.
- **Character's behavior :** We also find the attitude, the position, what the character is doing in the image.
- **Location :** The place where it all happens. You need a background that can be more or less detailed.

We could also have other categories like resolution (highly detailed, sharp focus, ...), additional details (stunningly beautiful, dystopian, ...) color of the global image, lighting.

All the things you want to see in the images are grouped together in a single prompt called *Positive prompt*, as you have already seen in a number of examples.

5.1.2 Negative prompt

Using positive prompts alone does not always produce the desired image. This is why the use of negative prompts [97] is also an effective method of guiding the image generation process. Indeed, instead of specifying what you desire, negative prompts involve indicating what you wish to avoid. For instance, negative keywords can pertain to objects or body parts that are undesirable in the generated image. As an illustration, if the rendering of hands is problematic in SD1.5 models, employing "hand" as a negative prompt proves useful in concealing them. There are universal negative prompts.

In SD1.5 models, numerous keywords have minimal impact, and negative prompts are even less crucial for the SDXL model. It is sufficient to specify only the elements you wish to exclude, such as adding "animated" as a negative prompt for generating a photo-type image or removing a specific object. So, although optional, they can be used regularly because they are useful and do no harm. For information, conversely, in SD2 models, the inclusion of negative prompts is indispensable to ensure the production of high-quality images [98].

Contrary to what we saw at Section 3.5.4, when we only managed a positive prompt. Here, the implementation of the negative instruction involves redirecting the process of unconditional sampling. Rather than utilizing an empty instruction that produces random images, a negative instruction is employed [44].

Let's test the influence of negative prompts.

- *Positive prompt :* man standing looking at the camera, in a chalet in the mountains.

You can see in Figure 5.1, where we don't use the negative prompt, that the chalet is automatically located in the mountains in the snow. However, by using the keyword "snow" in the negative prompt, we can now see looking at Figure 5.2 that the image generated no longer places the chalet in the mountains. This proves the usefulness of using a negative prompt.

Figure 5.1: Original image using only a positive prompt

Figure 5.2: Image using the same positive prompt and a negative prompt

5.1.3 Techniques

There are several techniques for using words in a prompt, whether to control the importance to be given to a keyword.

- **Prompt order :** Keyword placement [99, 100] is an interesting element to consider when creating a stable diffusion prompt. Keywords placed at the beginning of the prompt carry more weight than those placed at the end.

  The order of prompts is essential, as the model processes input sequentially. For complex instructions, prompt order is crucial, influencing the model's step-by-step interpretation of the task. The model assigns more weight to tokens encountered early on, ensuring they have a stronger impact. In general, the exact wording of a prompt significantly influences the specific output, making even small changes potentially lead to unpredictable variations in the generated content. This effect can be seen by comparing Figures 4.23 and 4.24. An analysis had already been made under these figures.

- **Keyword weight :** Prompt weighting [97, 101], achieved through the syntax *(keyword: factor)*, offers a valuable tool for influencing the importance of specific elements within a prompt, allowing for precise control over the generated image. The *factor*, a numerical value, determines the degree of importance, with values less than 1 indicating reduced significance and values greater than 1 enhancing importance. This technique is versatile and applicable to various prompt categories, including subject keywords, style, and lighting.

  Understanding that keywords are converted into embedding vectors, altering the importance of a word in the prompt involves adjusting the scale of the text embedding vector associated with that word—either increasing or decreasing it. Subsequently, the model utilizes these embeddings to condition its cross-attention layers for the purpose of image generation.

The sensitivity of the SDXL model to keyword weighting is evident [98], wherein adjusting the weight of keywords, such as (keyword: 1.1), results in a proportional increase in the significance of the keyword, specifically by 10%.

- **() and [] syntax :** An alternative method to modify the intensity of the keyword is to utilize parentheses () and brackets [] [97]. Using *(keyword)* enhances the keyword's intensity by a factor of 1.1, equivalent to (keyword:1.1). Conversely, *[keyword]* diminishes the keyword's intensity by a factor of 0.9, identical to (keyword:0.9). We can use several parentheses or brackets for a same keyword. Doing that the effect is multiplicative. So, ((keyword)) is equivalent to (keyword:1.21) since $1.1 \times 1.1 = 1.21$, (((keyword))) is equivalent to (keyword:1.33), ...

5.1.4 How to write a good prompt?

Finally, in order to conclude this section explaining the interesting methods for constructing good prompts, it is important to summarise them using the following rules :

1. **Be detailed and specific :** While AI is making significant strides, stable diffusion technology cannot intuitively understand your thoughts. It is essential to provide a detailed description of your image for accurate interpretation, especially with SDXL models. This must be done through both positive and negative prompts [102].

   Indeed, as said before, the SDXL model boasts a superior language model compared to the SD1.5. While the SD1.5 tends to treat prompts as a bag of words, the SDXL model has the capability to genuinely comprehend your input. Therefore, it allows you to describe an image in more elaborate and natural language detail. Despite this, the keyword-based approach remains effective [98].

2. **Use powerful keywords :** Certain keywords hold more influence than others in guiding the generated images [102, 103]. The model training relies on images and their accompanying captions, leading to the frequent use of specific words to describe these images. Notably, words such as celebrity names (e.g., Roger Federer), names of artists (e.g., Picasso), and terms related to artistic media (illustration, painting, photography, etc.) carry significant weight when incorporated into prompts. Thoughtful utilization of these influential keywords allows for precise control in directing the generated images toward the desired outcome. It's like learning the vocabulary of a new language. There are lists of keywords on the web like here.

An interesting development of a message-guide should be seen as an iterative process [97]. Always start with a simple guide message containing only the subject, the medium and the style, and generate several images in order to assess the quality. If this is satisfactory, you can go further by gradually adding other keywords. For intelligent assistance, ChatGPT could also be used to create prompts.

5.1.5 Consistency with prompts

The main and most obvious weapon for achieving relative consistency of characters from one generation to the next is to write the guide image in an extremely detailed and relevant way. Once you're happy with the result you've achieved for the guide image, you can reproduce images using similar information to that which you used as your reference.

Figure 5.3: First short comic strip without Face detailer

Figure 5.4: First short comic strip with Face detailer

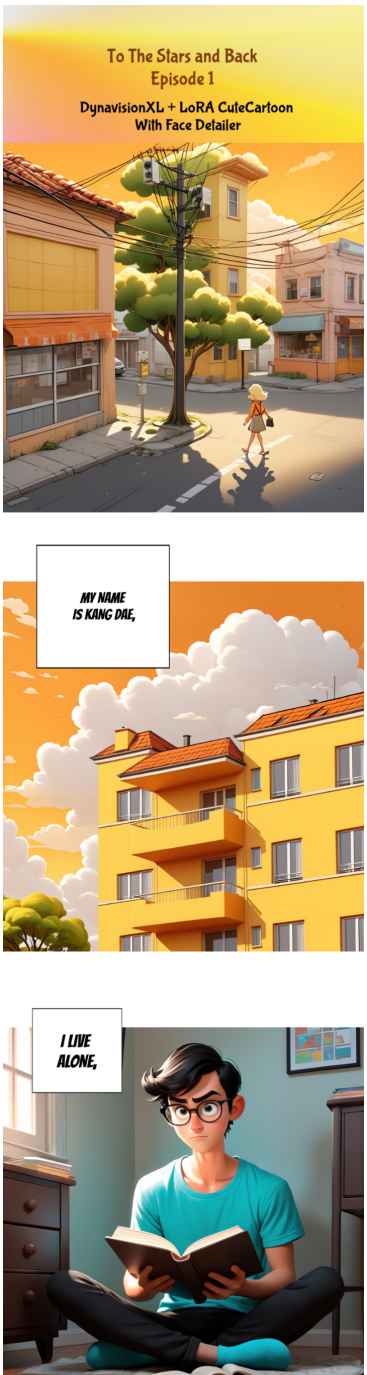


Figure 5.5: First short comic strip with a style LoRA and Face detailer

Furthermore, to showcase the consistency between characters solely through well-crafted prompts, tests were carried out in the examples presented in Section 4.4. The results revealed significant similarities in the generated images with minor variations. The level of coherence achieved can be influenced by the model's quality. If a model encourages diversity in its outputs, maintaining coherence with the prompt becomes more intricate. Therefore, alternative solutions need to be explored, potentially in combination with the current approach.

In a broader context, the diverse images generated using the SDXL workflows from the preceding chapter already enable us to create some appealing initial comic strips that closely resemble the reference comic. The first example, presented in Figure 5.3 without Face detailer, is followed by the second in Figure 5.4 featuring Face detailer, and the third in Figure 5.5 incorporating Face detailer along with an additional LoRA style. The details of LoRAs will be explored later, and for more information, please refer to Section 5.2.4. All the comics are good but the result seems a little better for the third case.

5.2 Methods for fine-tuning stable diffusion models

We will see a second possible solution to obtain consistency between characters : fine-tuning. As said before, fine-tuning [104] is a prevalent approach in machine learning, involving the additional training of a model initially trained on a broad dataset using a more specific dataset. The outcome is a model that tends to produce images resembling those present in the fine-tuning dataset. It allows the users to personalize the models.

There are various fine-tuning methods, each with its own advantages and disadvantages. Most of these methods have the same objective : to train a single concept, such as a subject or style, or several concepts simultaneously. Here, we will describe these methods one by one, understanding how they work and their characteristics. The choice will be guided by our final objective.

5.2.1 Hypernetwork

The Hypernetwork fine-tuning approach [105, 106] (see Figure 5.6), pioneered by Novel AI, involves inserting additional layers into the model. However, instead of updating these layers directly, a separate small secondary network called a Hypernetwork is used to generate the intermediate layers. These will be specifically applied to the cross-attention modules of the UNet noise predictor.

The Hypernetwork is a simple neural network that incorporates a fully connected linear network with dropout and activation functions. Its objective is to predict new weights (matrices) for the original network, essentially modifying the cross-attention module by transforming the key and query vectors.

During the learning phase, the connected Hypernetwork is allowed to adapt while the stable diffusion model remains fixed. In this case, it is the weights of the layers of the Hypernetwork, which is learning to create layers for the stable diffusion network, that are gradually updated, which improves the results.

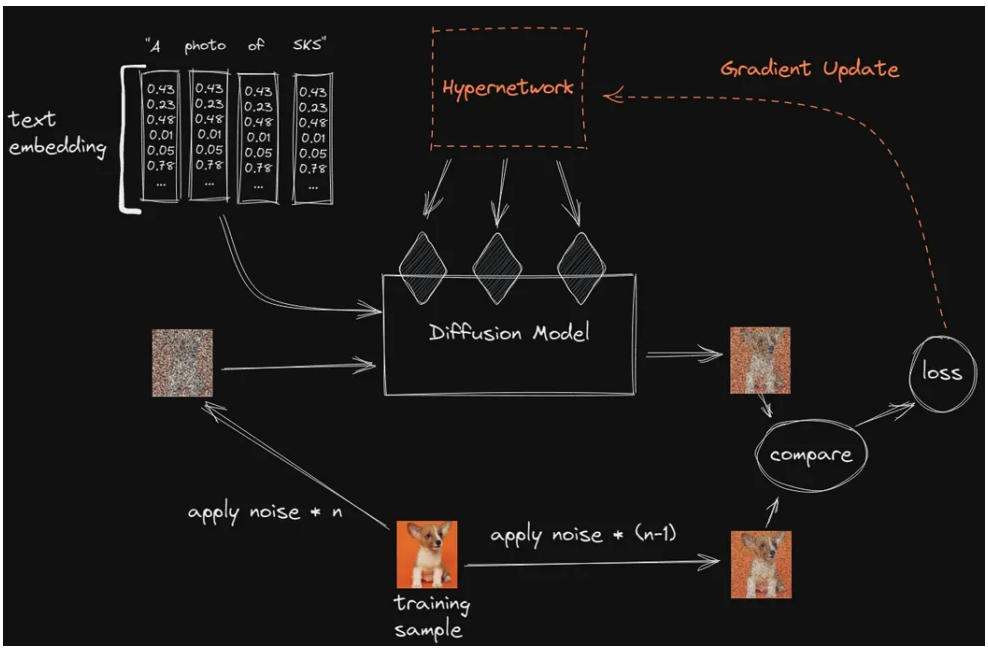


Figure 5.6: Hypernetwork training [107]

Hypernetworks [105], being smaller, require minimal resources for training, which takes place quickly. They also offer the advantage of a reduced file size. However, they will be less efficient than other methods due to their indirect approach.

5.2.2 Textual inversion

The fine-tuning method, referred to as Textual inversion [106, 108, 109] (see Figure 5.7), imparts a new word to the textual model using a small set of example images (3 -> 5). This process ensures the integration of the new word closely with a visual representation, introducing fresh keywords without altering the stable diffusion model's structure. Specifically, only the section related to text integration undergoes refinement, allowing the incorporation of new styles or objects into a predefined model.

The approach involves updating a vector embedding. To execute this method, a novel keyword is defined for the desired object or style, absent in the existing model. This defined keyword is then converted into a token, represented numerically like other prompts in the model. Each keyword is transformed into a distinct embedding vector utilized by the model for image generation.

The training phase initiates by embedding text and noise into the model, initially resulting in an image divergent from the desired concept. By penalizing the model when the generated image deviates from a slightly noisy reference, the vector is gradually updated to yield the desired outcome. This process determines the embedding vector associated with the new keyword.

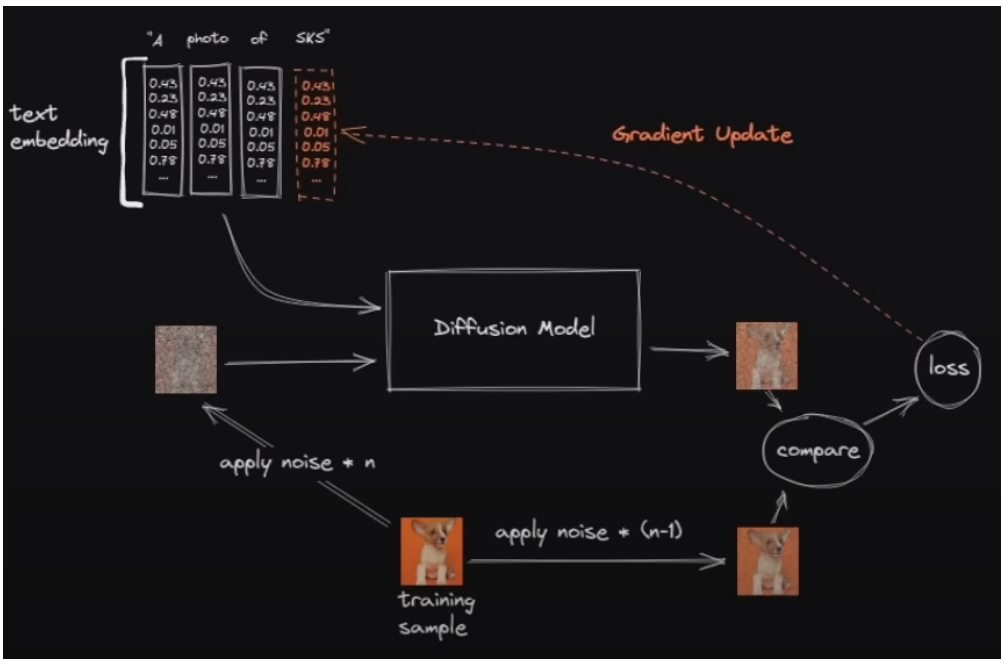


Figure 5.7: Textual inversion training [107]

Textual inversion [108] has the notable advantage that you don't train a whole new model, just a vector. The output is therefore a very small embedding, with a compact file size, which means that several inversions can be stored and easily shared. Several simultaneously generated embeddings can be applied to a single image. Textual inversion works well, although it doesn't always produce the desired effect since it can be more difficult to use than custom models, which can make it suboptimal for some applications. Another drawback to consider is that Textual inversions are model-specific, which limits their generalization.

5.2.3 Dreambooth

Introduced in 2022 by Google's research team, Dreambooth is a training technique designed to enhance the overall stable diffusion model [106, 110, 111] (see Figure 5.8). This method utilizes a small set of images associated with a specific concept, incorporating two inputs: the targeted concept and a unique, meaningless identifier. The inclusion of the identifier is crucial to prevent the AI from associating the concept with common words or other learned terms. The primary objective is for the model to learn the correlation between the unique identifier and the concept, as represented by the provided image samples.

The training process shares similarities with latent diffusion models but differs in that it starts with an existing model and retrains it by introducing new concepts. The procedure commences by converting the concept into an integrated text, where each word is represented by a vector. Subsequently, the model introduces noise to a sample image and attempts to denoise it to align with the given concept. The evaluation involves comparing the model's output with a slightly less noisy version, from which a loss is calculated. The model undergoes updates through gradient adjustments based on this calculated loss.

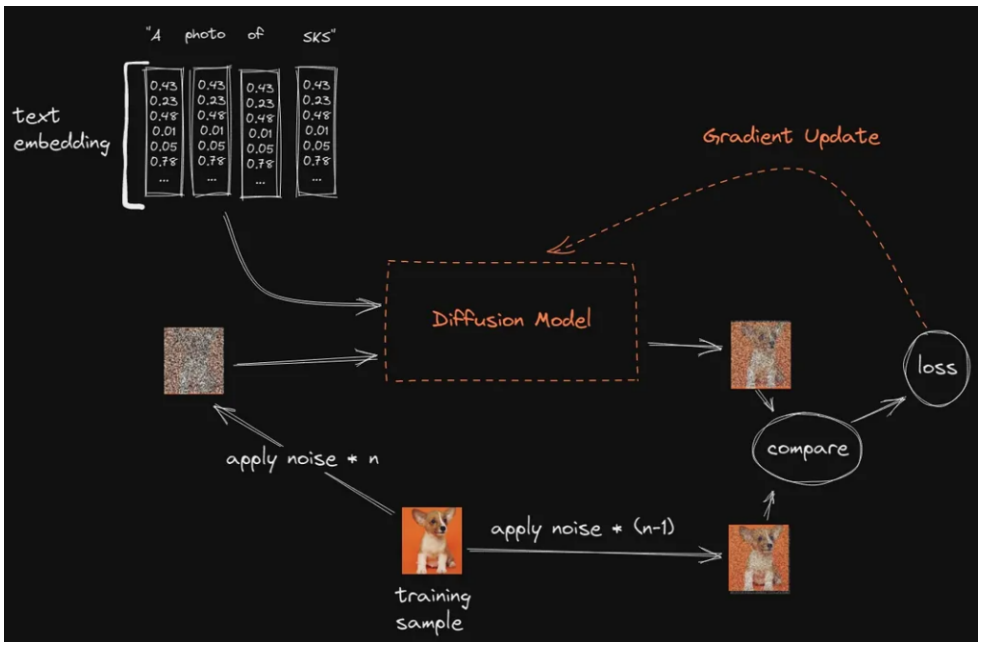


Figure 5.8: Dreambooth training [107]

The Dreambooth method [106] is likely the most effective way to finely tune and shape specific concepts. However, its storage efficiency is compromised as it necessitates the management of an entirely new model.

5.2.4 LoRA

LoRA, or Low-Rank Adaptation of Large Language Models, serves as a mathematical technique developed by Microsoft researchers in 2021 to streamline the parameter count during the fine-tuning of expansive language models [112].

Originally designed for large language models and showcased on transformation blocks, the versatility of the LoRA technique extends beyond its initial scope. Simo Ryu (@cloneofsimo) [113] ingeniously proposed implementing LoRA in the context of stable diffusion [114, 115]. The innovative idea involves integrating these compact LoRA models with an existing standard stable diffusion model, introducing subtle modifications to incorporate new concepts. This adaptation allows LoRAs to be effectively applied to the cross-attention layer, linking image representations to descriptive prompts. Notably, researchers discovered that refining this specific part of the model was sufficient for effective training [116, 117], a concept we will delve into in greater detail below.

5.2.4.1 Technical understanding of LoRAs

We'll start by trying to understand what LoRAs are and how they work technically. Then we'll see how they apply to stable diffusion models. We're going to use the paper that highlighted the LoRAs [118] and several articles summarising well the main principles of this paper [114, 117, 119, 120].

Decomposition of $\Delta \mathbf{W}$: During the standard fine-tuning process, adjustments are made to the weights of a previously trained neural network to adapt to a new task. This entails modifying the original weight matrix, denoted as $\mathbf{W}_0 \in \mathbb{R}^{d \times k}$. The cumulative modifications applied to $\mathbf{W}_0$ in this context are collectively referred to as $\Delta \mathbf{W}$, leading to updated weights expressed as $\mathbf{W}_0 + \Delta \mathbf{W}$.

However, the LoRA method takes a distinct approach by emphasizing the decomposition of $\Delta\mathbf{W}$ instead of directly altering the matrix $\mathbf{W}_0$. This decomposition proves crucial in mitigating the computational intricacies associated with fine-tuning extensive models.

Intrinsic rank hypothesis : The intrinsic rank hypothesis suggests that, despite initial full-rank weight matrices in the dense layers of a neural network, crucial changes when adapting to a new task can be efficiently represented in a lower-dimensional space. Rather than treating every element of the update matrix $\Delta\mathbf{W}$ as equally important, this hypothesis posits that a restricted subset of changes can adequately encapsulate the necessary adjustments. This simplifies the learning process during fine-tuning, allowing more efficient adaptation of models to new tasks.

Introduction of A and B matrices : Based on this assumption, LoRA proposes to represent $\Delta\mathbf{W}$ as the product of two smaller matrices, $\mathbf{A} \in \mathbb{R}^{r\times k}$ and $\mathbf{B} \in \mathbb{R}^{d\times r}$ of lower rank, with $r \ll \min(d, k)$ the rank of a LoRA module. The update matrix of a pre-entrained weight matrix $\mathbf{W}_0$ is then the low rank decomposition $\mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + \mathbf{BA}$.

Thus the modified forward pass for $\mathbf{h} = \mathbf{W}_0\mathbf{x}$ is given by :

$$\mathbf{h} = \mathbf{W}_0\mathbf{x} + \Delta\mathbf{W}\mathbf{x} = \mathbf{W}_0\mathbf{x} + \mathbf{BA}\mathbf{x}$$

During training, $\mathbf{W}_0$ remains fixed without gradient update, while $\mathbf{A}$ and $\mathbf{B}$ contain trainable parameters, representing an approximation to $\Delta\mathbf{W}$. This approach allows $\mathbf{A}$ and $\mathbf{B}$ to be fine-tuned instead of $\mathbf{W}$. The end result is a remarkably compact model, significantly smaller than $\mathbf{W}$. Both $\mathbf{W}_0$ and $\Delta\mathbf{W} = \mathbf{BA}$ are multiplied by the same input, and their output vectors are added together in the coordinate direction.

For reparametrization, as depicted on Figure 5.9, $\mathbf{A}$ is initialized with a random Gaussian distribution, and $\mathbf{B}$ is initialized with zeros. Thus, $\Delta\mathbf{W} = \mathbf{BA}$ is zero at the start of training. Subsequently, $\Delta\mathbf{W}\mathbf{x}$ is scaled by $\frac{\alpha}{r}$ where α is a constant in r you can tune. Setting α to the first r tried eliminates the need for further tuning hyperparameters and helps maintain stability when varying r.

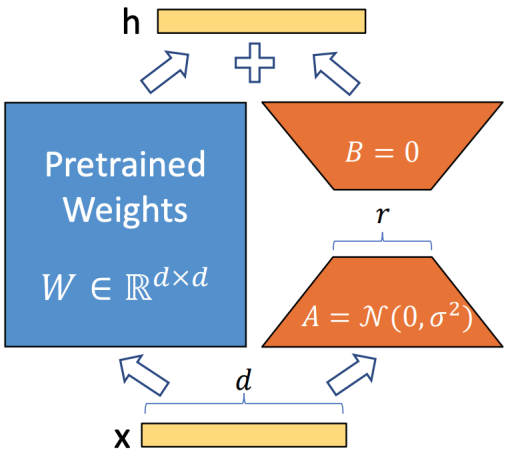


Figure 5.9: LoRA - Reparametrization [118]

Let's illustrate the reduction in size achieved by LoRAs through an example [117]. Consider a model with a matrix of dimensions 2000 rows by 4000 columns, totaling $8.000.000$ numbers (2000×4000). Storing such a large set of values in the model file isn't optimal. We can do better. With the LoRA technique, we can decompose this matrix into a 2000 by 2 matrix and a 4000 by 2 matrix, where the number 2 indicates the rank of the matrices. Consequently, only 12000 numbers $(2000 \times 2 + 2 \times 4000)$ need to be stored. This represents a reduction of 666 times in file size. The term "low-rank matrices" is aptly coined as we observe that the rank is significantly smaller compared to the original dimensions, and the rank can be as minimal as 1. This succinctly explains why LoRA files are considerably more compact.

Applying LoRAs to stable diffusion : LoRAs offer an innovative approach to optimize neural networks by specifically targeting subsets of weight matrices. This results in a reduction of the overall number of parameters that need to be trained. In the context of transformers, LoRAs exhibit a selective adjustment mechanism, focusing on certain weight projection matrices and self-attention layers, while keeping other Multi-Layer Perceptron (MLP) modules fixed. This targeted approach streamlines the fine-tuning process, especially in handling matrices like $\mathbf{W}_Q$ (or $\mathbf{W}_K$, $\mathbf{W}_V$, $\mathbf{W}_O$) within each attention module in a unified manner. Notably, LoRAs deviate from the traditional fine-tuning method by freezing pre-trained weights and introducing new low-rank decomposition matrices alongside existing ones. Subsequently, training is exclusively applied to these introduced matrices, effectively minimizing the total number of parameters to store. This approach ensures the preservation of the capability to precisely adjust cross-attention layers.

Understanding the foundational concept of LoRAs, its application to stable diffusion becomes straightforward as depicted in Figure 5.10. In the fine-tuning process of stable diffusion, LoRAs are exclusively applied to adjust the diffusion part, which adopts a UNet architecture. Given that LoRAs are specifically applied to the attention module, it becomes imperative to identify the number of attention modules within the diffusion model. Once this is determined, the corresponding attention layers are frozen, and LoRA matrices are created for the fine-tuning process. This meticulous approach ensures that the benefits of LoRAs are leveraged precisely where needed, contributing to the optimization of the stable diffusion model.

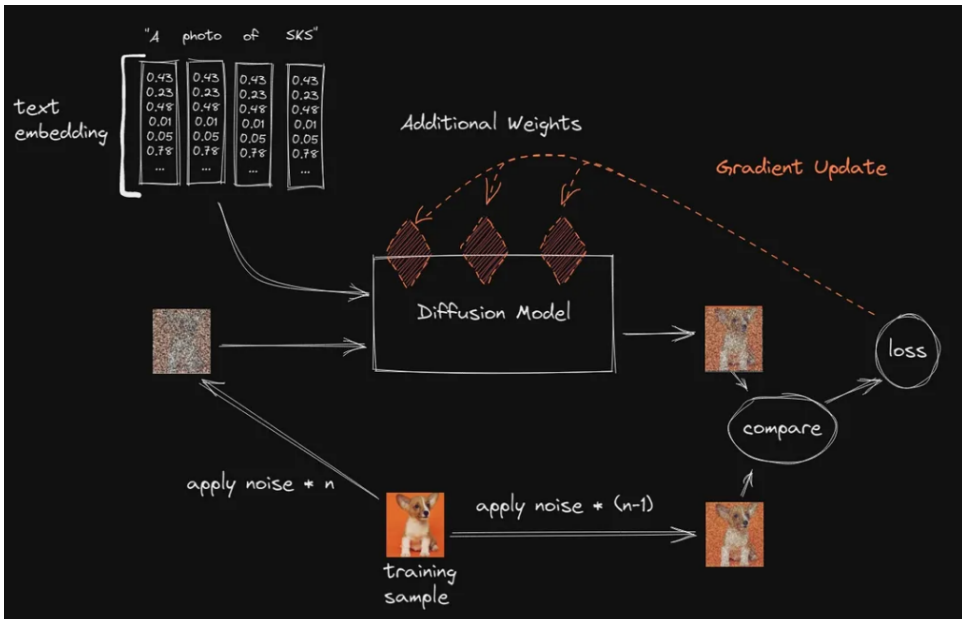


Figure 5.10: LoRA training [107]

5.2.4.2 Advantages of LoRAs

By reducing the number of parameters available for training, LoRAs come with several advantages [115, 119, 121], especially when fine tuning extended neural networks. Here are a few :

- **Faster training and less memory consuming :** LoRA focuses on updating specific layers rather than the entire model, leading to faster learning times and reduced memory usage. It facilitates the management of large-scale models.
- **Reduced output size :** Trained weights are considerably smaller. By introducing new trainable layers into a frozen original model, the weights for these layers can be saved in a single, lightweight file.
- **Adaptability to multiple models :** LoRAs can be integrated into different models since they produce smaller external files with fewer parameters for easier sharing, storage, and reuse.
- **Ease of access :** Users can create their own LoRA models or explore pre-trained ones on platforms like CivitAI or HuggingFace, which offer a broad range of models. The affordability of resources for training and using LoRAs contributes to this accessibility. The GPU requirements are low, and the training can be carried out on a good personal computer.
- **Control :** LoRA matrices are typically added to the attention layers of the original model, allowing control over the degree to which the model adapts to new training images through a scaling parameter.

In summary, the numerous advantages of LoRA training contribute to its popularity, establishing it as the most accessible choice in contemporary scenarios.

5.2.4.3 What can LoRAs do?

LoRAs exhibit remarkable versatility and have been employed by the stable diffusion community in various applications, including [112] :

- Enhancing quality,
- Exploring styles and aesthetics,
- Generating characters or depictions of individuals,.
- Creating representations of clothing or objects,.
- Designing diverse settings.

5.2.4.4 How to use LoRAs in ComfyUI?

To initiate the loading of a LoRA, you can utilize a node within ComfyUI named "Load LoRA" [122, 123] (see Figure 5.11). This node requires inputs such as the diffusion model and the specified CLIP model, which the designated LoRA will adjust to alter the denoising process for latents. Additionally, this node considers parameters like "strength_model," indicating the extent of modification to the diffusion model, and "strength_clip," denoting the degree of adjustment to the CLIP model. It's noteworthy that these values can be negative. The resulting outputs from this process are the models that have been appropriately modified. So, LoRAs are patches applied on top of the diffusion model and the CLIP model.

Figure 5.11: LoRA loader

To incorporate a LoRA with a weight in ComfyUI, we can employ the subsequent expression within the prompt [117] : <lora:LORA-FILENAME:WEIGHT>, where :

- "LORA-FILENAME" represents the LoRA file name, excluding the extension (.pt, .safetensor, etc.), and,
- "WEIGHT" signifies the strength assigned to the LoRA model. This is similar to the keyword weight. The impact of the LoRA model can be adjusted by modifying the weight value, allowing for an increase or decrease in its effect. A higher value amplifies the influence of LoRA on the outcome, typically ranging between 0 and 1. The default setting is 1, where 0 disables the model. Notably, as stated in the ComfyUI Community Manual, it's possible to define LoRA strength values as negative, leading to occasionally intriguing effects.

Importantly, note that the LoRA expression itself is not considered as part of the prompt ; it will be removed once the LoRA has been applied.

Moreover, the utilization of LoRAs may necessitate the inclusion of trigger keywords in the prompt to activate specific concepts. Simultaneously, multiple LoRAs can be employed by specifying various LoRA expressions. We can do this in ComfyUI by chaining Lora loaders. Ultimately, enhanced outcomes are achieved when models are linked with LoRAs derived from the common base model on which they were originally trained.

Chapter 6

LoRAs analysis and training

In summary, LoRAs provide a balanced trade-off between file size and training capability. Therefore, based on information from Section 5.2, our preference about the fine-tuning methods leans toward LoRAs. Consequently, in this chapter, we will evaluate their effectiveness and proceed to train our own LoRAs. We will assess how prompts can impact LoRAs and link the coherence achievable through prompts to that attainable in LoRAs.

6.1 Exploring consistency and flexibility in LoRA-generated characters

In this section, we will try to see the influence that LoRAs can have on a model. First, its important to note that modifying the "strength_model" and "strength_clip" parameters plays a crucial role on the result. However, since the way in which these parameters are set is specific to the LoRA model that has been formed, no tests have been carried out on the LoRAs chosen. We'll use the values recommended by the people who trained the LoRAs we're going to use for our tests. Therefore, thanks to the following subsections, we'll see if it's possible to modify the appearance of the LoRAs while maintaining consistency, and we'll evaluate the importance of the trigger words[1] linked to the LoRAs. This will tell us whether it is worth using LoRAs and possibly give us some ideas for the design of the application.

6.1.1 Study of the consistency of animated-style LoRAs with modifications via prompts

Our primary objective is to strike a balance between consistency and diversity in user-generated characters through LoRA training. Indeed, our exploration extends to the fascinating prospect of introducing flexibility while maintaining consistency. We aim to assess whether LoRA characters can be seamlessly modified while retaining some distinct characteristics. Initial experiments concentrate on LoRA characters trained with animated-style images.

These tests are tailored to a comic strip style, given its prevalent animated aesthetic. If successful, subsequent experiments will involve LoRAs trained with images of real people, aiming to generate stylized images suitable for a comic strip. This approach allows us to gauge whether training on real people's photos enables the generation of stylized images and whether subsequent modifications are

[1] A trigger word is a keyword that triggers the application of the LoRA when we generate an image.

feasible. This process simplifies the assembly of a diverse dataset.

Before delving into experiments, we make assumptions about the feasibility of our approach. LoRA training allows the integration of new concepts, with the foundational CLIP and denoising models well-versed in a myriad of concepts. Proper training and prompt utilization, leveraging CLIP-recognized keywords, are expected to enable the model to associate existing knowledge with new concepts.

This first study of LoRA modification via prompts is therefore based on 4 different animated-style LoRAs, 2 female and 2 male, using the same type of modification for each LoRA and comparing with 3 different seeds in order to draw relevant conclusions. The 4 SDXL LoRAs used were found on CivitAI and are : Alice (Figure 6.3), Makima (Figure 3 in the appendix), Choi Kang-rim (Figure 6.4) and Gojou Satoru (Figure 4 in the appendix). Below, we give their trigger words and the positive original prompts, corresponding to "Prompt 0".

Model : dynavisionXLAllInOneStylized

- **Alice :**
  - *Trigger words :* AliceDV, orange hair, orange eyes, short twintails, hair ornament.
  - *Positive prompt :* cartoon style, woman drinking a beer in a tavern in the far west, AliceDV, white shirt, (black jacket:1.2), jeans trousers, brown belt, red scarf tied around her neck, high quality.
- **Makima :**
  - *Trigger words :* makima, braided ponytail, ringed eyes, collared shirt, black necktie, black pants and red hair and yellow eyes if necessary.
  - *Positive prompt :* cartoon style, woman strolling through a snowy Christmas market at night, makima, heavy light brown coat, denim trousers, white shirt, white snow boots, high quality, (beautiful face), beautiful eyes.
- **Choi Kang-rim :**
  - *Trigger words :* Anime boy, Choi Kang-rim.
  - *Positive prompt :* anime boy dancing by the river, Choi Kang-rim, smiling, black cargo trousers, (beige jumper:1.2), forest by the river, high quality.
- **Gojou Satoru :**
  - *Trigger words :* gojou satoru.
  - *Positive prompt :* cartoon style, gojou satoru is holding an ice cream on the beach, under a palm tree on his deckchair, casual area, white shorts, red t-shirt, sun is shining, high quality, beautiful face.

Here are the different modifications chosen for the tests :

- *Prompt* 0 : It defines the original image and the seed on which the other prompts will be based.
- *Prompt* 1 : We want to add a light brown cowboy hat.

- *Prompt* 2 : We want to change the hair colour to pink and the eyes colour to purple.
- *Prompt* 3 : We would like to make the character bigger.
- *Prompt* 4 : We want the character to be fat, have pink hair, purple eyes and wear a light brown cowboy hat.
- *Prompt* 5 : We want to add green sunglasses.

By analysing the images generated between the different seeds for the same LoRA character, we can see that consistency is achieved and that it is also possible to play with the prompt to modify the appearance of the characters slightly. That said, you absolutely must have a good starting prompt, which is sometimes difficult to obtain. In terms of physical appearance, the game is fairly consistent, but this isn't always the case for the way the characters are dressed. We still have the problem of understanding the colours in the images.

Other examples show that it is possible to modify a LoRA character and provide us with further information. Indeed, 2 other modifications have been applied to the Alice LoRA (see Figures 6.1 and 6.2). The second attempts to change the length of Alice's hair and remove her ornament. However, after repeated attempts, it was not possible to do better than the result obtained in Figure 6.2 where it was complicated to remove the ornament. This is probably due to the fact that the images and captions used for training all have this feature, this keyword. The model is too well trained on this feature. In order to change this, we would also need to have data without this feature. It's more difficult to remove a style element than to add one. Other examples have been produced, present in the appendix, concerning the LoRA Choi Kang-rim in Figures 5, 6, 7, 8 and concerning the LoRA Makima in Figure 9.

Figure 6.1: Modification : brown hair and blue eyes - Seed a - Alice LoRA

Figure 6.2: Modification : no hat, very long hair and without the ornament - Seed a - Alice LoRA

In conclusion, overall, the tests of the 4 animated-style LoRAs are satisfactory and confirm our initial hypotheses. What's more, by not using all the trigger words and using weights on added elements, it's possible to change the character's appearance. We can now go one step further and move on to the next stage mentioned above. It should be noted that these tests depend on the quality of the LoRAs training carried out.

Figure 6.3: Study of consistency and modifications of LoRAs via prompts - Alice LoRA

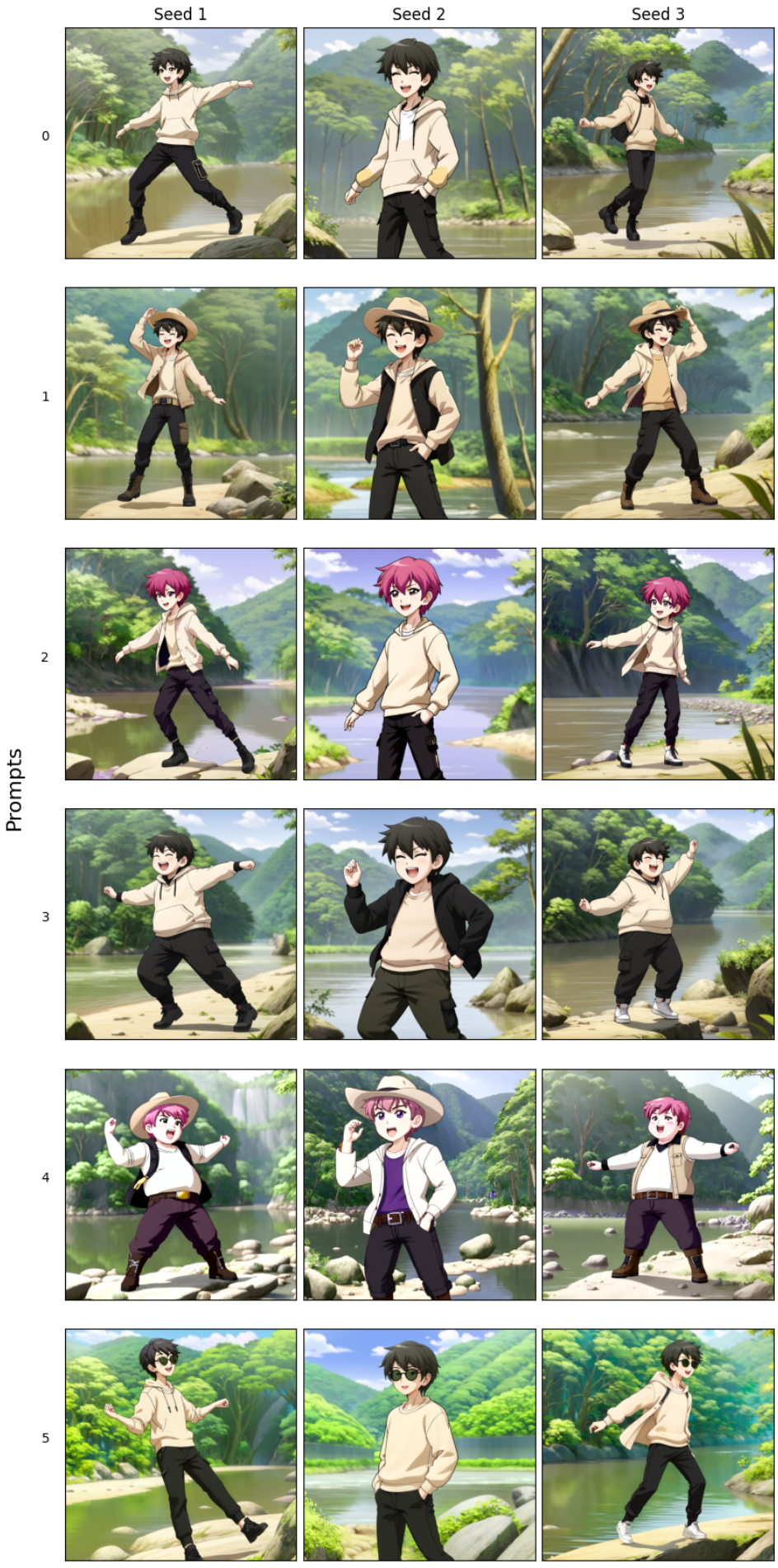


Figure 6.4: Study of consistency and modifications of LoRAs via prompts - Choi Kang-rim LoRA

6.1.2 Study of the consistency of real-person LoRAs with modifications via prompts

Having established the feasibility of maintaining consistency in LoRAs trained on animated-style data and making slight modifications, we aim to explore if similar findings can be extended to LoRAs trained on datasets comprising real photos of individuals. Our focus lies in assessing whether these LoRA models, despite being trained on authentic images, possess the capacity to correlate with stylized models or cartoon-related descriptors, thereby generating outputs different from their training data format. Additionally, we will investigate the potential for altering their appearance through prompts.

This is very interesting because it's easier to create training data on photos you've taken of yourself or someone else than it is to create a character in a particular style and have several copies of it.

Thus, this second study of LoRA modification via prompts is based on 3 different celebrity LoRAs, 1 female and 2 male, using the same type of modification for each LoRA and comparing with 3 different seeds in order to draw relevant conclusions. In addition, a LoRA was also used to obtain cartoon-style images : the CuteCartoon LoRA, with the trigger words "CuteCartoonAF" and "Cute Cartoon". The 3 celebrity SDXL LoRAs used were found on CivitAI and are : Cillian Murphy (Figure 6.5), Jason Momoa (Figure 10) and Rihanna (Figure 6.6 in the appendix). Below we show their trigger words, the original positive prompts without LoRA style, corresponding to "Prompt 0", and with LoRA style, corresponding to "Prompt 2". Only one modification, "blue hair", is applied for now, as we mainly want to see if it's possible to have cartoon images based on training images of photos of real people.

Model : dynavisionXLAllInOneStylized

- **Cillian Murphy :**
  - *Trigger words :* ci11i@nmur&ŷ, a man.
  - *Positive prompt without style LoRA :* (cartoon style:1.5), a man is holding an ice cream on the beach, under a palm tree on his deckchair, ci11i@nmur&ŷ, casual area, white shorts, red t-shirt, sun is shining, high quality, beautiful face, (beautiful eyes).
  - *Positive prompt with style LoRA :* (CuteCartoonAF, Cute Cartoon), a man is holding an ice cream on the beach, under a palm tree on his deckchair, (ci11i@nmur&ŷ:1.2), casual area, white shorts, red t-shirt, sun is shining, high quality, beautiful face, (beautiful eyes).
- **Jason Momoa :**
  - *Trigger words :* jason momoa.
  - *Positive prompt without style LoRA :* (anime art:1.5), (cartoon style:1.5), man strolling through a snowy Christmas market at night, jason momoa, smiling, happy, heavy light brown coat, denim trousers, white shirt, white snow boots, high quality, (beautiful face), beautiful eyes.
  - *Positive prompt with style LoRA :* (anime art:1.4), (Cute Cartoon, CuteCartoonAF:1.45), man strolling through a snowy Christmas market at night, jason momoa, smiling, happy, heavy light brown coat, denim trousers, white shirt, white snow boots, high quality, (beautiful face), beautiful eyes.

- **Rihanna :**
  - *Trigger words :* ohwx woman, ohwx.
  - *Positive prompt without style LoRA :* (cartoon style:1.4), ohwx woman drinking a beer in a tavern in the far west, ohwx, white shirt, (black jacket:1.2), jeans trousers, brown belt, red scarf tied around her neck, high quality.
  - *Positive prompt with style LoRA :* (CuteCartoonAF, Cute Cartoon:1.2), ohwx woman drinking a beer in a tavern in the far west, ohwx, white shirt, (black jacket:1.2), jeans trousers, brown belt, red scarf tied around her neck, high quality.

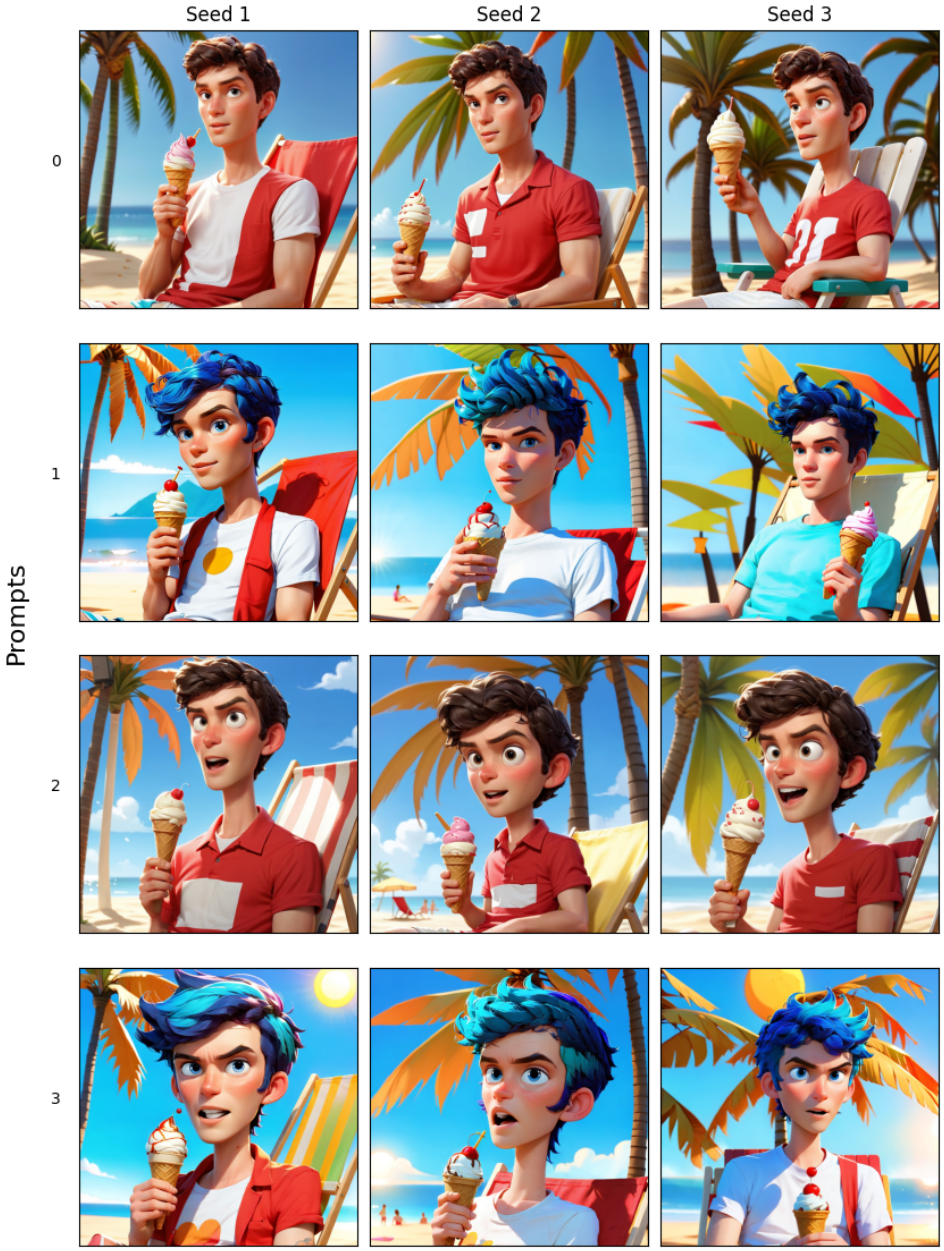


Figure 6.5: Study of consistency and modifications of LoRAs via prompts with real-person LoRA - Cillian Murphy LoRA

Figure 6.6: Study of consistency and modifications of LoRAs via prompts with real-person LoRA - Rihanna LoRA

Through an examination of the aforementioned tables, it becomes evident that the diffusion model can produce cartoon images utilizing LoRAs trained on real people's photos, and the resulting character in the image is subject to modification. This outcome is logical, as the foundational diffusion model, although not specifically acquainted with cartoon concepts through LoRA training, successfully establishes a connection due to its extensive training on numerous cartoon images. Moreover, it's also due to the fact that we give more weight on the "cartoon style" keyword, given that we want the character to have a cartoon style. The association becomes even more evident with the application of LoRA CuteCartoon, resulting in a more stylized image. This demonstrates the successful combination of multiple LoRAs, contingent upon their effective training, encompassing parameters, training duration, and the diversity of training images.

For instance, the style LoRA yields aesthetically pleasing generations ; however, variations in the character's age, frequent frowning, and open mouth smiles are noticeable. Similar inconsistencies

are observed among the LoRA characters, indicating varying quality levels. Notably, Rihanna's LoRA stands out as it avoids appearing over-trained on specific characteristics, resulting in higher-quality generated images compared to the other two LoRAs. In summary, the conducted tests yielded equally satisfactory results.

6.1.3 Conclusion

So, a final examination would involve conducting a test identical to the one in Section 6.1.1 for Rihanna's LoRA character, but this time using a cartoon-style LoRA like in Section 6.1.2. As illustrated in Figure 6.7, the table presents results for 3 different seeds and 6 distinct prompts, incorporating the same modifications as before. Below, we specify the trigger words and the original positive messages associated with "Prompt 0".

Model : dynavisionXLAllInOneStylized

- **Rihanna :**
  - *Trigger words :* ohwx woman, ohwx.
  - *Positive prompt :* (CuteCartoonAF, Cute Cartoon:1.2), ohwx woman drinking a beer in a tavern in the far west, ohwx, white shirt, (black jacket:1.2), jeans trousers, brown belt, red scarf tied around her neck, high quality.

We observe that the findings derived from the images produced in this test align with our observations from prior tests. Consequently, the results are decisive, enabling us to progress to the subsequent significant phase. Our next step involves training our own LoRA using real photos of individuals, and this will be the focus of the upcoming section.

In addition, as the tests proved conclusive, an interesting idea would be to create a LoRAs database of real people with different characteristics. So, a future project, outside the scope of this work, would be to create and use an AI system to associate the keywords entered in the prompt by the user, relating to the physical description of the characters, with the trigger words of the LoRAs characters with similar physical characteristics. We could then combine the characteristics of different LoRA characters to create a unique character.

Figure 6.7: Study of consistency and modifications of LoRAs via prompts with real-person LoRA - Rihanna LoRA

6.2 LoRA training

After conducting numerous tests on pre-trained LoRAs available on established platforms like HuggingFace or CivitAI, we observed that maintaining and altering consistency is feasible by adjusting the character through prompts. This adaptability applies to both images featuring animated characters and those based on real characters, including celebrities. Notably, we successfully altered the character's style with ease.

Given these findings, we believe it is advantageous to train our own LoRA specific to our character for the purpose of ensuring consistency in our upcoming comic. As mentioned earlier, incorporating a dataset with diverse LoRAs from real individuals adds richness to our content. Consequently, we have decided to train a LoRA specifically for myself, Romain. The objective is to assess the quality of the results and identify potential areas for improvement. The ultimate goal is to showcase a comic creation integrating the valuable insights gained during this thesis, evaluate its satisfaction level, and explore avenues for enhancement.

Now that we know who's going to be the focus of attention in these tests, we need to select data, caption the data, choose which tool we're going to use to carry out the training, train and finally test the results obtained.

It's crucial to highlight that the computational power required for training SDXL LoRAs posed a limitation, preventing the attainment of LoRAs of significantly higher quality. VRAM plays a crucial role in the graphics card's ability to handle large data sets and complex models. Indeed, a larger VRAM allows larger data to be stored and manipulated in memory, which can speed up the training process. It can enable the use of larger batches, which can improve training stability and lead to more robust model updates. It can also facilitate the training of more complex models. Specifically, the LoRAs underwent training on Google's cloud services using an NVIDIA T4 with 16GB VRAM. Nevertheless, utilizing a datacenter GPU such as the A100 with 80GB VRAM or an RTX 3090/4090 with 24GB VRAM would have been more appropriate and desirable for comprehensive testing [124]. Although higher-performance GPUs would expedite the training process, they come with a higher cost. The subsequent discussion will delve into the decisions made and the challenges encountered as a consequence.

6.2.1 Kohya-ss LoRA training

For the training of LoRAs, we won't develop new methodologies to avoid time-consuming efforts and potential subpar outcomes. Instead, we plan to leverage well-established practices that have been in existence for several months, undergone numerous refinements, and gained widespread adoption.

To achieve this, we will utilize existing scripts authored by Kohya-ss [125], which are accessible on his GitHub page and also designed to support SDXL. These scripts, using PyTorch, encompass various functionalities essential for stable diffusion, including :

- DreamBooth training, featuring UNet and Text encoder,
- Fine-tuning through native training, incorporating UNet and Text encoder,
- LoRA training, based on the cloneofsimo repository [113], which interests us,

- Textual inversion training,
- Image generation,
- Template conversion, supporting both 1.x and 2.x, stable diffusion ckpt/safetensors, and diffusers.

Furthermore, bmaltais [126] has contributed a user-friendly graphical interface on their GitHub for Kohya's stable diffusion trainers, compatible with both Windows and Linux. This interface facilitates the definition of training parameters and the generation and execution of the necessary CLI commands for model training.

6.2.2 Training data pre-processing

The first and most important step is to collect images and caption them. Next, you need to organise them in a format suitable for the training algorithm. To get an idea of how to make a good dataset and captions, we consulted several articles [112, 124, 127].

6.2.2.1 Gathering training images

To construct our dataset, we captured numerous photographs from various perspectives, including close-ups of the face, medium shots, and full-body shots. The photos featured diverse lighting conditions and shadows, albeit within the realm of typical scenarios. We incorporated variations in postures, clothing, backgrounds, and facial expressions with the objective of achieving maximal diversity. Given that this is an important element, especially in webtoon-style comics, we wanted more qualitative images of the face to capture as much detail as possible. This diversity in the dataset aims to mitigate the risk of the model unintentionally memorizing specific unwanted attributes present in the training images.

To further enhance dataset quality, we deliberately excluded blurred images, pictures with additional individuals, unconventional styles or poses, and instances where the person is partially obscured by objects. These precautions were taken to prevent any potential bias in the model and maintain a focus on the intended facial recognition task.

Having a larger quantity of images is advantageous, but the quality of the images is a crucial factor. It's not advisable to include images solely for the sake of increasing quantity, especially if those images are of low quality. In our case, we opted to utilize approximately thirty images, as illustrated in Figure 6.8. Although testing with more images would have been desirable, the constraints imposed by the available resources prevented us from doing so. Typically, the practice involves working with a dataset comprising 30 to 150 images that are accurately labeled.

An extensive, high-quality dataset is vital for training machine learning models, ensuring broad applicability and minimizing biases. Diverse data enhances adaptability, avoiding overfitting and enhancing accuracy across various scenarios. Strategic image selection is crucial to impart flexibility in handling elements like clothes and backgrounds. Teaching the AI about specific elements requires clear images with and without the element, enabling precise control and preventing overemphasis. The UNet model's learning process seems centered around recognizing differences, underscoring the significance of diverse examples in effective training.

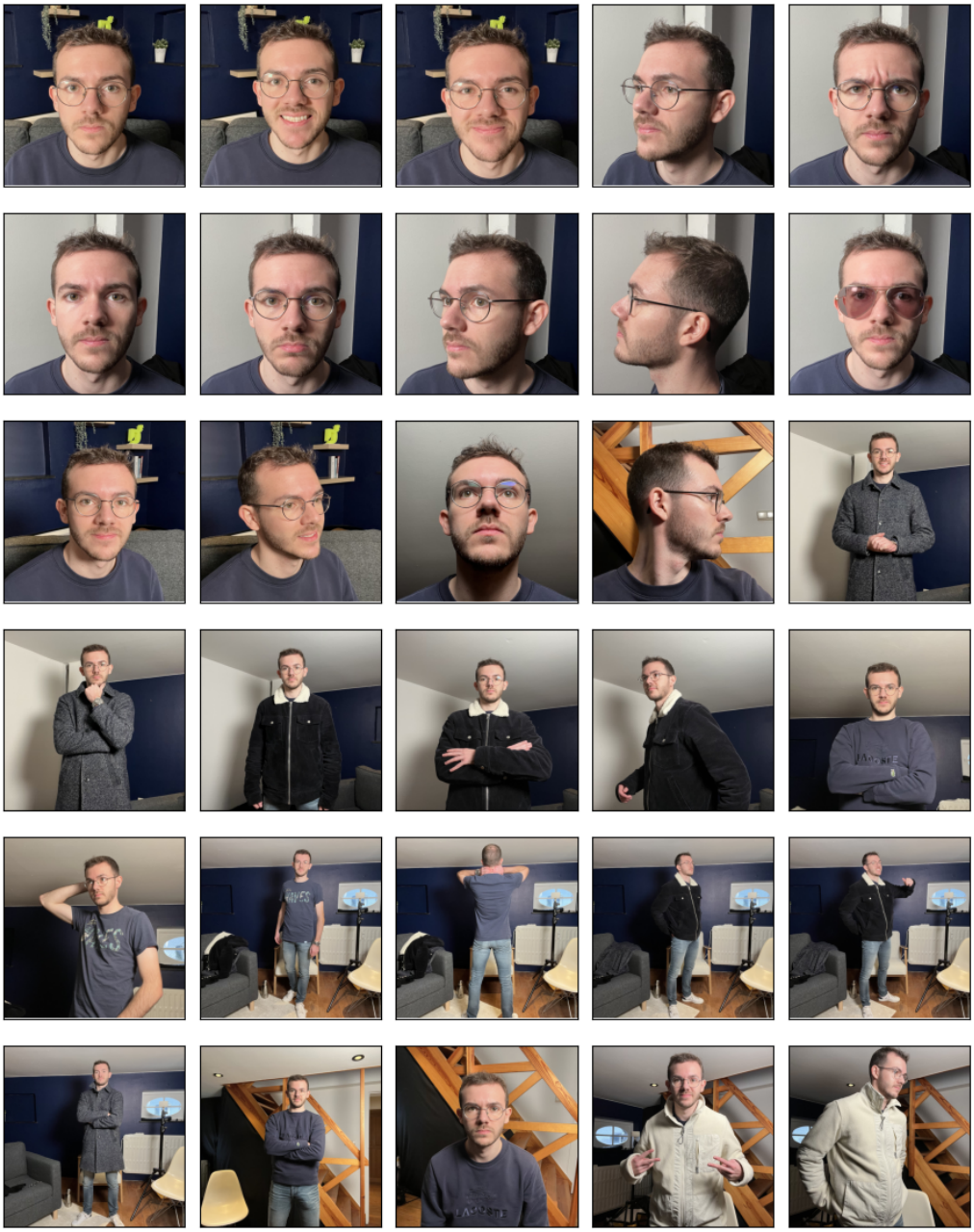


Figure 6.8: Training images

As concern the image size, it is advisable to use images that are ideal squares with dimensions greater than or equal to 1024×1024 to avoid compatibility issues with SDXL LoRA. Consequently, we have opted exclusively for 1024×1024 images. This decision stems from the fact that larger images entail a higher demand for computational power during processing. This is primarily attributed to the increased volume of data and pixels to be managed and stored at each stage of the processing, resulting in a higher number of parameters to handle. It is noteworthy that all images adhere to the same JPG format.

6.2.2.2 Captioning training images

Next, once we've selected our images, we need to caption all of them for the course. We'll create a .txt file for each image describing its content.

Image captioning entails adding descriptive labels or tags to images, emphasizing particular details or components. This procedure aids the model in understanding the connection between these tagged elements and whether they are present or absent in various images. It plays a crucial role in ensuring uniformity and minimizing confusion for the model, especially when dealing with variations in appearance. The accuracy of model outcomes is contingent on the effectiveness of the captioning process.

In order to accomplish this, it is crucial to recognize the essential classifications for captioning, which include the perspective (such as close-up, medium-shot, full body), the style (photo, sketch, painting), the subject (like a man, a woman, a boy), the activity (such as walking, eating, running), attire (such as wearing a shirt, wearing trousers, donning a dress), physical attributes (like eyes, hair, facial expressions), background, and additional elements (like watermarks, text, low-quality). Providing captions for backgrounds and outfits can be beneficial in mitigating recall issues when there is a limited diversity in the image dataset.

Captions will more or less follow the following format : TRIGGER, CLASS, <any element that is not universally present in all images>. Some examples are presented in Table 6.1.

- "TRIGGER" : designates a term which must be included in the prompt once the LoRA is to be applied. It must be a rare token, i.e. a word which is unique so that it is not misinterpreted by the model. Here, we use "rc_test".
- "CLASS" : is the subject of the training. In our case, we use "man".

6.2.2.3 Folder structure

Once our dataset has been established, we need to structure it in a way that the training algorithm can understand. Consequently, the typical folder structure used is as follows. The folder contains 4 other folders :

- **img :** This is where all the images and captions collected for training will be stored. Inside this img folder, before going back to all the training content, there is a pre-folder named according to convention : REPEAT_TRIGGER CLASS.
  - "REPEAT" : represents the number of passes that the training algorithm will make per epoch on each image.

  In the end, the folder will be called : 20_rc_test man.
- **model :** where the checkpoints of the model you decide to save will be placed.
- **log :** optional. This is where the logs will go.
- **reg :** where you place the regularisation images if you decide to use them. In this case, a pre-folder will have to be created, similar to the img folder, except that this time the convention to be respected is: REPEAT_CLASS where "CLASS" is the same as that used previously for img.

Captions	Training images
rc_test, photo of a man, close up, from front, looking at the camera, sitting on a couch, round glasses, brown beard, short curly brown hair, green-brown eyes, dark blue jumper, dark blue background	
rc_test, photo of a man, close up, from front, looking at the camera, happy, smiling with mouth open, visible teeth, sitting on a couch, round glasses, brown beard, short curly brown hair, green-brown eyes, dark blue jumper, dark blue background	
rc_test, photo of a man, medium shot, walking in a room, looks off to the side in front of him, round glasses, brown beard, short curly brown hair, green-brown eyes, black velvet jacket with molton collar, dark blue and white background	
rc_test, photo of a man, full body, from front, looking at the camera, standing in a room next to a couch, round glasses, brown beard, short curly brown hair, green-brown eyes, blue t-shirt, jeans trousers, denim trousers, silver watch on wrist, white shoes, dark blue and white background with a small window, a radiator	

Table 6.1: Examples of captions for training images

A regularization image [128] refers to any image sharing the same class as the subject of training but is not the specific subject being trained. The use of regularization images is discretionary. It introduces a moderating influence on training, helping to prevent class drift and overfitting. It is recommended to generate these images using the base model employed for training. However, they are not used here, as they are only necessary when there is a large number of training images, which is not the case here.

6.2.3 Training parameters

Now that the images have been chosen and captioned, and everything has been put into the right folders with the right names, we can select the parameters to be used for training. There are a number of them, and we're going to take a look at some of the most important. We'll briefly discuss how they work and what they're intended to achieve. This will enable us later to select the right values with full knowledge of the facts. It's important to note that specific datasets and hardware may perform better with different parameters.

First of all, we need to specify that we want to carry out a standard LoRA type training and that we want to do it for an SDXL1.0 model with 1024×1024 images. We must also specify the stable diffusion model on which the LoRA training will be based. It is preferable to train on a basic SDXL1.0 model such as the following model : sdXL_VAEfix.safetensors or sd_xl_base_1.0.safetensors.

Here is a non-exhaustive list of useful parameters to choose from in order to optimise your training and get the best possible quality [128, 129] :

- **Batch size :** The batch size indicates the number of images considered simultaneously during training. The batch size should evenly divide the total number of training images. A larger batch size accelerates training but demands more VRAM. While it enhances training quality, excessively large sizes yield diminishing returns, necessitating careful selection based on GPU capacity. In this instance, a maximum achievable size of 1 was chosen.
- **Epoch :** The epochs represent the number of passes the learning algorithm undertakes across the entire dataset, updating the LoRA based on accumulated information. This concept is closely tied to steps and repeats, where a step involves training the model on an individual image.

  The total number of steps is computed using the formula :

  $$\text{Total steps} = \frac{\text{Number of training images} \times \text{Repeats} \times \text{Epochs} \times \text{Regularization images}}{\text{Batch size}}$$

  where "Regularization images" acts as a multiplier, being 1 when no regularization images are employed and 2 when they are used. After several training sessions, we realized that a maximum of 10 eras was sufficient. More significantly increases training time for equal or worse results. Indeed, the greater the number of steps, the longer the training will last.
- **Learning rate :** The learning rate is an important adjustable value that determines the step size that a machine learning algorithm takes when optimizing to minimize the loss function. The larger it is, the larger the adjustments to the model weights at each iteration, which can lead to rapid convergence, but with the risk of not reaching an optimal solution and jumping around the latter. The smaller it is, the smaller the adjustments, which can lead to a slower

but more stable convergence, often beneficial to achieve an optimal solution without excessive oscillations. The choice often depends on the specific problem and requires empirical adjustments. Most of the time we decided to set this parameter to a value of $1e-4$ as recommended [129].

- **Learning rate scheduler :** The learning rate scheduler adapts the learning rate during training to enhance convergence and performance. For instance, this might enable the model to rapidly grasp the fundamental traits of the data initially, followed by a gradual decrease in the learning rate to identify more intricate details. Various schedulers are available, including constant, constant with warmup, linear, linear with warmup, cosine, and cosine with restarts, providing flexibility in optimizing the learning process. We decide to keep it constant but we could have tested others as cosine.

- **Optimizer :** The optimizer is a crucial component that adjusts the weights of a model during training to minimize the loss function. One can use several such as AdamW or AdamW8bit, commonly used. The latter is faster and saves memory while the former is more accurate. Thus, the best choice is probably the first. However, in view of the VRAM restrictions, we will use AdamW8bit which remains rather precise. Other solutions are also possible like Adafactor, SGDNesterov, DAdapt, Lion, Prodigy, ...

- **Text encoder learning rate :** This represents the learning rate for the Text encoder, and any additional training applied to Text encoders impacts the entire UNet. As a rule, this rate is typically set lower than the learning rate assigned to each UNet block. If a numerical value is specified, it supersedes the learning rate. As for the learning rate, we take back the recommended value which is $5e-5$ [129].

- **UNet learning rate :** This represents the learning rate used for extra training on individual attention blocks within the UNet. If a numerical value is specified, it supersedes the learning rate. We decide to use as for the others the recommended value which is $1e-4$ [129].

- **Network rank (dimension) :** This denotes the quantity of neurons in the middle layer of the extra small neural network incorporated into the attention blocks of the UNet. A higher count allows for the storage of more information and relevant details about the training target. However, it also implies training a greater number of parameters, leading to larger checkpoint files, increased VRAM usage, and a heightened risk of learning superfluous information (overfitting). This decision holds significant importance. Here we chose to evolve with 128 neurons per layer. We would have liked to try 256 in order to capture even more details but this was not possible. We could also have tried lower values like 64, we would have had less details but gain flexibility. The number of neurons depends on the complexity of the task, the size of the images, the nature of the model, and the material resources available.

- **Network alpha :** This measure was introduced for convenience to prevent weights from being rounded down to zero during the saving process in LoRA. Because of its structure, LoRA tends to diminish the values of neural network weights. If these values become excessively small, they may become practically indistinguishable from zero, essentially signifying a lack of meaningful learning.

  The technique involves maintaining a large actual weight value while consistently weakening the weight during training to create the illusion of a smaller value. The alpha network dictates this weight-weakening rate, with a smaller alpha resulting in a larger stored LoRA weight. The extent to which the weight weakens during usage is calculated by $\frac{\text{Network alpha}}{\text{Network rank}}$

(typically ranging between 0 and 1), closely tied to the Network rank number.

If post-learning accuracy is insufficient, weight data may be reduced to zero, and in such cases, it is advisable to lower the Network alpha value. The default value is 1, aiming to maximize the stored weight value. It is crucial to ensure that the alpha value does not exceed the network's rank value to avoid unintended LoRA behavior.

When setting Network alpha, its impact on the learning rate should also be taken into consideration. If Alpha and Rank are equal, the applied force will be 1, with no impact on the learning rate.

- **Scale weight norms :** It enhances training stability by restricting the weight standard of the network. When employed alongside other LoRAs, it can effectively mitigate LoRA overfitting and enhance overall stability. The suggested value for this parameter is 1.
- **Min SNR Gamma :** In LoRA, learning involves adding noise to the training image, but the stability of the learning depends on the strength of the applied noise and its proximity to the target. The Min SNR Gamma is introduced to compensate for variations, especially when images have little noise. This value, ranging from 0 to 20 with a default of 0 (optimal at 5 according to the source article [130]), attenuates random peaks in training losses, promoting smoother learning without negatively impacting speed or VRAM. We decide to keep the optimal value which is 5.

6.2.4 Testing

For these tests, we mainly played with the value of Network alpha. Its definition is not obvious, some tests allow us to have an overview of what is better to use as a value between 16 and 32. Thus, we have described 2 different prompts to double the examples that you can admire in Figure 6.9. An image every 2 epochs was generated using the same seed and without changing the parameters. We used the LoRA CuteCartoon and the Face detailer, although the latter for these trainings is not mandatory. The "strength_model" was set to 1.2. Here are the prompts used :

Model : dynavisionXLAllInOneStylized

- **Prompt 1 :**
  - *Positive prompt :* (CuteCartoonAF, Cute Cartoon:1.2), (rc_test), photo of a man, holding a book in one hand, looking his book, front view, dubious face, glasses, masterpiece, intricate detail, high quality, <lora:loras_romain_charles_test:0.9>.
  - *Negative prompt :* ugly, disfigured, bad quality, bad eyes, bad nose, bad face, bad hands, text, watermark, blurred, blurry, worse, worst, tiling, deformed, mutated, low quality.
- **Prompt 2 :**
  - *Positive prompt :* (CuteCartoonAF, Cute Cartoon:1.2), (rc_test), photo of a man, full body, (from face:1.2), swimming in the ocean around fish and corals, glasses, (green jersey shorts), masterpiece, intricate detail, high quality, <lora:loras_romain_charles_test:0.9>.
  - *Negative prompt :* ugly, disfigured, bad quality, bad eyes, bad nose, bad face, bad hands, text, watermark, blurred, blurry, worse, worst, tiling, deformed, mutated, low quality.

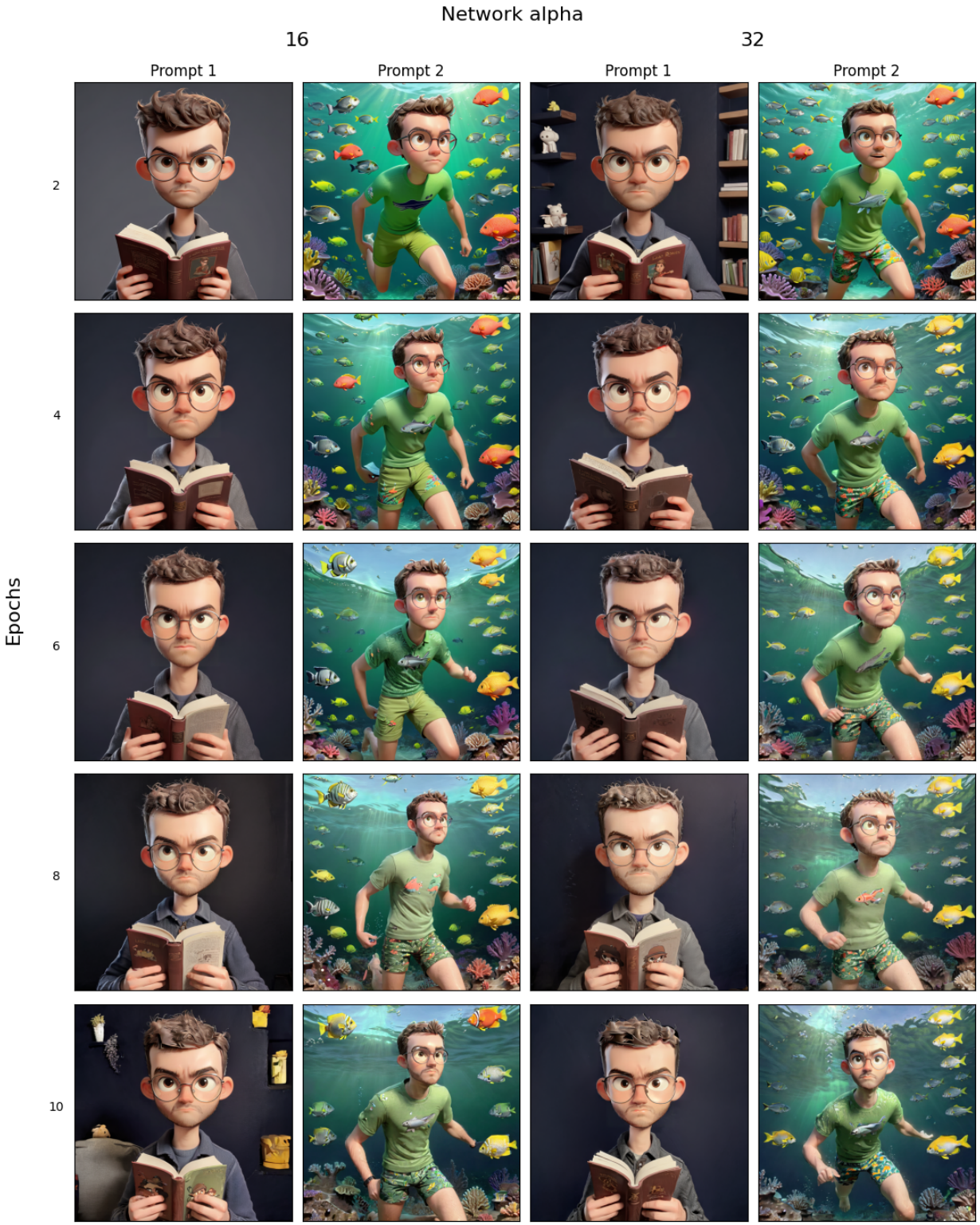


Figure 6.9: Network alpha tests

We can already see that the generations are quite remarkable, and there is consistency among characters. We observe that even for 2 epochs, the results are very good, which is good news from the comic book user's perspective if we were to propose generating their own LoRA. Indeed, approximately 30 minutes to 1 hour would be sufficient under the same conditions, with the same resources. However, we can notice that the quality of the generated images deteriorates quickly, although not catastrophically in all cases. It is true that for prompt 2, with an Network alpha of 32, we see a deterioration from the fourth epoch already. We can see that the model has already learned too

much; it is overfitting, and artifacts appear. A model becomes "overfitted" when it tries to replicate the training images too aggressively, and the results are simply bizarre. We can clearly see this phenomenon in the generation corresponding to an alpha network of 16, with prompt 1 at epoch 10. The background closely resembles that of the training images, and the quality is lower.

Furthermore, we observe that the model cannot make the character look at the book, even when increasing the weight. However, as mentioned in the prompts section, specific keywords can direct the gaze, such as "looks down", and it works, as seen in Figure 6.10. Finally, for the LoRA Romain model to trigger, some trigger words are important, such as "rc_test, glasses, man".

Figure 6.10: Keyword test "looks down"

These tests allow us to choose which training and which epoch seems to be the best. The results obtained with an alpha network of 16 appear to be the best overall, especially for epochs 2 and 4. It is the first of the two that we will choose to create our final short comic strip.

It should be noted that multiple tests have been performed, you have here 2 of the best. Other tests could have been done to really determine the influence of other parameters.

Chapter 7

Final result - Creation of a comic strip

To conclude this research work and these multiple tests, we will now attempt to create a short, consistent comic strip of a few panels using an SDXL model.

Figure 7.1: Final result part 1

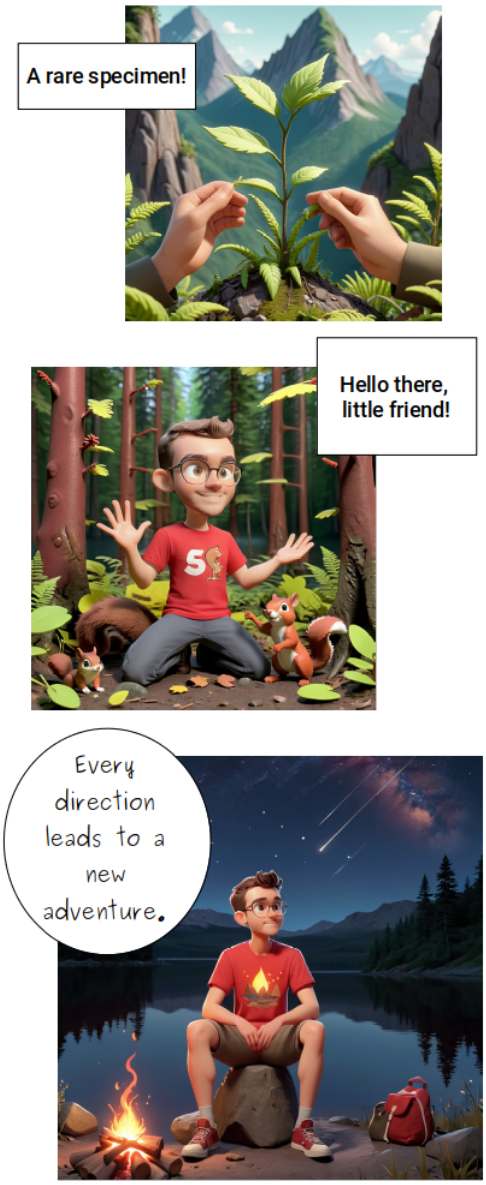


Figure 7.2: Final result part 2

For this test, we used the training set that provided the best compromise between quality, consistency, and flexibility in generations. You can find the script with the parameters used in Figure 11 in the appendice, the epoch used is the second. Subsequent epochs were already starting to overfit. For this creation, we used the CuteCartoon style LoRA and the Face detailer. We used the same seed for the generations to maximize the chances of consistency between the characters. We had to experiment a lot with the LoRAs' strength parameters and prompts to achieve the best possible results. The obtained results are very promising and pave the way for further research. With limited resources, it is possible to obtain convincing results.

Chapter 8

Conclusion and Future works

To conclude our discussion on the research and findings, let's revisit the primary objective of this thesis. The goal of this study was to create an inventive tool allowing authors with limited or no drawing skills to produce comics characterized by a distinct graphic style and a meticulously crafted aesthetic. The aim was to unleash the creativity of these individuals through the utilization of generative AI, facilitating the rapid generation of high-quality images tailored to their narratives. By employing features like prompts, drawings, and image editing, these creators could articulate their artistic vision within minutes. In addition to focusing on image quality, the challenge also involved achieving consistency across characters during the image generation process. Hence, the project aimed to adeptly navigate the limitations inherent in generative artificial intelligence, ultimately striving for a compelling and user-friendly end result.

8.1 Key findings

With that in mind, let's take a look at the various stages involved in arriving at the final result revealed in Chapter 7.

Firstly, in **Chapter 3**, we endeavored to grasp the essence of generative AI as a whole. To comprehend the subtleties of stable diffusion, we delved into various key concepts, including deep learning models in the broad sense, generative models, VAEs (Variational Autoencoders), diffusion models, and latent diffusion models. The latter constitute the core of stable diffusion. As a reminder, stable diffusion is an open-source diffusion model with an efficient image generation process, facilitating numerous applications such as text-to-image, image-to-image generation, inpainting, or outpainting. We also explored the ability to control the influence of prompts on generation and the integration of image control architectures like ControlNet with stable diffusion, enabling generation based on sketches, segmentation maps, or depth information. All these technical concepts are crucial for understanding stable diffusion.

However, despite the theoretical nature of this chapter, we conducted significant tests related to image quality. Specifically, we compared the older SD1.5 models with the new SDXL1.0 models, both in technical aspects and through examples of image generations.

- In terms of technical aspects, we observed that the SDXL1.0 models process images that are twice as large. Moreover, these models have a significantly increased number of parameters compared to the SD1.5 models due to improvements in their UNet and Text encoder architectures. This enhancement is expected to increase the quality of generated images and improve

prompt understanding. However, it also increases complexity and, consequently, the time it takes for images to be generated.

- After the technical comparison of these two versions of stable diffusion, we aimed to determine if the difference in quality was substantial and, more importantly, what the optimal trade-off between quality and generation speed was. The conclusion is that overall, the SDXL models do indeed offer superior image quality compared to the SD1.5 models. The prompt is better respected, and the composition, depth, color palette are more sophisticated, and the images are larger. However, as anticipated, its relative slowness remains its major drawback. This relative slowness is not a significant issue given the superior quality. Therefore, we will continue working with the SDXL1.0 models, especially considering their recent development and the emergence of techniques to reduce image production time.

Finally, a section comparing SDXL with other well-known diffusion models such as DALL-E 3 and MidJourney 5.2 was conducted. We described these models and compared some of their generations. All three models demonstrate high performance, with no significant standout. Consequently, we will continue working with SDXL as the other two models are not open-source, preventing us from achieving our ultimate goal of providing users of the comic creation tool with a minimum level of control.

Once we understood how stable diffusion operates and chose to work with the SDXL1.0 models, in **Chapter 4**, our initial task was to select the ideal user interface for utilizing stable diffusion. We compared the two most popular GUIs currently available : *AUTOMATIC1111* and ComfyUI. Through this comparison, we noticed that ComfyUI offers several advantages in terms of performance speed, flexibility, and transparency of data flow. While it is more complex, this complexity allows for a greater range of choices. Its major drawback is a lack of tools for inpainting. Following this comparison, we opted to continue with ComfyUI, which, in addition to its numerous previously mentioned qualities, allows communication with an API—a particularly useful feature for developing a mobile or web application.

With ComfyUI selected, our next step was to understand how it integrates with stable diffusion. We described a basic workflow for text-to-image using SDXL, clearly showing and explaining all the nodes and parameters involved. Additionally, we introduced additional nodes to discover that other useful applications were possible, such as the Face detailer, enabling the detection and regeneration of faces in images.

Finally, we concluded this chapter by examining what could already be accomplished with the available tools. Consequently, we created three workflows for different applications : text-to-image, text-to-image with ControlNet (Scribble), and inpainting. We aimed to test these flows based on the initial goal of enabling the creation of high-quality and consistent comics. Thus, we attempted to generate images resembling those from a randomly selected webtoon comic. This allowed us to understand how images reacted based on the desired image type (characters and landscapes), prompts, and the selection of certain parameters (cfg, denoise, strength, end_percent), ultimately enabling us to establish fixed parameters. Ultimately, we noted that skillfully crafted prompts could generate high-quality images and maintain consistency. However, achieving the desired variation and uniformity across characters in different panels requires more than just well-crafted prompts. We also found the Face detailer to be useful, as well as the scribble tool, provided parameters were set appropriately for each case. However, we noted a lack of high-quality inpainting tools.

Now that we have effective SDXL workflows capable of producing high-quality images, **Chapter 5** will explore techniques for creating image consistency. Building on the understanding from the previous chapter that prompts can be beneficial, we delved deeper into this concept to grasp all the crucial specifics involved in crafting effective prompts. As a result, we identified key elements to include in a prompt and recognized the significant utility of negative prompts for specifying undesired aspects in our images. Additionally, practical techniques were discovered to emphasize certain keywords in the prompt, such as placing the most important ones at the beginning or assigning varying degrees of importance using parentheses or brackets. Furthermore, the importance of having detailed and specific prompts for obtaining consistent, high-quality images was highlighted. Connecting this insight with the tests conducted in Chapter 4, the results were deemed promising but not sufficient.

In response, we explored alternative solutions, such as fine-tuning methods, enabling additional training of an initial model using a more specific dataset. Various methods, including Hypernetworks, Textual Inversion, Dreambooth, and LoRAs, were considered. Each method's technical workings, along with their respective advantages and disadvantages, were outlined. Ultimately, our focus turned to LoRAs due to their faster training, lower memory consumption, reduced output size, adaptability to multiple models, controllable adaptability, and ease of access requiring fewer GPUs. Consequently, we discovered the ability to train a specific character, a pivotal step in achieving the sought-after consistency. Finally, the integration of LoRAs into ComfyUI was discussed, demonstrating its straightforward implementation.

Ultimately, in **Chapter 6**, recognizing the LoRA method as the most promising, we delved deeper through a series of tests to examine its impact on a model. The objective was to assess the potential for achieving both consistency and flexibility.

- The initial study sought to determine the feasibility of altering the appearance of LoRA-animated characters using prompts while preserving coherence. Analysis of the test-generated images revealed that we achieved coherence and could subtly adjust the characters' appearance through prompt manipulation. However, obtaining an effective starting prompt was crucial, and clothing consistency posed challenges. Additionally, the model's understanding of colors, particularly in complex scenarios, was not flawless, emphasizing the dependence of these tests on the quality of LoRA training.
- Building on the positive outcomes, subsequent experiments continued with LoRAs, aiming to evaluate their capacity to generate cartoon-style images when trained on real people's photos. This approach, leveraging a dataset of real images transformable into various styles, demonstrated the diffusion model's ability to produce coherent cartoon images. The usage of a LoRA trained in a specific style further emphasized the potential for style associations, underscoring the importance of well-trained LoRAs.
- Concluding the series, the final tests explored whether prompt-driven modifications remained viable for real-person LoRAs with the application of a cartoon-style LoRA. These tests provided conclusive results.

In summary, these analyses affirmed the appropriateness of selecting the LoRA method, offering a multitude of possibilities while maintaining both consistency and flexibility.

Subsequently, our focus shifted to training a personalized LoRA using real individuals' photos. This involved employing Kohya-ss's scripts for SDXL LoRA training and collecting training images linked

with captions adhering to various quality criteria. Despite a minimal dataset of only 30 images, a comprehensive exploration of crucial training parameters, including epochs, learning rates, optimizers, and others, paved the way for several successful training sessions. Notably, the significance of parameters like Network rank and Network alpha, along with the justified use of the face detailer, became apparent. The impressive final results underscored the potential of the LoRA model, even with limited GPU resources.

8.2 Implications

The approach undertaken in this thesis has resulted in the findings presented in **Chapter 7**. The generated comic panels vividly illustrate the achievement of both consistency and high-quality images. The initially outlined objectives have, for the most part, been met. As a result, Deuse Company can utilize this work for the development of its application, with a commitment to refining it by taking into account the identified limitations and the suggestions for future research that will be discussed in the upcoming section.

8.3 Limitations and Future research directions

While the outcomes show promise, the realm of generative AI is still in its early stages, with continuous advancements offering the potential to overcome the limitations identified in this thesis. Several points are worth considering :

- Initially, the prolonged image generation time of SDXL models compared to its predecessor, SD1.5, is a drawback. Yet, ongoing efforts like SDXL Turbo [57] or LCM-LoRA [58] present possibilities to reduce computation time, signifying avenues for exploration and emphasizing the importance of understanding and utilizing SDXL models presently.
- A notable limitation is the lack of tests for the SDXL workflows in inpainting, an aspect that could be a specific focus for future endeavors.
- Regarding techniques for achieving consistency, only prompts and fine-tuning methods like LoRAs are mentioned. However, alternative ideas, such as the Roop technique or Fast face swap, could have been explored. Supplementing this research with these ideas might uncover optimal choices or combinations for achieving consistency.
- The section on LoRA training could have been more comprehensive, exploring various parameters, incorporating more training images, or employing regularization images. Resource limitations, particularly VRAM, constrained the depth of these investigations.

Looking ahead, beyond these limitations, several future work ideas emerge :

- Considering the success of consistency tests in Chapter 6, a compelling prospect is creating a database of LoRAs featuring real people with diverse characteristics. A subsequent project could involve employing AI to match user-entered keywords describing physical characteristics with LoRAs possessing similar attributes, enabling the creation of unique characters by combining features from different LoRA characters.
- The issue of image composition, crucial in comic panels, was not addressed in this research. Techniques like Scribble, Openpose, Regional prompter, Area composition, and Attention couple might offer assistance, though further exploration is warranted.

Bibliography

[1] Stable Tom. Stable Diffusion, c'est quoi ? — stablediffusion.blog. https://www.stablediffusion.blog/introduction-stable-diffusion, 2023.

[2] Andrew. Beginner's Guide to ComfyUI - Stable Diffusion Art. https://stable-diffusion-art.com/comfyui/, 2024.

[3] Yubin. Beginner's guide to comfyui for stable diffusion - aituts. https://aituts.com/comfyui/, 2023.

[4] Comfyanonymous. GitHub - comfyanonymous/ComfyUI: The most powerful and modular stable diffusion GUI, api and backend with a graph/nodes interface. https://github.com/comfyanonymous/ComfyUI.

[5] Plates-formes de GPU | Documentation Compute Engine | Google Cloud. https://cloud.google.com/compute/docs/gpus?hl=fr, 2018.

[6] DEUSE : Experts en Ingénieurie Informatique. https://www.deuse.be/fr/.

[7] Odevio, Sending your app to the moon. https://www.odevio.com/.

[8] Stripik. https://stripik.com/.

[9] Django (framework) — Wikipédia. https://fr.wikipedia.org/wiki/Django_(framework).

[10] Flutter (logiciel) — Wikipédia. https://fr.wikipedia.org/wiki/Flutter_(logiciel).

[11] React — Wikipédia. https://fr.wikipedia.org/wiki/React.

[12] Docker (logiciel) — Wikipédia. https://fr.wikipedia.org/wiki/Docker_(logiciel).

[13] Sovit Rath Vaibhav Singh. Diffusion Models for Image Generation – A Comprehensive Guide. https://learnopencv.com/image-generation-using-diffusion-models/, 2023.

[14] deeplizard. Intro to Latent Diffusion Models - Stable Diffusion Masterclass. https://www.youtube.com/watch?v=IxHXQpk5kkg, IntrotoLatentDiffusionModels-StableDiffusionMasterclass, 2023.

[15] Jojo John Moolayil. A Layman's Guide to Deep Neural Networks. https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb, 2019.

[16] What is Generative AI? | NVIDIA. https://www.nvidia.com/en-us/glossary/data-science/generative-ai/.

[17] Prakash Pandey. Deep Generative Models. https://towardsdatascience.com/deep-generative-models-25ab2821afd3, 2018.

[18] Kinza Yasar. What is Generative Modeling? |Definition from TechTarget. https://www.techtarget.com/searchenterpriseai/definition/generative-modeling, 2023.

[19] Abid Ali Awan. What is a Generative Model? https://www.datacamp.com/blog/what-is-a-generative-model, 2023.

[20] Yzy1996. GitHub - yzy1996/Awesome-Generative-Model: A collection of resources on Generative Model. https://github.com/yzy1996/Awesome-Generative-Model.

[21] Improving Diffusion Models as an Alternative To GANs, Part 1 | NVIDIA Technical Blog. https://developer.nvidia.com/blog/improving-diffusion-models-as-an-alternative-to-gans-part-1/.

[22] Improving Diffusion Models as an Alternative To GANs, Part 2 | NVIDIA Technical Blog. https://developer.nvidia.com/blog/improving-diffusion-models-as-an-alternative-to-gans-part-2/.

[23] glouppe. INFO8010: Deep Learning. https://glouppe.github.io/info8010-deep-learning/?p=lecture11.md.

[24] Arxiv Insights. Variational Autoencoders. https://www.youtube.com/watch?v=9zKuYvjFFS8,VariationalAutoencoders-ArxivInsights, 2019.

[25] Shubham Patel. All you need to know about Variational AutoEncoder. https://blog.bayeslabs.co/2019/06/04/All-you-need-to-know-about-Vae.html, 2019.

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[27] Lilian Weng. What are Diffusion Models? https://lilianweng.github.io/posts/2021-07-11-diffusion-models/, 2021.

[28] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[29] Steins. Stable Diffusion Clearly Explained! https://medium.com/@steinsfu/stable-diffusion-clearly-explained-ed008044e07e, 2023.

[30] Andrew. How does Stable Diffusion work? https://stable-diffusion-art.com/how-stable-diffusion-work/?utm_source=pocket_saves, 2024.

[31] glouppe. INFO8010: Deep Learning. https://glouppe.github.io/info8010-deep-learning/?p=lecture12.md.

[32] Subhradip Roy. A Beginner's Guide to Diffusion Models: Understanding the Basics and Beyond. https://roysubhradip.hashnode.dev/a-beginners-guide-to-diffusion-models-understanding-the-basics-and 2023.

[33] Jeremy Zhang. UNet Line by Line Explanation. https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5, 2019.

[34] Katherine (Yi) Li. Vanishing and exploding gradients in neural network models: Debugging, monitoring, and fixing. https://neptune.ai/blog/vanishing-and-exploding-gradients-debugging-monitoring-fixing, 2023.

[35] Vanishing Gradient Problem | BotPenguin. https://botpenguin.com/glossary/vanishing-gradient-problem.

[36] Nikolas Adaloglou. Intuitive Explanation of Skip Connections in Deep Learning | AI Summer. https://theaisummer.com/skip-connections/, 2020.

[37] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[38] PhD J. Rafid Siddiqui. What are Stable Diffusion Models and Why are they a Step Forward for Image Generation? https://towardsdatascience.com/what-are-stable-diffusion-models-and-why-are-they-a-step-forward-f 2022.

[39] Aayush Agrawal. Stable diffusion using Hugging Face. https://towardsdatascience.com/stable-diffusion-using-hugging-face-501d8db 2022.

[40] J Alammar. The illustrated stable diffusion, 2022.

[41] Anel Islamovic. Stability AI launches SDXL 0.9: A Leap Forward in AI Image Generation — Stability AI. https://stability.ai/news/sdxl-09-stable-diffusion, 2023.

[42] Jonathan Whitaker. Mid-U Guidance: Fast Classifier Guidance for Latent Diffusion Models. https://wandb.ai/johnowhitaker/midu-guidance/reports/Mid-U-Guidance-Fast-Classifier-Guidance-for-Latent-Diffusion-Model 2023.

[43] Isamu Isozaki. Understanding Prompt To Prompt for Editing Images using Diffusion Models with the Source Code Part.... https://isamu-website.medium.com/understanding-prompt-to-prompt-for-editing-images-using-diffusion- 2023.

[44] Andrew. How does negative prompt work? - Stable Diffusion Art. https://stable-diffusion-art.com/how-negative-prompt-work/, 2023.

[45] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[46] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[47] Susung Hong, Gyuseong Lee, Wooseok Jang, and Seungryong Kim. Improving sample quality of diffusion models using self-attention guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7462–7471, 2023.

[48] Sander Dieleman. Guidance: a cheat code for diffusion models, 2022.

[49] Gabriel Mongaras. Diffusion Models — DDPMs, DDIMs, and Classifier Free Guidance. https://betterprogramming.pub/diffusion-models-ddpms-ddims-and-classifier-free-guidance-e07b297b 2023.

[50] Stable Diffusion - Wikipedia. https://en.wikipedia.org/wiki/Stable_Diffusion.

[51] Narindra R. Stable Diffusion : tout savoir en 5 minutes. https://intelligence-artificielle.com/stable-diffusion/, 2023.

[52] aws samples. https://github.com/aws-samples/amazon-bedrock-workshop/blob/main/05_Image/.

[53] Andrew. Stable Diffusion XL 1.0 model - Stable Diffusion Art. https://stable-diffusion-art.com/sdxl-model/, 2023.

[54] Stable Tom. Comment Fonctionne Stable Diffusion ? https://www.stablediffusion.blog/comment-fonctionne-stable-diffusion, 2023.

[55] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Muller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[56] Ertuğrul Demir. The Arrival of SDXL 1.0. https://towardsdatascience.com/the-arrival-of-sdxl-1-0-4e739d5cc6c7, 2023.

[57] Design Team. Introducing SDXL Turbo: A Real-Time Text-to-Image Generation Model — Stability AI. https://stability.ai/news/stability-ai-sdxl-turbo, 2023.

[58] Simian Luo, Yiqin Tan, Suraj Patil, Daniel Gu, Patrick von Platen, Apolinário Passos, Longbo Huang, Jian Li, and Hang Zhao. Lcm-lora: A universal stable-diffusion acceleration module. *arXiv preprint arXiv:2311.05556*, 2023.

[59] Jim Clyde Monge. Dall-E 3 VS MidJourney 5.2 VS Stable XL — Same Prompt, Different Results. https://generativeai.pub/dall-e-3-vs-midjourney-5-2-vs-stable-xl-same-prompt-different-resu 2023.

[60] https://www.facebook.com/profile.php?id=100088971607226. DALL·E 3 vs Stable Diffusion XL: A comparison - Stable Diffusion Art. https://stable-diffusion-art.com/dalle3-vs-stable-diffusion-xl/, 2023.

[61] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2023.

[62] Georgia Weston. What is Midjourney AI and how does it work? https://101blockchains.com/midjourney-ai-tutorial/, 2023.

[63] Raktim Bora. Controlling Stable Diffusion image generation with ControlNet. https://medium.com/@reachraktim/controlling-stable-diffusion-image-generation-with-controlnet-3643 2023.

[64] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[65] Andrew. ControlNet v1.1: A complete guide - Stable Diffusion Art. https://stable-diffusion-art.com/controlnet/, 2023.

[66] Steins. Stable Diffusion — ControlNet Clearly Explained! https://medium.com/@steinsfu/stable-diffusion-controlnet-clearly-explained-f86092b62c89, 2023.

[67] Yubin. Beginner's guide to comfyui for stable diffusion - aituts. https://aituts.com/comfyui/, 2023.

[68] Tout ce qu'il faut savoir sur les modèles Stable Diffusion. https://www.stablediffusion.blog/modeles-guide, 2023.

[69] Load Checkpoint - ComfyUI Community Manual. https://blenderneko.github.io/ComfyUI-docs/Core%20Nodes/Loaders/LoadCheckpoint/.

[70] CLIP Text Encode (Prompt) - ComfyUI Community Manual. https://blenderneko.github.io/ComfyUI-docs/Core%20Nodes/Conditioning/CLIPTextEncode/.

[71] C.Y. Wong. Two Text Prompts (Text Encoders) in SDXL 1.0 | myByways. https://mybyways.com/blog/two-text-prompts-text-encoders-in-sdxl-1-0, 2023.

[72] Empty Latent Image - ComfyUI Community Manual. https://blenderneko.github.io/ComfyUI-docs/Core%20Nodes/Latent/EmptyLatentImage/.

[73] KSampler - ComfyUI Community Manual. https://blenderneko.github.io/ComfyUI-docs/Core%20Nodes/Sampling/KSampler/.

[74] Andy H. Tu. Comprehensive Guide for the ComfyUI User Interface. https://www.andyhtu.com/post/comprehensive-guide-for-the-comfyui-user-interface, 2023.

[75] Ltdrdata. GitHub - ltdrdata/ComfyUI-Manager. https://github.com/ltdrdata/ComfyUI-Manager.

[76] Ltdrdata. GitHub - ltdrdata/ComfyUI-Impact-Pack. https://github.com/ltdrdata/ComfyUI-Impact-Pack.

[77] Rohit Kundu. YOLO Algorithm for Object Detection Explained [+Examples]. https://www.v7labs.com/blog/yolo-object-detection, 2023.

[78] Kevin Rexis Velasco. YOLO (You Only Look Once). https://towardsdatascience.com/yolo-you-only-look-once-17f9280a47b0, 2019.

[79] glenn-jocher AyushExel. Home. https://docs.ultralytics.com/, 2023.

[80] Ltdrdata. ComfyUI-extension-tutorials/ComfyUI-Impact-Pack/tutorial/detectors.md at Main · ltdrdata/ComfyUI-extension-tutorials. https://github.com/ltdrdata/ComfyUI-extension-tutorials/blob/Main/ComfyUI-Impact-Pack/tutorial/detectors.md.

[81] Nikolaj Buhl. Meta AI's New Breakthrough: Segment Anything Model (SAM) Explained. https://encord.com/blog/segment-anything-model-explained/, 2023.

[82] Segment Anything. https://segment-anything.com/.

[83] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[84] Peglo. To The Stars and Back - Episode 1. https://www.webtoons.com/en/slice-of-life/to-the-stars-and-back/episode-1/viewer?title_no=4047&episode_no=1.

[85] Andrew. Stable Diffusion Samplers: A Comprehensive Guide - Stable Diffusion Art. https://stable-diffusion-art.com/samplers/, 2023.

[86] Load Image - ComfyUI Community Manual — blenderneko.github.io. https://blenderneko.github.io/ComfyUI-docs/Core%20Nodes/Image/LoadImage/.

[87] Invert Image - ComfyUI Community Manual. https://blenderneko.github.io/ComfyUI-docs/Core%20Nodes/Image/InvertImage/.

[88] Load ControlNet Model - ComfyUI Community Manual. https://blenderneko.github.io/ComfyUI-docs/Core%20Nodes/Loaders/LoadControlNet/.

[89] Chong Mou, Xintao Wang, Liangbin Xie, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *arXiv preprint arXiv:2302.08453*, 2023.

[90] Apply ControlNet - ComfyUI Community Manual — blenderneko.github.io. https://blenderneko.github.io/ComfyUI-docs/Core%20Nodes/Conditioning/ApplyControlNet/.

[91] Anthony Fu. Stable Diffusion QR Code 101. https://antfu.me/posts/ai-qrcode-101.

[92] VAE Encode (for Inpainting) - ComfyUI Community Manual. https://blenderneko.github.io/ComfyUI-docs/Core%20Nodes/Latent/inpaint/VAEEncodeForInpainting/.

[93] Repeat Latent Batch - ComfyUI Community Manual. https://blenderneko.github.io/ComfyUI-docs/Core%20Nodes/Latent/batch/RepeatLatentBatch/.

[94] Javi Lopez. Consistency of Characters, Objects, and Styles with Generative AI. https://www.linkedin.com/pulse/consistency-characters-objects-styles-generative-ai-javi-lopez, 2023.

[95] Igor Nowacki. Creating Consistent Characters in Stable Diffusion AI Art. https://ts2.space/en/creating-consistent-characters-in-stable-diffusion-ai-art-2/, 2023.

[96] Understanding Character Consistency in AI Comic Factory. https://aicomicfactory.com/blog/character-consitency.

[97] Andrew. Stable Diffusion prompt: a definitive guide - Stable Diffusion Art. https://stable-diffusion-art.com/prompt-guide/?utm_source=pocket_saves, 2024.

[98] Andrew. 15 Stable Diffusion XL prompts + tips - Stable Diffusion Art. https://stable-diffusion-art.com/sdxl-prompts/?utm_source=pocket_saves, 2023.

[99] vera. Stable Diffusion Prompt Guide - Promptify. https://promptify.cc/stable-diffusion-prompt-guide/, 2023.

[100] Umberto Grando. Stable Diffusion Ultimate Guide pt. 2: Prompting. https://medium.com/@inzaniak/stable-diffusion-ultimate-guide-pt-2-prompting-35eacb3dc5f4, 2023.

[101] Prompt weighting. https://huggingface.co/docs/diffusers/using-diffusers/weighted_prompts.

[102] Andrew. How to use Stable Diffusion - Stable Diffusion Art. https://stable-diffusion-art.com/beginners-guide/, 2023.

[103] Andrew. How to come up with good prompts for Stable Diffusion - Stable Diffusion Art. https://stable-diffusion-art.com/how-to-come-up-with-good-prompts-for-ai-image-generation/, 2023.

[104] Andrew. Stable Diffusion Models: a beginner's guide - Stable Diffusion Art. https://stable-diffusion-art.com/models/, 2023.

[105] Andrew. What are hypernetworks and the ones you should know - Stable Diffusion Art. https://stable-diffusion-art.com/hypernetwork/, 2023.

[106] koiboi. Demystifying LoRA, Dreambooth, Textual Inversion, and Hypernetworks. https://www.toolify.ai/ai-news/demystifying-lora-dreambooth-textual-inversion-and-hypernetworks-2 2023.

[107] koiboi. LoRA vs Dreambooth vs Textual Inversion vs Hypernetworks. https://www.youtube.com/watch?v=dVjMiJsuR5o, 2023.

[108] Andrew. How to use embeddings in Stable Diffusion - Stable Diffusion Art. https://stable-diffusion-art.com/embedding/, 2023.

[109] Onkar Mishra. Textual Inversion: A method to finetune Stable Diffusion Model. https://medium.com/@onkarmishra/how-textual-inversion-works-and-its-applications-5e3fda4aa0bc, 2023.

[110] DreamBooth. https://huggingface.co/docs/diffusers/training/dreambooth.

[111] Yubin. Beginner's guide to training/fine-tuning stable diffusion models. https://aituts.com/beginners-guide-to-training-fine-tuning-stable-diffusio 2023.

[112] Yubin. Stable diffusion lora models: A complete guide (best ones, installation, training) - aituts. https://aituts.com/stable-diffusion-lora/, 2023.

[113] Cloneofsimo. GitHub - cloneofsimo/lora: Using Low-rank adaptation to quickly fine-tune diffusion models. https://github.com/cloneofsimo/lora.

[114] Sayak Paul Pedro Cuenca. Using LoRA for Efficient Stable Diffusion Fine-Tuning. https://huggingface.co/blog/lora, 2023.

[115] anotherjesse cloneofsimo, andreasjansson and zeke. Introducing LoRA: A faster way to fine-tune Stable Diffusion. https://replicate.com/blog/lora-faster-fine-tuning-of-stable-diffusion, 2023.

[116] Stable Tom. Les LoRAs Stable Diffusion et leur utilisation. https://www.stablediffusion.blog/lora-stablediffusion, 2023.

[117] Andrew. What are LoRA models and how to use them in AUTOMATIC1111 - Stable Diffusion Art. https://stable-diffusion-art.com/lora/, 2023.

[118] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[119] Bhavin Jawade. Understanding LoRA — Low Rank Adaptation For Finetuning Large Models. https://towardsdatascience.com/understanding-lora-low-rank-adaptation-for-finetuning-large-models 2023.

[120] Bào Bùi. LoRA Keras implementation for fine-tuning Stable Diffusion. https://medium.com/@bobi_29852/lora-keras-implementation-for-fine-tuning-stable-diffusion-1c3318d 2023.

[121] Low-Rank Adaptation of Large Language Models (LoRA). https://huggingface.co/docs/diffusers/v0.17.1/training/lora.

[122] Load LoRA - ComfyUI Community Manual. https://blenderneko.github.io/ComfyUI-docs/Core%20Nodes/Loaders/LoadLoRA/.

[123] comfyanonymous. Lora Examples. https://comfyanonymous.github.io/ComfyUI_examples/lora/.

[124] Yubin. How to train an sdxl lora (koyha with runpod). https://aituts.com/sdxl-lora/, 2023.

[125] Kohya-ss. GitHub - kohya-ss/sd-scripts. https://github.com/kohya-ss/sd-scripts.

[126] Bmaltais. GitHub - bmaltais/kohya_ss. https://github.com/bmaltais/kohya_ss.

[127] THE OTHER LoRA TRAINING RENTRY. https://rentry.org/59xed3, 2023.

[128] Yubin. Stable diffusion lora training settings for koyha ss, explained. https://aituts.com/lora-training-settings/, 2023.

[129] bmaltais. LoRA training parameters. https://github.com/bmaltais/kohya_ss/wiki/LoRA-training-parameters.

[130] Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via min-snr weighting strategy. *arXiv preprint arXiv:2303.09556*, 2023.

Appendix

Test 5 - Real person (close-up face) :

- *Positive prompt :* realistic photography, angry man, close-up face, looks at the camera, blond beard, blond hair, black eyes, green t-shirt, on a farm, intricate detail, masterpiece, high quality.
- *Negative prompt :* ugly, disfigured, bad hands, bad eyes, poor face, deformed, blurry, text.

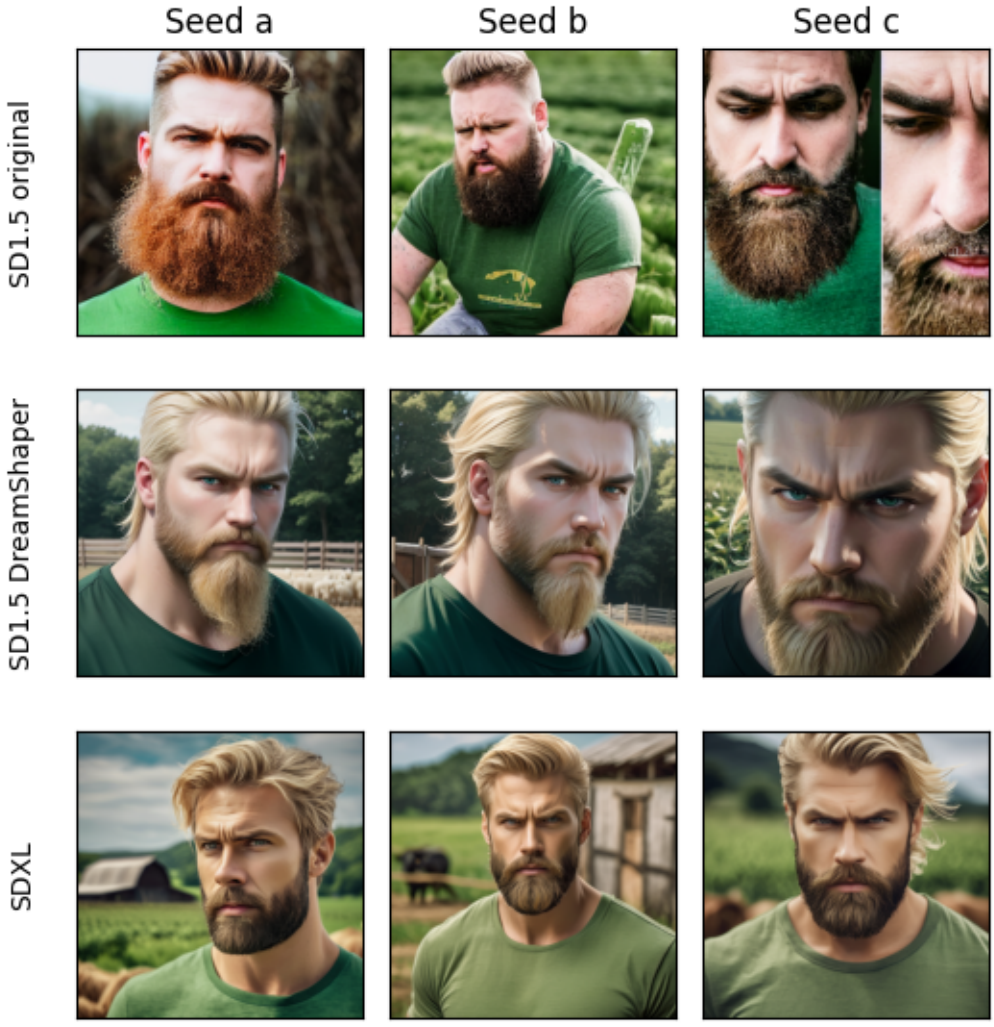


Figure 1: Test 5 - SD1.5 vs SDXL1.0 - Real person generation (close-up face)

Test 6 - Cartoon character :

- *Positive prompt :* cartoon style, a boy eating a burger on a bench in a garden, black hair, black eyes, intricate detail, masterpiece, high quality.
- *Negative prompt :* ugly, disfigured, bad hands, bad eyes, poor face, deformed, blurry, text.

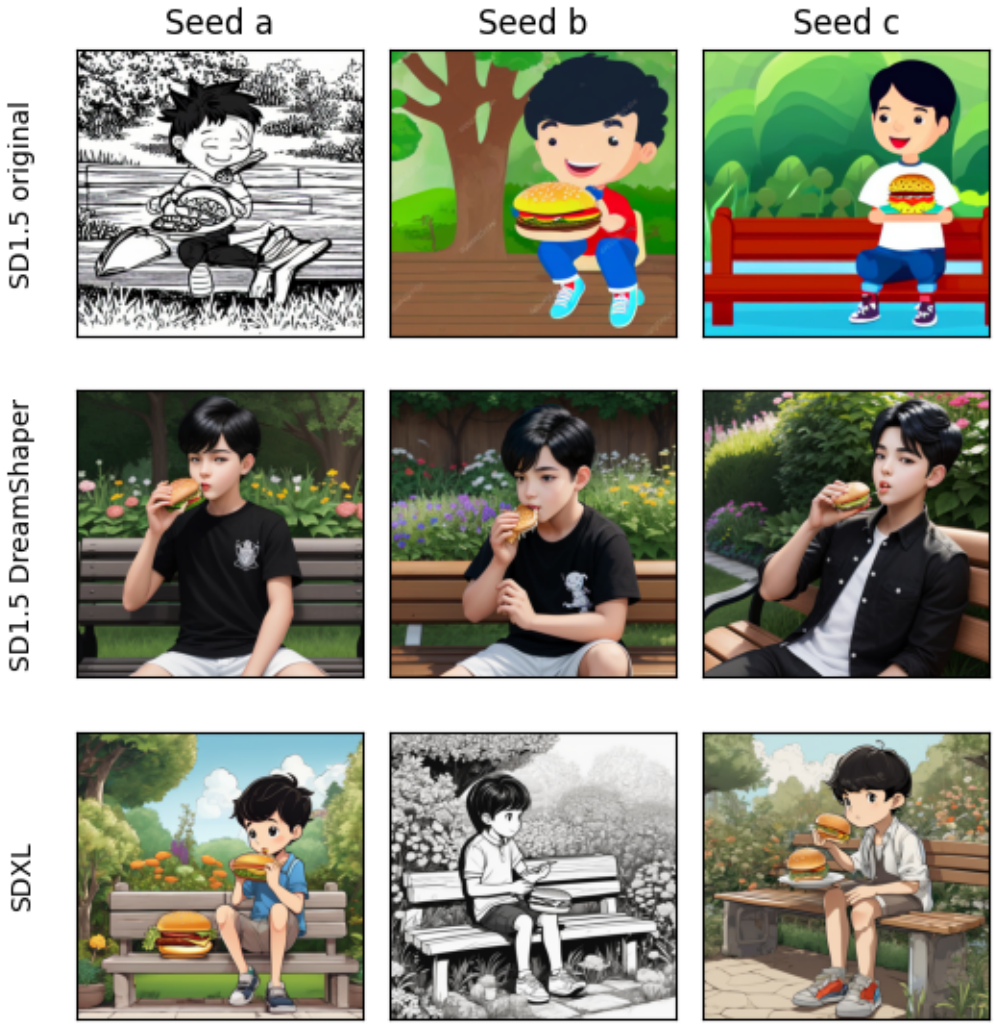


Figure 2: Test 6 - SD1.5 vs SDXL1.0 - Cartoon character generation

Figure 3: Study of consistency and modifications of LoRAs via prompts - Makima LoRA

Figure 4: Study of consistency and modifications of LoRAs via prompts - Gojou Satoru

Figure 5: Modification : blue hair and green eyes - Seed a - Choi Kang-rim LoRA

Figure 6: Modification : no hair - Seed a - Choi Kang-rim LoRA

Figure 7: Modification : fat boy, blue hair and green eyes - Seed a - Choi Kang-rim LoRA

Figure 8: Modification : blue hair, green eyes and orange headband - Seed a - Choi Kang-rim LoRA

Figure 9: Modification : very long black hair - Seed a - Makima LoRA

Figure 10: Study of consistency and modifications of LoRAs via prompts with real-person LoRA - Jason Momoa LoRA

```
NETWORK_DIM=128
NETWORK_ALPHA=16
TEXT_LR=5e-5
UNET_LR=1e-4
LR=1e-4
SCHEDULER="constant"
BATCH_SIZE=1

cd sd-scripts
source venv/bin/activate

accelerate launch --num_cpu_threads_per_process=2 "./sdxl_train_network.py" --pretrained_model_name_or_path="/home/tchavet/stable-diffusion-webui/models/Stable-diffusion/sd_xl_base_1.0.safetensors" --
train_data_dir="/home/tchavet/$BASE_DIR/img" --resolution="1024,1024" --output_dir="/home/tchavet/$BASE_DIR/model" --logging_dir="/home/tchavet/$BASE_DIR/log" --network_alpha=$NETWORK_ALPHA --
save_model_as=safetensors --network_module=networks.lora --text_encoder_lr=$TEXT_LR --unet_lr=$UNET_LR --network_dim=$NETWORK_DIM --output_name="$1_$2" --lr_scheduler_num_cycles="10" --no_half_vae --
learning_rate=$LR --lr_scheduler=$SCHEDULER --train_batch_size=$BATCH_SIZE --max_train_steps="6000" --save_every_n_epochs="2" --mixed_precision="fp16" --save_precision="fp16" --cache_latents --
cache_latents_to_disk --optimizer_type="AdamW8bit" --max_data_loader_n_workers="0" --bucket_reso_steps=64 --gradient_checkpointing --xformers --bucket_no_upscale --noise_offset=0.0 --lowram --
output_name="romain_charles_test_16" --caption_extension=".txt" --seed=1234 --network_train_unet_only --cache_text_encoder_outputs --min_snr_gamma=5 --scale_weight_norms=1 --network_dropout=0.3
```

Figure 11: Best training script