

ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM APRENDIZAGEM DE MÁQUINAS PROFUNDAS

JORDAN KOPPER

SDXL DEBIASING MECHANISM: A FAIRNESS IMPROVEMENT TO IMAGE GENERATION
THROUGH DIFFUSION KERNELS

Porto Alegre
2024

PÓS-GRADUAÇÃO - *STRICTO SENSU*



Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

**SDXL DEBIASING
MECHANISM: A FAIRNESS
IMPROVEMENT TO IMAGE
GENERATION THROUGH
DIFFUSION KERNELS**

JORDAN KOPPER

Thesis submitted to the Pontifical Catholic
University of Rio Grande do Sul in partial
fulfillment of the requirements for the
degree of Master in Computer Science.

Advisor: Prof. Rodrigo C. Barros

Porto Alegre
2024

Ficha Catalográfica

K83s Kopper, Jordan

SDXL debiasing mechanism : a fairness improvement to image generation through diffusion kernels / Jordan Kopper. – 2024.

77 p.

Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Rodrigo Coelho Barros.

1. Diffusion Models. 2. Stable Diffusion. 3. Fairness. 4. Bias Mitigation. I. Barros, Rodrigo Coelho. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).

Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

Jordan Kopper

**SDXL DEBIASING MECHANISM: A FAIRNESS IMPROVEMENT
TO IMAGE GENERATION THROUGH DIFFUSION KERNELS**

This Master Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor/Master of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on March 27, 2024.

COMMITTEE MEMBERS:

Prof. Dr. Duncan Dubugras Alcoba Ruiz (PPGCC/PUCRS)

Prof. Dr. Cristiano André da Costa (PPGCC/Unisinos)

Prof. Dr. Rodrigo Coelho Barros (PPGCC/PUCRS - Advisor)

SDXL DEBIASING MECHANISM: A FAIRNESS IMPROVEMENT TO IMAGE GENERATION THROUGH DIFFUSION KERNELS

RESUMO

Esta dissertação explora a relevante questão dos vieses indesejados em modelos de geração texto-para-imagem, focando especialmente em mecanismos de redução de vies dentro de modelos como o Stable Diffusion, especialmente com o lançamento do Stable Diffusion eXtra Large (SDXL). Ao propor uma abordagem centrada na modificação do Classifier-Free Guidance, o estudo visa direcionar a geração de imagens de modo a se afastar de vieses sociais prejudiciais, preservando ao mesmo tempo a fidelidade da imagem e o poder de geração do modelo original. Aproveitando contribuições de tentativas anteriores de redução de vies, esta pesquisa utiliza os mecanismos internos dos Modelos de Difusão para desenviezar conceitos prejudiciais diretamente no cerne do modelo. Assim, contribui para um uso mais equitativo e responsável de sistemas de IA geradora. Os resultados demonstram que o método proposto mitiga efetivamente os vieses sem comprometer a qualidade da imagem, ao custo de um aumento no tempo de inferência. Apesar das limitações atuais, este trabalho representa um passo importante em direção a uma geração de imagens mais justa e destaca a importância das considerações éticas no desenvolvimento de IA.

Palavras-Chave: Modelos Difusores, Stable Diffusion, Fairness, Redução de Viés.

SDXL DEBIASING MECHANISM: A FAIRNESS IMPROVEMENT TO IMAGE GENERATION THROUGH DIFFUSION KERNELS

ABSTRACT

This thesis explores the pressing issue of unwanted biases in text-to-image synthesis models, focusing particularly on debiasing mechanisms within Stable Diffusion models, notably the Stable Diffusion eXtra Large (SDXL) release. By proposing an approach centered on modifying Classifier-Free Guidance, the study aims to steer image generation away from harmful societal biases while preserving image fidelity and full capabilities of SDXL model. Leveraging insights from previous debiasing attempts, this research pioneers the usage of inner workings of Diffusion Models to debias harmful concepts at its core. Thereby, it contributes to a more equitable and responsible usage of generative AI systems. Results demonstrate that the proposed method effectively mitigates biases without compromising image quality, at the cost of an increase in inference time. Despite current limitations, this work represents a crucial step towards fairer image generation and underscores the importance of ethical considerations in AI development.

Keywords: Diffusion Models, Stable Diffusion, Fairness, Bias Mitigation.

LIST OF FIGURES

Figure 2.1 – Diffusion process with cosine schedule. Source: [Nichol and Dhariwal, 2021].	31
Figure 2.2 – Generative learning trilemma. Source: [Vahdat et al., 2021].	32
Figure 2.3 – Sliced Score Matching (SSM) Loss. Source: [Song and Ermon, 2019]. . .	34
Figure 2.4 – Denoising process from diffusion model. Source: [Ho et al., 2020].	35
Figure 2.5 – Interpolating DDIM samples. Source: [Song et al., 2020].	37
Figure 2.6 – Inference on Stable Diffusion v1.5.	38
Figure 2.7 – Inference on Stable Diffusion eXtra Large.	40
Figure 2.8 – Classifier-Free Guidance on SDXL.	40
Figure 2.9 – Denoising Process with Classifier-Free Guidance on SDXL.	41
Figure 3.1 – Safety Concept Guidance. Source: [Schramowski et al., 2023].	44
Figure 4.1 – SDXL Preprocessing Graph.	47
Figure 4.2 – SDXL Denoising Loop Graph.	48
Figure 4.3 – Added Text Latent from expanded Embeddings.	49
Figure 4.4 – Modified Classifier-Free Guidance with Bias Concept.	50
Figure 4.5 – Simplified Classifier-Free Guidance with Bias Concept on Data Space. .	51
Figure 4.6 – Generated images with our method.	52
Figure 4.7 – User Interface for Image Annotation.	53
Figure 5.1 – Samples of the generated grid of images.	62
Figure 5.2 – Images generated from each method using the same seed.	62
Figure 5.3 – Average of all 200 images in pixel space from each method.	64
Figure 6.1 – Semantic changes between original SDXL and our method for firefighter concept.	68
Figure 6.2 – Semantic changes between original SDXL and our method for nurse concept.	69
Figure 6.3 – Semantic changes between original SDXL and our method for Business Leader concept.	69
Figure A.1 – Samples of the generated grid of images.	77

LIST OF TABLES

Table 4.1 – Hyperparameters of SDXL & debiased SDXL.	52
Table 5.1 – Original SDXL vs. our method: Genders on firefighter prompt.	60
Table 5.2 – Original SDXL vs. our method: Ethnicity on firefighter prompt.	60
Table 5.3 – Original SDXL vs. our method: Apparent age on firefighter prompt.	60
Table 5.4 – Original SDXL vs. our method: Genders on nurse prompt.	60
Table 5.5 – Original SDXL vs. our method: Ethnicity on nurse prompt.	61
Table 5.6 – Original SDXL vs. our method: Apparent age on nurse prompt.	61
Table 5.7 – Original SDXL vs. our method: Gender on business leader prompt.	61
Table 5.8 – Original SDXL vs. our method: Ethnicity on business leader prompt.	61
Table 5.9 – Original SDXL vs. our method: Apparent age on business leader prompt.	61
Table 5.10 – Average and Standard Deviation of Individual Similarities	63
Table 5.11 – Cosine Similarity Based on Average Image	64
Table 5.12 – Cosine Similarity of Extracted Features between Default and Debias Methods	64

LIST OF ABBREVIATIONS

ADAM. – Adaptive Moment Estimation
ANN. – Artificial Neural Network
CNN. – Convolutional Neural Network
DDPM. – Denoising Diffusion Probabilistic Model
DDIM. – Denoising Diffusion Implicit Model
DL. – Deep Learning
DM. – Diffusion Model
ELBO. – Evidence Lower Bound
GAN. – Generative Adversarial Network
FID. – Frechet Inception Score
KL Divergence. – Kullback-Leibler Divergence
ML. – Machine Learning
NCSN. – Noise Conditional Score Network
SD. – Stable Diffusion
SDE. – Stochastic Differential Equation
SDXL. – Stable Diffusion eXtra Large
SGD. – Stochastic Gradient Descent
VAE. – Variational Autoencoder

CONTENTS

1	INTRODUCTION	19
2	BACKGROUND	21
2.1	MACHINE LEARNING	21
2.1.1	SUPERVISED LEARNING	22
2.1.2	UNSUPERVISED LEARNING	23
2.2	ARTIFICIAL NEURAL NETWORKS	24
2.3	LEARNING PROCESS	25
2.3.1	LOSS FUNCTION	25
2.3.2	BACKPROPAGATION	26
2.3.3	OPTIMIZER	26
2.4	DEEP LEARNING	27
2.4.1	AUTOENCODERS	28
2.4.2	CONVOLUTIONAL LAYERS	28
2.4.3	POOLING LAYERS	29
2.5	IMAGE SYNTHESIS IN DEEP LEARNING	29
2.5.1	VARIATIONAL AUTOENCODER – VAE	29
2.5.2	GENERATIVE ADVERSARIAL NETWORK – GAN	30
2.5.3	DIFFUSION MODELS	31
2.5.4	DEEP LEARNING IMAGE SYNTHESIS TRILEMMA	32
2.5.5	EVOLUTION OF DIFFUSION MODELS	33
2.5.6	CLASSIFIER-FREE GUIDANCE	40
3	RELATED WORK	43
3.1	UNCOVERING SOCIETAL BIASES ON STABLE DIFUSSION	43
3.2	HARMFUL CONTENT FILTERING ON STABLE DIFFUSION	43
3.3	MITIGATING BIAS ON STABLE DIFFUSION	45
4	METHODOLOGY	47
4.1	EVALUATING BIAS PERCEPTION FROM HUMAN ANNOTATIONS	47
4.1.1	STAGE 1 - UNDERSTANDING THE DIFFUSERS LIBRARY	47
4.1.2	STAGE 2 - MODIFYING THE DIFFUSERS LIBRARY	49
4.1.3	STAGE 3 - CREATING THE POOL OF IMAGES	51

4.1.4	STAGE 4 - SETTING A PLATFORM TO EVALUATE BIAS PERCEPTION . .	53
4.2	MEASURING CONSISTENCY OF OUR MODEL <i>VERSUS</i> COMPETING TECH- NIQUES	54
4.2.1	ORIGINAL SDXL	54
4.2.2	OUR DEBIASED SDXL	55
4.2.3	SDXL NEGATIVE PROMPT	55
4.2.4	HUMAN PROMPT ENGINEERING	56
4.2.5	LOOKUP TABLE	57
5	EXPERIMENTAL ANALYSES	59
5.1	EVALUATING DEBIASING CAPABILITIES OF OUR METHOD FROM HU- MAN PERCEPTION	59
5.1.1	FIREFIGHTER	59
5.1.2	NURSE	60
5.1.3	BUSINESS LEADER	60
5.2	OUR METHOD’S CONSISTENCY TO SDXL <i>VERSUS</i> COMPETING TECH- NIQUES	62
5.2.1	AVERAGE OF INDIVIDUAL SIMILARITY	63
5.2.2	SIMILARITY OF AVERAGE IMAGE	63
5.2.3	FEATURE EXTRACTION VIA RESNET-50	64
5.3	AUTONOMOUS BIAS DETECTION ENGINE ON SDXL	65
5.3.1	AUTONOMOUS BIAS DETECTION ENGINE ON SDXL	65
6	DISCUSSION	67
6.1	EVALUATING HUMAN BIAS PERCEPTION	67
6.2	QUANTITATIVE SIMILARITY EXPERIMENT	70
6.3	BIAS SELECTOR EXPERIMENT	71
7	CONCLUSION AND LIMITATIONS	73
7.0.1	LIMITATIONS	74
	REFERENCES	75
	ATTACHMENT A – Grid of Images for each Method	77

1. INTRODUCTION

The area of text-to-image synthesis has witnessed remarkable advancements, culminating in the development of generative models like the Diffusion Probabilistic Models (DDPM) [Ho et al., 2020], Denoising Diffusion Implicit Models (DDIM) [Song et al., 2020], and more recently, the Stable Diffusion eXtra Large (SDXL) model [Podell et al., 2023]. The SDXL model has demonstrated state of the art capabilities in generating high-fidelity images from textual prompts, unlocking avenues for creative expression and practical applications across various domains.

However, despite their remarkable achievements, the efficacy of current Stable Diffusion models (particularly SDXL) is constrained by inherent biases prevalent in the training data. These biases pose critical ethical concerns, as biased image generation perpetuates and exacerbates societal biases, reinforcing stereotypes and potentially contributing to discriminatory outcomes in various contexts. Therefore, addressing and mitigating these biases lie at the heart of ensuring responsible and ethical use of AI-driven text-to-image synthesis systems.

In this work, we explored the bias issues ingrained within Stable Diffusion models, especially focusing on SDXL, and propose a novel debiasing mechanism at the core of the diffusion process. The foundation of this mechanism revolves around modifying the concept of Classifier-Free Guidance to steer the generation process away from harmful concepts.

The default Classifier-Free Guidance traditionally conditions the noise towards the encoded text prompt, ensuring coherence and relevance in the generated images. However, in this research, we go beyond the conventional approach by modifying the Classifier-Free Guidance to not only condition the generation towards the embedded text prompt but also away from a harmful concept. Unlike previous approaches that simply filter out or blur explicit content like nudity or violence, we repurpose the Classifier-Free Guidance concept to reshape the image during the generation steps. This method acts directly at the core of the denoising mechanism within SDXL.

Our approach draws inspiration from the Safe Latent Diffusion paper, which utilized a harmful text concept as a filter for explicit content in Stable Diffusion 1.5. However, in our research, we make use of this harmful concept as an active debiasing element. We also expand the work to the improved version of Stable Diffusion, the SDXL [Podell et al., 2023], now 3 times larger than the previous model. In our work, we manually define a few societal biases as text, which will become the harmful concepts to be avoided. By guiding the diffusion process away from societal biases encoded as this harmful concept, we strive to create a more equitable, inclusive, and unbiased generation process, contributing to a more responsible AI ecosystem.

Moreover, the intention of mitigating biases at the core of the diffusion process is that it changes the original images as little as possible. This preserves the powerful generation of Stable Diffusion with most of the original elements still present on the image, minimizing potential impacts on image quality.

Therefore, the hypothesis of this thesis is that the guiding mechanism at the core of the diffusion process, known as classifier-free guidance, can consistently debias harmful concepts and still preserve the original elements of the image.

In summary, this work leverages the paradigm of Classifier-Free Guidance to imbue SDXL models with a robust debiasing mechanism, guiding image generation away from societal biases. Through this innovative approach, we aim to foster ethical advancements in text-to-image synthesis, ensuring responsible AI innovation that aligns with fairness principles.

This work is organized as follows: Chapter 2 covers a brief background on the AI image generation field; Chapter 3 presents a recent literature review to explain current debiasing alternatives; Chapter 4 details the methodology of this research, pointing on how the method was designed and how data was collected; Chapter 5 presents the results of the experimental analysis, while Chapter 6 discusses the findings; finally, Chapter 7 depicts the conclusions of this thesis together with limitations of the proposed approach.

2. BACKGROUND

This chapter covers multiple levels of complexity to build a general understanding of how Stable Diffusion eXtra Large is used for text-to-image generation. We recommend the reader to skip the initial sections if those are well understood.

2.1 Machine Learning

An intuitive understanding of Machine Learning (ML) is that of a computational algorithm which allows the computer to learn by itself. This learning process, given a certain task, is the results of performance improvement over time on that particular task. As the performance itself can only be enhance considering all previous attempts of the algorithm, which constitute the necessary learning experience.

In order to properly build a machine learning algorithm, one must understand this field as the intersection of several other fields of knowledge. Probability and statistic, artificial intelligence, data engineering and computational complexity theory are present in the foundation of machine learning. But in order to efficiently extract the capabilities of this foundation, the specific domain application also must be understood. Thus, we may expand the knowledge requirements to fields such as philosophy, psychology, neurobiology or marketing, business management, finance and so on.

A very specific and formal understanding of what is machine learning was provided by Tom Mitchel [Mitchell, 1997]. He said that “a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .”. It translates, in a freer format, into the intuitive understanding provided at the start of this section.

It is a fairly common mistake to take machine learning by data mining. Even though the line between the two fields seems to be blurred, machine learning techniques often serves as tools along the data mining knowledge discovery process. Still, an important distinction is made clear when comparing the definitions provided by Tan [Tan et al., 2005] and Mitchell [Mitchell, 1997]: as data mining relies on human experience to learn new information, machine learning is self sufficient on the learning task.

In fact, machine learning can be often used as the automated data mining algorithm to extract some level of information in KDD, which then is understood by the human scientist. Complementary, data mining techniques are often a requirement to feed machine learning algorithms, mostly related to data preprocessing – e.g., feature selection, dimensionality reduction, data normalization or data subsetting.

We might expand on the scope of machine learning and even see it as a bridge between traditional data mining and artificial intelligence.

An interesting example of early machine learning applications resides on an algorithm that learns to play checkers. As the algorithm iterates over time and accumulates learning experience, it measures itself by the winning rate at each game. By doing so, it eventually understands the patterns driving the wins and repeats them, avoiding the patterns associated to losses. This whole process is understood as a form of training the algorithm, in other words, learning from the necessary experience and efficiently completing the task.

Following the formal definition of Mitchell, the example of checkers can be broken down into three main components. The given task T is represented by the action of playing the game, the performance P is measured by the percent of games won against the opponents and the training experience E represented by the games played against itself.

In the scope of this work, the task of image synthesis is the focus – even though the model can be expanded into tangential tasks. In a similar exercise, the task T is represented by generating a new image – or synthesizing. The performance P is a measure of quality and diversity of the images generated. And finally, the experience E is obtained through the training iterations of the model, given the processes of feed forward, backpropagation and parameters updates – all to be described in more details along this work.

2.1.1 Supervised Learning

There is an important distinction in the machine learning process when it comes to the learning experience. Some machine learning models will be treated as supervised; while some will be treated as unsupervised learning. The condition that drives each definition is the existence or not of a predefined class in the data – referred as *target* [Fawcett and Provost, 2013].

The supervised learning process is presented as a learning experience which relies on the existence of this predefined class in the data. The model then tries to segment the data, supported by the uncovered patterns, in order to fit each datapoint into one of the classes. Then it compares the current classification with the *target*, adjusting the learning experience until the predicted classification ideally converges to the same class provided by the *target*.

A more intuitive way to understand the concept of supervised learning is to think of the metaphor of a teacher supervising the student in the learning process. The teacher, in this case, is responsible to provide the *target* or ground truth information through his answers. With the student lies the responsibility of refining his own understanding of the subject guided by the ground truth.

The understanding of a problem between supervised and unsupervised learning can be subtle, but important. It is pointed out in [Fawcett and Provost, 2013] the distinct techniques applied to each learning model, which should not be interchanged from one another.

One last consideration must be mentioned for supervised learning. That is the limitation the dataset might impose for a certain *target*. To illustrate this limitation, one might think of a target that proposes the prediction of a customer churn event after the first six months. Suppose the dataset contains only a couple of months of data. Now it becomes clear the limitation imposed by the lack of data given this particular *target*.

To further illustrate this application of supervised learning, the K-Nearest Neighbors algorithm can be examined. It consists on a classifier which represents each datapoint in a d -dimensional space, being d the number of features. For each new datapoint, the proximity to existing datapoints is measured in d -dimensions. Since the existing datapoints have an assigned class (thus, the supervised nature), we can then compute which class is the nearest to the new datapoint and assign the predominant class to it [Tan et al., 2005]. Clearly, it shows how the classification is guided by the classes in the data, which are understood as the *targets*.

2.1.2 Unsupervised Learning

In opposition to supervised learning, the unsupervised process is not dependent on a predefined *target* in the dataset. Fawcett defines the unsupervised method as a more open task, where no specific purpose preexists and the model is allowed a certain freedom to find patterns in the data as it sees fit [Fawcett and Provost, 2013].

As a result from this method, the model can aggregate data into unlabeled groups through the learned patterns. In each group the pattern is consistent and the datapoints share similarities with each other, but the meaning of the group is not guaranteed to be useful. Leveraging the metaphor of the student, the unsupervised method can be seen as an exercise of the student searching for information on its own. He will eventually learn something from the studies, but his assumptions and understanding are not guaranteed to be accurate.

One advantage of unsupervised learning, inherit from its nature, is that the model will learn whatever pattern it can, given whatever data it has access to. This removes the limitation imposed to supervised learning on which the data must contain examples of the given task in order to be useful.

Typically, this method is used for tasks around clustering. The K-means algorithm is a fine example of unsupervised learning. Even though no class is known on the data, each datapoint has its corresponding location in the d -dimensional space. This technique assumes that similar datapoints are near of each other in that space and assigns a certain class to that group. Similarly, the datapoints centered in other region of the space, with a significant gap of data from the first group, belong to a separate class.

The K-means algorithm defines K random centroids in the d -dimensional space, where K is an arbitrary number of centroids chosen by the user. Then each datapoint is assigned to the closest centroid as some sort of class defining the group. Eventually all points are assigned,

creating clusters of data dependent of the centroids. To refine the assignment, all centroids are recalculated using the d -dimensional mean of the respective cluster.

Finally, the datapoints are reassigned to the new closest centroid and the entire process is repeated until the centroids no longer change in position. This enables a spherical segmentation of data in space, finding patterns of similarity in each cluster [Tan et al., 2005].

2.2 Artificial Neural Networks

The technique named Artificial Neural Networks (ANN) is heavily inspired by the way biological brains evolved to learn. In biological neural systems, the learning consists on a process dependent on a phenomenon called neural synapse. A basic description of it is given by Tan, starting from nerve cells called *neurons*, which are connected to each other by a fiber called axons. These axons can transmit electrical impulses through the nerve tissue from one neuron to the next, given a certain stimulation on the neuron or activation of the neuron. More specifically one neuron is connected to the axons of the next neuron via dendrites, an extension from the cell body of the neuron. The contact point between the axons and the dendrites is called a synapse. This is relevant for the scope of this work since neurologists discovered that by adjusting the strength of the synaptic connection on each neuron, the learning process occurs [Tan et al., 2005]. Mathematically, this adjustment can be seen as weights calibration between neurons, as will be shown later. Moreover, the adjustments in biological neural systems happen repeatedly, until the learning pattern is absorbed. Similarly, the iterative process to train artificial neural networks is based on reinforcing the identified patterns.

The first attempt to describe a mathematical model of the human brain was proposed in [McCulloch and Pitts, 1943]. In their work, the structure of neurons and the synaptic process was accounted in an equation and a mathematical description of a biological neuron network was presented. Their equations point how these networks can change over time – given a recurrent stimulus – and how neural activity can be calculated, once translated into electrical impulses travelling through the brain. Even though theoretical at the time, this method was relevant as it enabled a mathematical model, an equation, which could be designed to describe impulses travelling in this network, from receptors activated all over the body, into the neural system and finally into the brain cells.

A decade later, the foundation of current ANNs is found on the Perceptron idea [Tan et al., 2005] proposed by Rosenblatt in 1958 [Rosenblatt, 1958]. Here each neuron is composed by several input nodes and one output node. The input nodes receive the input data and adjusts the information by a certain weight before forwarding the result into the output node. There in the output node, all adjusted inputs are added together with the addition of another factor called bias. Once the output node computes the whole calculation of weights and bias, the resulting number is then evaluated by an activation function which determines whether the neuron output should be a positive or a negative input into the next neuron. By the perceptron

definition, the output is correlated to the sign of the adjusted inputs. If the computed number is positive, the neuron output is set to 1. If negative, the neuron output is set to -1 .

Artificial Neural Networks fall into the supervised learning method, since the learning process is based on a predefined target, often called ground truth. Along the training process, this simple network learns by adjusting its weights until the output of the perceptron is consistent to the true output of the training data. The process of adjusting the weights carries one more parameter, called learning rate. This learning rate dictates how fast or slow the learning process will occur as it directly adjusts the weights of the model.

A single basic perceptron with the linear activation function is not very powerful. The real benefit of ANNs comes with the Multilayer Artificial Neural Network and non-linear activation functions.

As the name suggests, Multilayer ANNs are built with multiple layers in sequence and each layer is built with several nodes. The whole network can be understood as an expansion of the regular perceptron, adding complexity at each layer. The first layer is called the input layer, as the input data is passed into the network. Next, the following layers vary in quantity but are called hidden layers and the corresponding nodes are called hidden nodes. Finally, the very last layer of the network is called the output layer, as it provides the learned classification.

A more visual interpretation of Multilayer ANNs consist on projecting hyperplanes into the d -dimensional space for data segmentation. A traditional perceptron can create a single hyperplane, but as the correct classification of data typically requires multiple hyperplanes to split classes, is easy to see how limited ANNs can be. As an example, by using a two layer feed-forward neural network, we project one hyperplane for each hidden node and therefore can add more complexity to segment the data. After both planes are applied, the output node is responsible for combining the boundaries to return the classification decision [Tan et al., 2005].

As for the the activation function, it can be configured into nonlinear curves to explore more complex patterns, such as ReLU, Sigmoid, TanH or Softmax. This enables each node to propagate the information with nonlinearity, a desired condition to several learning patterns in order to improve learning capabilities.

2.3 Learning Process

2.3.1 Loss Function

The method to compute the error of what has been learned by the networks is often called the Loss Function. A simple example of a Loss function is the sum of squared errors between the predicted result \hat{y} and the ground truth y . As we calculate the Loss through different combinations of weights on the network, it creates the Loss Landscape.

This landscape is a helpful way to visualize how the Loss behaves across the possible combinations and the main objective in the learning process is to find the minimum global spot of this landscape – meaning the combination of weights which result in the lowest possible Loss. Interestingly, different activation functions will shape different Loss Landscapes.

2.3.2 Backpropagation

To accomplish this task of optimizing the Loss, greedy algorithms like gradient descent were developed. The mentioned learning rate has a direct impact in the gradient descent as it multiplies a certain derived term before it updates the weights on the network, which is the core of backpropagation process.

Finally, a technique called backpropagation has been developed to handle complex weight adjustments on Multilayer ANNs, more specifically on its hidden nodes with non-linear activation functions. This technique is composed by two stages. First the ANN goes through the forward stage (from input layer to output layer direction), computing the output value of each neuron by leveraging whatever weights were previously set – either from previous iterations or randomness. The second stage is called backward and runs from output layer to input layer direction. The backward stage recalculates the weights, adjusting each value by a term that includes the learning rate and a derived term that accounts how much the weight should be increased in a direction that reduces the overall error term [Tan et al., 2005].

2.3.3 Optimizer

The so called optimizer is an algorithm responsible for adjusting neural network’s parameters, such as weights and learning rate. Several techniques can be applied for that adjustment, but Mini-Batch SGD and ADAM [Kingma and Ba, 2014] present state-of-the-art performance and are widely applied in modern architectures.

Mini-Batch SGD

Gradient descent is a key concept in neural network’s training and this technique is responsible for gauging the accuracy of the learning process – when combined with the loss function – as well as responsible for the parameters’ updates as the network learns.

Traditionally gradient descent is done over the entire dataset and updates the parameters only after that, which is not feasible for large-scale datasets. The stochastic component of Stochastic Gradient Descent comes to tackle this limitation. In SGD a single instance of the dataset is stochastically selected for gradient descent routine, enabling memory optimization

even for large datasets. Still, SGD process is not optimal as it takes time to iterate through the entire dataset and each iteration creates noise as it evaluates only a single instance.

Mini-Batch SGD tries to bridge the benefits of both previous methods. It breaks down the dataset in equal parts, called mini-batches, and shuffles the data randomly such that each part is similarly diverse. The network learns from each mini-batch and applies gradient descent on the group of instances. It still optimizes memory usage and enables frequent updates of the parameters, but reduces a lot of the noise from SGD.

ADAM

Still a common problem found in Mini-Batch SGD is when the gradients are trapped in a local minima. This prevents the network from keep learning and reaching the desired global minima.

In order to work around this issue, ADAM or Adaptive Moment Estimation optimizer was designed [Kingma and Ba, 2014]. ADAM leverages the concept of momentum, similar to how physics portrait it. If the learning rate is too accelerated as it decays along loss landscape (meaning the momentum is high), the parameter is adjusted such that its velocity decreases and it does not jump over a local minima. Similarly, if it is trapped in a local minimum with low momentum, the learning rate is increase in order to build enough momentum to get out of the trapped region and resume learning.

2.4 Deep Learning

Deep Learning builds on the concept of traditional Machine Learning, consisting of very large and complex ANNs – therefore, it can be seen as a subset of ML.

It is widely explained in [Goodfellow et al., 2016] and it has achieved significant success on several tasks on latest years, typically around unstructured data. Some successful tasks are found on human level image detection, speech recognition, image synthesis or complex text generation. An important contrast in Deep Learning (DL) when compared to traditional ML is that the first revolves around an automated feature selection, while the later typically needs a manual feature selection.

This condition allows the DL network to select the best features as it learns, converging more easily to optimum results on complex tasks. An interesting pattern is expected on DL networks: the first layers of the network are typically learning a broader representation of the groundtruth and as we move deeper into it, more refined and detailed representations can be seen. The network's output data is then constructed by of a chain of events, on which complex representations are built on top of broader representations.

This behavior enables the network to select the best branches of features (representations), thus achieving a minimum local or global loss during learning phase. This process is

known as representation learning and enables the network to quickly adapt to new tasks as it learns.

The idea of Representation Learning is powerful, but representing human understanding is complex and a more sophisticated architecture is required. This is known as an autoencoder.

2.4.1 Autoencoders

This algorithm combines an encoder function, converting the input data into a reduced representation. Then a decoder function, converting the representation back to the original format. Although this process seems pointless at first glance, the technique is powerful to extract key properties of the data.

The architecture is built in a way that it creates a bottleneck between the encoder and decoder, a data space that is known as the latent space. By applying this restriction, we force this network to select a more efficient way to represent the data and then learn to rebuild it out of that restricted representation. This architecture is currently found on most deep generative models due to its ability to simplify data representation.

2.4.2 Convolutional Layers

Convolutional or deconvolutional blocks are one of the main components of autoencoders. Those are key to compress and expand data along the network.

It is based on a mathematical operation referred as convolution. In this operation, two functions are being overlapped, while one is mirrored on the y axis and shifted along the x axis. It is then computed the integral of the overlap.

Translating this process into matrix representation, there is typically a smaller matrix referred as kernel which overlaps the entire original matrix, shifting its position at each iteration. Then the product of each overlapped cell is computed and added into a single value. This process happens for every position the kernel assumes as it moves along the original matrix.

Lastly, the output is a smaller matrix, which combines the results generated by the kernel into each corresponding cell.

Depending on the defined kernel, it is possible to extract very specific features on data – such as isolating only the horizontal and vertical edges of an image.

2.4.3 Pooling Layers

Another key component of autoencoders are the pooling layers. Much like convolutional layers, this layer consists of reducing the size of a larger matrix by overlapping a smaller matrix with it – called filter. It then shifts the smaller matrix along the larger one, computing different operations.

The main difference from convolutional layers resides on the performed operation. Instead of a convolution, pooling layers can compute the max value of each filter window (known as max pooling [Zhou and Chellappa, 1988]), the average of all values and so on.

2.5 Image Synthesis in Deep Learning

Currently there are three main deep generative models for image synthesis with different advantages over each other.

Variational Autoencoders [Kingma and Welling, 2013, Rezende et al., 2014] were the first to present good generative capabilities and have a well based theory behind its functionality, as well as good sampling speed and training process.

Generative Adversarial Networks [Goodfellow et al., 2014] came after and to this day present a groundbreaking generative capability on high-resolution images with also fast sampling. But GANs came with an important drawback. The requirement to train an adversarial network creates instability in the training process as the network can easily collapse.

The most recent model proposed was the Diffusion Model [Song and Ermon, 2019, Ho et al., 2020], on which both training and sampling are relatively easier and very stable processes, but also present the ability to generate highly complex images which directly compete with GANs. Still not perfect, Diffusion Models have their main drawback in the sampling speed as it present thousands of iterations for each synthesized image.

2.5.1 Variational Autoencoder – VAE

A VAE consists of an encoder-decoder architecture on which images \mathbf{x} are transformed into a latent space representation \mathbf{z} at a connection point called bottle neck. This bottle neck is located between the output of the encoder and the input of the decoder.

As an image distribution travels through the network, it is first compressed by the encoder in direction of the bottle neck region and then, given this latent representation, it is decompressed by the decoder back into a similar image.

An intuitive understanding of this learning process is that by restraining the high dimensional features of the original image and reconstructing it from a compressed latent representation, the network will focus on the most relevant features during the reconstruction.

As summarized in [Goodfellow et al., 2016], during training the encoder network is used to obtain the latent representation \mathbf{z} – given an original image \mathbf{x} – which is then inputted into the decoder to reconstruct a similar image of \mathbf{x} . During this process, VAEs follow a compound loss function on which the general objective is to maximize the variational lower bound associated with data points of image \mathbf{x} .

$$\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \log p_{\text{model}}(\mathbf{x} | \mathbf{z}) - D_{\text{KL}}(q(\mathbf{z} | \mathbf{x}) || p_{\text{model}}(\mathbf{z})) \quad (2.1)$$

This loss can be interpreted as compound as it breaks down into two terms. The first term reflects the expectation of the latent variable as it travels through the encoder network q . This expectation is then connected to the log likelihood (i.e., logarithmic probability) of returning the original image via the decoder, provided a latent variable. In this process, the encoder adjusts its weights such that it maximizes the expectation of finding this latent representation, while the decoder adjusts its weights such that it maximizes the probability of correctly reconstructing the image, given the latent representation found. In this arrangement, both networks adjust their weights simultaneously.

The second term tries to approximate the latent representation from the encoder to the latent representation of the decoder by leveraging KL divergence between the data distributions. This reflects how well both representations match each other, returning zero on the event of a perfect match and tending to positive infinite as the distributions diverge from each other. As the overall goal is to maximize the equation, this second term subtracts from the first.

Finally, during sampling the VAE network discards the encoder and leverages the learned representations from the decoder. By providing a sample latent variable \mathbf{z} into the differentiable generator network $\mathbf{g}(\mathbf{z})$, which in its turn serves as input into the distribution model $\mathbf{p}(\mathbf{x} | \mathbf{g}(\mathbf{z}))$ (or decoder), we can sample a synthesized image \mathbf{x} – or the simplified version of sampling given by $\mathbf{p}(\mathbf{x} | \mathbf{z})$, which simply omits the generator network.

2.5.2 Generative Adversarial Network – GAN

GANs were introduced in [Goodfellow et al., 2014] and then improved in [Karras et al., 2018, Karras et al., 2019]. It consists of a zero-sum game problem between two deep neural networks. The first network – called the generator – has to learn the best way to produce images through its function $\mathbf{g}(\mathbf{z}; \theta(\mathbf{g}))$, while it competes against a second network – called the discriminator – which on its turn has to distinguish between samples originated from the groundtruth or the generator itself.

In other words, the discriminator has to decide whether a sample is real or fake, thus produced by the generator. The discriminator is then defined by a probability value returned from $\mathbf{d}(\mathbf{x}; \theta(\mathbf{d}))$, which states how likely the sample is a fake one.

At the same time, the generator network has to improve in order to fool the discriminator and produce fake images as close as possible to reality. At the convergence point, the generator should produce images indistinguishable from the groundtruth, such that the discriminator loses the ability to predict. At this point, training is done and the discriminator is no longer necessary.

A formal learning equation of GANs is explained in [Goodfellow et al., 2016] as the zero-sum game defined by function $v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)})$. Here the performance of the discriminator is evaluated by real and generated samples.

$$v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log d(\mathbf{x}) + \mathbb{E}_{\mathbf{x} \sim p_{\text{model}}} \log(1 - d(\mathbf{x})) \quad (2.2)$$

Although these models can generate impressive images, the authors recognize the challenging training process involved and the instability of the network when it comes to ensuring a convergence point. [Goodfellow et al., 2016] points that a simultaneous gradient descent on two networks does not guarantee an equilibrium point.

At last, this model demands careful planning and maintenance during training so it does not collapse.

In reality, the generator aims to increase the log-probability that the discriminator makes a mistake, rather than aiming to decrease the log-probability that the discriminator makes the correct prediction. Stabilization of GAN learning remains an open problem.

2.5.3 Diffusion Models

Another type of generative models is called Diffusion Models. Its basic functionality is to gradually add noise into an image as an iterative process (as shown in Figure 2.1), then reverse this noise-adding process with as many iterations. By doing so, the groundtruth of the noise-removal process is created.



Figure 2.1 – Diffusion process with cosine schedule. Source: [Nichol and Dhariwal, 2021].

Finally, an ANN, typically a U-Net, can learn to mimic this noise-removal process by computing the loss between a random attempt to remove noise and the actual groundtruth. It performs this learning process for every groundtruth created for the corresponding timestep.

For example, an image is corrupted 1000 times by adding small amounts of noise each time. Then 1000 examples are generated which describe how the original image turned into pure noise. This 1000 examples become the groundtruth for a single image. The U-Net then uses these examples to go backwards on the process. It starts from pure noise and learns what portion of noise should be removed in order to move towards the original image. This is training process of a diffusion model.

Once the training is complete, the U-Net can start from pure noise and remove small amounts of noise over 1000 iterations to reconstruct an image.

There are nuances on Diffusion Models, such as Schedulers, Guiding Scales and other in-depth concepts that will not be explained on this background.

As a general rule, Diffusion Models deliver a high-quality image generation with good diversity, all with much less complexity than GANs to setup the network as it does not require adversarial training. But it comes with a cost on sampling time, as Diffusion Models are much slower to sample due to its natural iterative process. Improve sampling time is an active area of research.

2.5.4 Deep Learning Image Synthesis Trilemma

Variational Autoencoders (VAEs) [Kingma and Welling, 2013, Rezende et al., 2014], Generative Adversarial Networks (GANs) [Goodfellow et al., 2014, Karras et al., 2018, Karras et al., 2019] and Diffusion Models [Song and Ermon, 2019, Ho et al., 2020, Song et al., 2020] are the three main flavors of current Deep Generative Models, but each method displays advantages and disadvantages when compared to the others. This comparison has been explored in [Vahdat et al., 2021] and reflects the *Generative Learning Trilemma* shown in Figure 2.2.

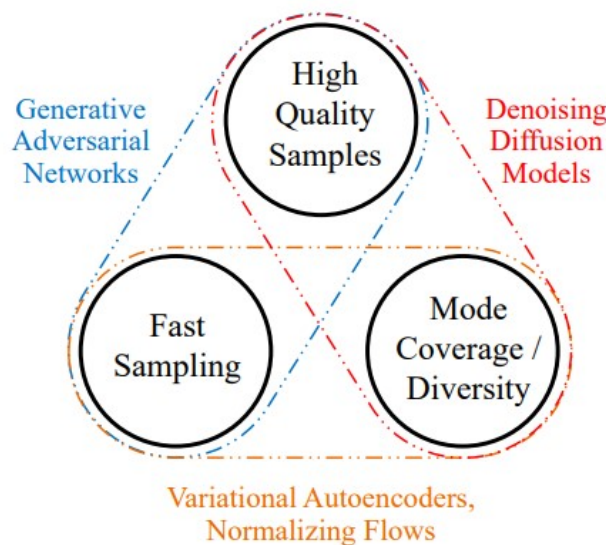


Figure 2.2 – Generative learning trilemma. Source: [Vahdat et al., 2021].

As supported by the trilemma, Diffusion Models present high-quality image generation that VAEs cannot achieve, while they also present a diversity in generation that GANs do not have. Additionally, Diffusion Models display stability during training, in contrast to GANs' adversarial training which possibly collapses if left unattended.

Even though Diffusion Models show very attractive characteristics, on the downside they have an intrinsic slow sampling process due to the necessary iterative process.

2.5.5 Evolution of Diffusion Models

Diffusion Models rely on two major designs: score-matching models and probabilistic models. The latter is expanded into an optimized framework which is explained along this work, but still relies on the probabilistic nature.

Denoising Score-Matching

Score-matching [Song and Ermon, 2019] is the first of the two frameworks. It uses a neural network known as a Noise Conditional Score Network (NCSN), which is responsible to estimate the gradient of the logarithmic data density, called the score function.

As usual in Diffusion Models, the forward process consists of a gradual noise-adding process and the reversed process generates data by gradually removing that noise, in this case, guided by the score.

The forward process follows equation below. Suppose the data distribution of \mathbf{x}_{t-1} is known, then noise \mathbf{z}_t is added to it as well as the score function, all weighted by a given step size ϵ . The result is the data density of $\tilde{\mathbf{x}}_t$. Then the process is repeated until there is pure Gaussian noise, becoming $\mathbf{X}\mathbf{T}$.

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2} \nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon} \mathbf{z}_t \quad (2.3)$$

The reverse process (or denoising) relies on sampling the data distribution of \mathbf{x}_t by using only the score function. This way, an ANN can be trained such that it approximates the score function ($s_{\theta}(x) \approx \nabla_x \log p_{\text{data}}(x)$) and computes the loss as the expected squared error of score estimates – a process known as Score Matching.

$$\frac{1}{2} \mathbb{E}_{p_{\text{data}}} [\|\mathbf{s}_{\theta}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})\|_2^2] \quad (2.4)$$

This method introduces two important problems [Song and Ermon, 2019]. The first is that the score estimator is inconsistent for data on low-dimensional manifold, leading to an inconsistent loss shown in the left chart. To adjust this problem, a very small amount of noise is added to the data on each iteration (perturbing the data) but ensuring that at least noise is

present on the whole manifold space and leads to high-dimensional manifold – in other words, data distribution has full support. This enables loss convergence as shown in the right chart of Figure 2.3.

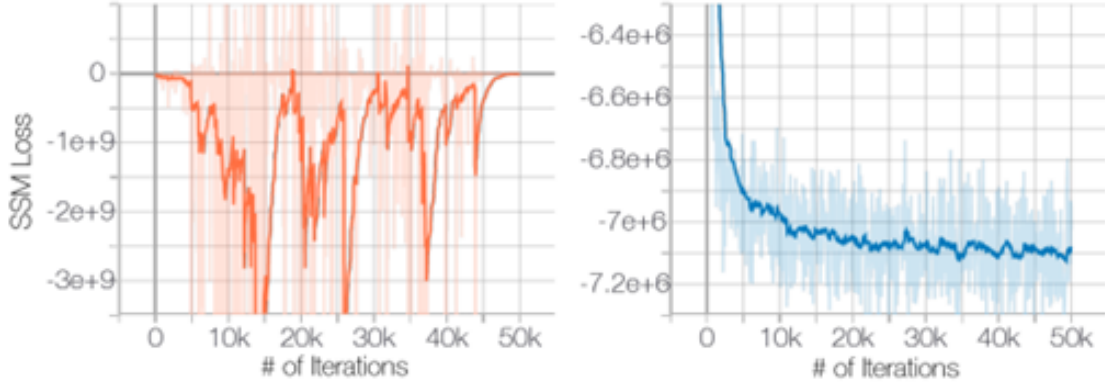


Figure 2.3 – Sliced Score Matching (SSM) Loss. Source: [Song and Ermon, 2019].

The second problem resides on low density regions in the manifold, which does not provide enough training signal to estimate the score. Similarly, this is solved by adding Gaussian noise over those regions, which has the positive effect of filling the space and intensifying the signal for score estimation. Therefore, the second good reason to add some noise during iterations.

Even though this method is proven to be a powerful generative framework, it still requires an undesirably long iterative process, which can be prohibitive in some applications.

Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Model [Ho et al., 2020], or DDPM, is another framework largely implemented for Diffusion Models. It has the same foundational concept of Score-Matching models – that is, to gradually add noise during forward, then learn a gradual denoising process – but it proposes the objective function based on the probability of data distributions, rather than a gradient score.

The forward process is done via a diffusion kernel, which returns the corresponding noise on \mathbf{x}_t , given an original image \mathbf{x}_0 . The noise is then added over multiple iterations until there is pure Gaussian noise on $\mathbf{X}\mathbf{T}$.

This kernel is helpful as it isolates the noise from the image, enabling the noise to be incorporated into the original image at any time step, but also to compare predicted noise to groundtruth directly with no need to compute the image itself.

$$\int \underbrace{q(\mathbf{x}_0)}_{\substack{\text{Input} \\ \text{data dist.}}} \underbrace{q(\mathbf{x}_t | \mathbf{x}_0)}_{\substack{\text{Diffusion} \\ \text{kernel}}} d\mathbf{x}_0 \quad (2.5)$$

Furthermore, the framework uses a Schedule parameter called β for each time step. β defines how much noise is incorporated at each time step. This β schedule is then consolidated into a Markovian Chain in the form of α – which reflects the compound probability of adding noise to any given time step \mathbf{t} .

Sampling from this framework is done by providing the original image \mathbf{x}_0 as well as a pure Gaussian distribution ϵ . After that, α defines how much of the original image will be preserved and how much noise is added to \mathbf{x}_t .

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon \quad (2.6)$$

Another interpretation of sampling is that α changes the mean of the data distribution (for each data point), slowly shaping a Gaussian normal distribution into the characteristic distribution of an image. The variance at each time step follows the scheduled β and by consequence, α .

Putting things together, this diffusion kernel based on α is efficient as it enables the forward process to sample image \mathbf{x}_t directly at any time step while in closed form – even though it still needs to compute the iterative noise on the timestep.

Finally, the denoising process is conceptually done in the same way as Score-Matching. As depicted in Figure 2.4, it aims to remove noise at each iteration, starting from $\mathbf{X}\mathbf{T}$, until \mathbf{x}_0 becomes the groundtruth.

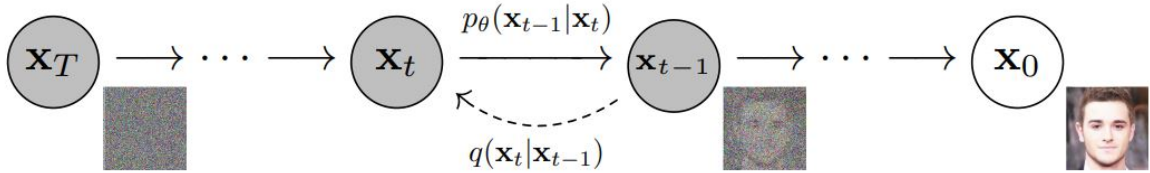


Figure 2.4 – Denoising process from diffusion model. Source: [Ho et al., 2020].

The biggest difference in DDPM when compared to Score-Matching diffusion is on the learning objective. Instead of approximating the score, the objective now is to approximate the predicted noise distribution itself.

A complete loss function is built from three terms, two compared by KL divergence. The first term, L_T , refers to timestep \mathbf{t} and can be ignored with the assumption that the distribution of $\mathbf{X}\mathbf{T}$ (being a pure Gaussian noise) has no important distinction between groundtruth and predicted values – this is supported by a fixed β schedule during training. The second term refers to all timesteps between \mathbf{t} and 0 and is where the actual learning happens, as the predicted noise (a Markovian chain) drives the loss in comparison to groundtruth. Lastly, the third term refers to the transition into the groundtruth (or \mathbf{x}_0).

$$\mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T | \mathbf{x}_0) \| p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0} \right] \quad (2.7)$$

Since some terms can be ignored, the authors also propose a simplified objective function. This compact form of the loss covers the expected squared error between the noise (ϵ) and the predicted noise (ϵ_θ) for each timestep. This is done providing the groundtruth image distribution and a pure Gaussian noise.

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t, \mathbf{x}_0, \epsilon} \left[\left\| \epsilon - \epsilon_\theta(\sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \epsilon, t) \right\|^2 \right] \quad (2.8)$$

Even with the new features proposed by DDPM’s framework, one major limitation is still present: sampling time requires multiple iteration and is typically prohibitive for real time applications.

Denoising Diffusion Implicit Models

Denoising Diffusion Implicit Models (DDIM) [Song et al., 2020] were proposed as an optimization of DDPM framework, aiming to improve sampling speed and give more control over latent space. This work is related to the proposed hypothesis, but more importantly, comes as a background to enable latent space control.

DDIM is still an iterative generative model, but with somewhat faster sampling while generating high quality images when compared to other frameworks. Still not as fast as sampling on GANs, which require only one pass through the network.

This framework is defined as an implicit probabilistic model and shares the same objective function as DDPM. It also uses the same network as DDPM, but the main difference is in the non-Markovian diffusion process, which can be simulated in less steps during sampling.

Additionally, DDIM have properties that DDPM does not share. By controlling the latent variables, DDIM is consistent on generated high-level features while sampling over various lengths of Markov chains. That is to say that 50 steps, 100 steps or 500 steps of generation in DDIMs would result in similar high-level features, given the same latent variable – which weights in the control appeal of this framework.

Another important aspect of DDIM’s control is the ability to interpolate latent variables and still preserve semantic information, shown in Figure 3.1. This is possible because in DDIM the high-level features are encoded by $\mathbf{X}\mathbf{T}$ (as the noise-adding steps are no longer stochastic), as opposed to DDPM which has a stochastic process to generate $\mathbf{X}\mathbf{T}$. This leads to very different \mathbf{x}_0 given the same $\mathbf{X}\mathbf{T}$ in DDPM, but very similar \mathbf{x}_0 for similar $\mathbf{X}\mathbf{T}$ in DDIM.

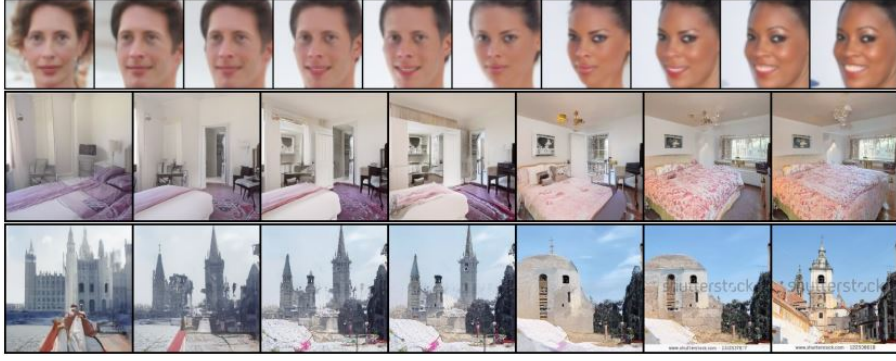


Figure 2.5 – Interpolating DDIM samples. Source: [Song et al., 2020].

The diffusion process in DDIM is redesigned into a non-Markovian iteration, as \mathbf{x}_t is dependent on \mathbf{x}_{t-1} and \mathbf{x}_0 at the same time.

$$q_{\sigma}(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{q_{\sigma}(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) q_{\sigma}(\mathbf{x}_t | \mathbf{x}_0)}{q_{\sigma}(\mathbf{x}_{t-1} | \mathbf{x}_0)} \quad (2.9)$$

Similarly, the generative process is redesigned to account the non-Markovian denoising, given a fixed prior \mathbf{XT} as a specific latent variable from Gaussian noise.

$$p_{\theta}^{(t)}(\mathbf{x}_{t-1} | \mathbf{x}_t) = \begin{cases} \mathcal{N}\left(f_{\theta}^{(1)}(\mathbf{x}_1), \sigma_1^2 \mathbf{I}\right) & \text{if } t = 1 \\ q_{\sigma}\left(\mathbf{x}_{t-1} | \mathbf{x}_t, f_{\theta}^{(t)}(\mathbf{x}_t)\right) & \text{otherwise} \end{cases} \quad (2.10)$$

This enables Diffusion Models to perform a faster sampling with somewhat small trade-off in image quality, but still requires the traditional iterative training from DDPM as the objective function stays the same.

Stable Diffusion

Stable Diffusion 1.5 (SD 1.5) encompasses a multi-step process during inference that involves several crucial components, including a Variational Autoencoder (VAE) network, two Tokenizers and Text Embedding networks and a U-Net network. Each component plays a distinct role in generating high-quality images from textual prompts while leveraging a diffusion-based generative process.

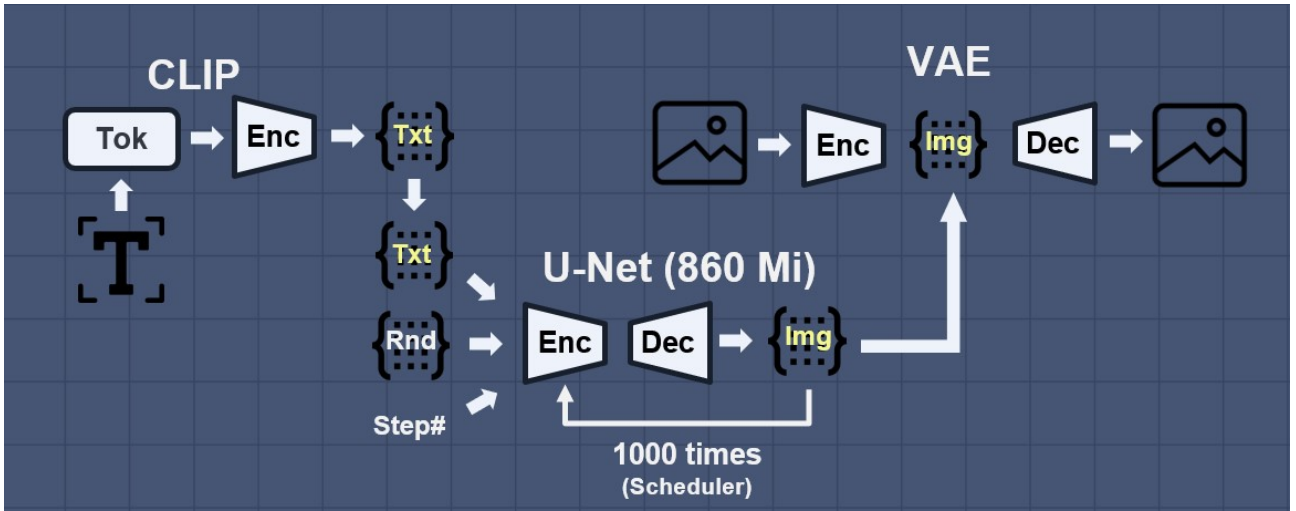


Figure 2.6 – Inference on Stable Diffusion v1.5.

The Tokenizer and Text Embedding Networks work in sequence by first processing the inputted text prompt through a CLIP Tokenizer and then processing the tokens by a CLIP embedding network. In other words, these networks convert the input text into a structured representation suitable for the model to understand – referred as a latent representation of the text.

The text embeddings encode semantic information from the text, which guides the image generation process. The text embeddings are the reference to ensure that the generated images aligns with the content specified in the text prompts.

Operating as a bi-directional translator between the pixel space and the latent space, the VAE network is designed to convert images while preserving significant semantic information. Throughout the training phase of the Stable Diffusion model, the encoder component of the VAE undertakes the responsibility of transforming original images from the pixel space into a condensed latent representation. This transformation is pivotal as it involves compressing the intricate details and features inherent in the pixel space, distilling them into a lower-dimensional latent space. During inference, the VAE decoder is then utilized to convert the latent representations back into pixel space to finally render the image.

Because the latent representation compresses the pixel information, it is possible to run the iterative diffusion mechanism much faster during both training and sampling stages. Still, the representation is a comprehensive encoding of the semantic information contained in the original images, as described in [Rombach et al., 2022].

As the generative process unfolds, the U-Net network functions as a denoising autoencoder. It is the kernel of the diffusion process. The U-Net receives three inputs: an initial gaussian noise (in latent space), the encoded text prompt and the number of the diffusion timestep for the scheduler. With these inputs, it predicts how much noise should be subtracted from the inputted noisy latent. After the predicted noise is subtracted from the latent variable, the updated latent serves as an input into the U-Net again for the next iteration - keeping the same encoded text prompt as guide and advancing to the next timestep.

The U-Net architecture, known for its encoder-decoder structure, performs successive transformations on the noise sequence. The encoder part of the U-Net encodes the noisy information, while the decoder part reconstructs denoised versions of the noisy inputs. This denoising process helps refine the noise sequence step by step, gradually moving it towards the space of realistic images.

In a summary, SD 1.5 translates textual information into high-quality image generation by iteratively refining noise sequences in a diffusion-based process guided by the semantic content of the textual prompts.

Stable Diffusion eXtra Large – SDXL

Stable Diffusion 1.5 (SD 1.5) and Stable Diffusion eXtra Large (SDXL) [Podell et al., 2023] are iterations within the Stable Diffusion framework, each introducing enhancements and modifications, primarily in model architecture, scale, and performance.

SD 1.5 represents an improvement upon earlier versions, focusing on refining stability, training dynamics, and better generation quality. It includes modifications in the denoising process, updating the U-Net architecture used for image restoration and refinement. The model retains the core components of the Stable Diffusion approach, emphasizing text-to-image synthesis with improved stability and fidelity.

On the other hand, SDXL is an advancement that significantly scales up the Stable Diffusion model in terms of size and capacity. It incorporates a larger architecture, encompassing more parameters and layers, enabling it to handle more complex tasks and generate higher-resolution images with better fidelity. SDXL’s increased scale aims to enhance the model’s capacity for generating details and diverse images.

In terms of parameters in the model, the improved SDXL carries two U-Net networks of over 2B parameters each, while the previous version 1.5 carries only 860M parameters, making the each U-Net on the newer version 3 times larger. On the U-Nets, the SDXL counts with two distinct networks using the concept of specialized denoising kernels from the work of eDiff-I model [Balaji et al., 2023].

The first U-Net, referred as “Base”, is specialized in the initial steps of the denoising process, therefore drawing the general shapes, colors and elements in the image. While the second U-Net, referred as “Refiner”, is specialized in the later steps of the denoising process, adding better details and textures to the image.

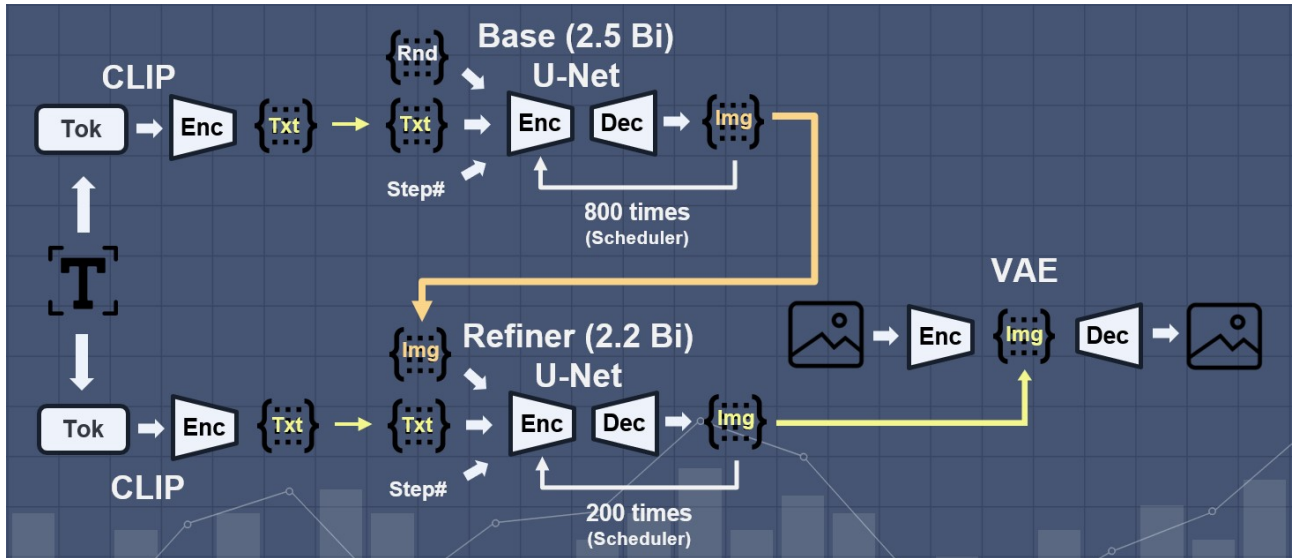


Figure 2.7 – Inference on Stable Diffusion eXtra Large.

In essence, while SD 1.5 concentrates on refining the model’s training dynamics and improving generation quality, SDXL primarily emphasizes scaling up the model’s architecture, allowing it to handle more complex and high-resolution images.

2.5.6 Classifier-Free Guidance

The Classifier-Free Guidance (CFG) is at the core of the diffusion process and, as the name suggests, it guides the image generation process towards the desired image representation during inference.

This mechanism is applied within the denoising loop of Stable Diffusion and works in conjunction with the predicted noise from the U-Net. As detailed on Figure 2.6 and 2.7, the U-Net receives as input the conditional text embeddings from the CLIP network. This serves as a guide for the U-Net to start predicting the noise that should be removed.

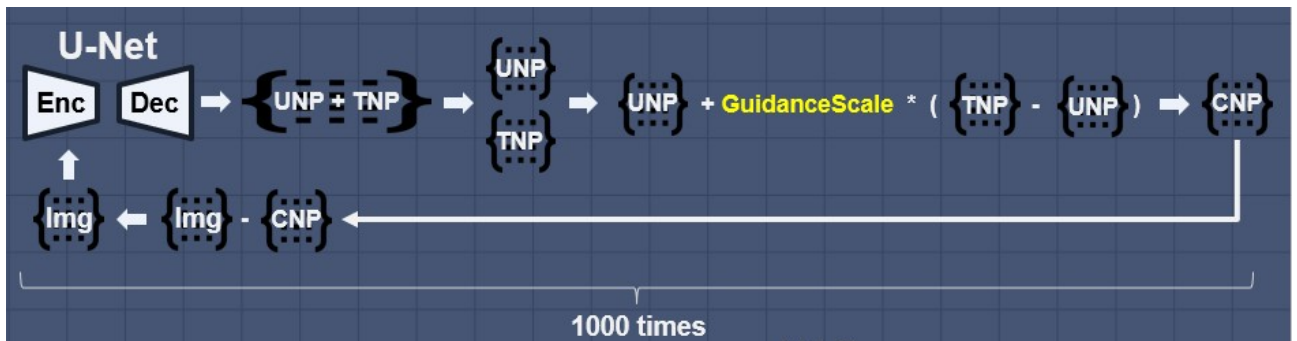


Figure 2.8 – Classifier-Free Guidance on SDXL.

The predicted noise is composed by two sections on the default model. The first part of the predicted noise reflects the Uncondicional Noise, being the noise that would be subtracted

from the random latent in order to generate a random image. The second part of the predicted noise reflects the Textual Noise (from the text embeddings) and it is used to shape the image towards a very specific direction in the data space, rather than any direction.

These Textual Noise (TNP) is used on the following equation to influence the Unconditional Noise (UNP) towards a certain direction in data space, weighted by a Guidance Scale (gs). This results in the Conditional Noise predicted (CNP):

$$CNP = UNP + gs \cdot (TNP - UNP) \quad (2.11)$$

Lastly, the CNP is subtracted from the initial random latent representation, therefore removing a certain amount of noise from it. This produces a slightly less noisy version of the latent representation.

The cycle then repeats as the U-Net is now fed the slightly less noisy latent and predicts the Unconditional and Textual noises of the subsequent step in the diffusion process. The Conditional Noise is calculated again and subtracted from the latent representation again. This process happens typically over 50 iterations distributed along 1000 denoising steps (also known as *Timesteps*). Which timestep correspond to each iteration is defined by the specific Scheduler used on the diffusion process.

An intuitive understanding of CFG is displayed at Figure 2.9.

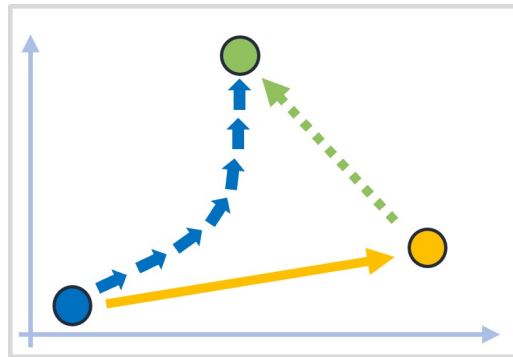


Figure 2.9 – Denoising Process with Classifier-Free Guidance on SDXL.

Give a random latent representation which started at the blue position in the data space, if left unconditioned during the denoising process, let's assume it would move towards the yellow position in the data space. This would represent a random image concept at the end of the denoising process, such as a bicycle, a human being or a flower of any kind.

Because of the CFG method, we are able to steer the denoising process towards a desired region in the data space. Assuming the guidance is represented by the green arrow, we have a vector pushing the generation towards the green position. So on each step of the denoising process, the latent moves a little in the intended direction. Finally, the latent stops at the green position which represents the desired image concept.

3. RELATED WORK

Generative AI, particularly Text-to-Image (T2I) models have witnessed remarkable strides, with SDXL [Podell et al., 2023] emerging as a prominent player. As an open-source alternative to MidJourney (www.midjourney.com), it has demonstrated significant potential in image generation. However, the presence of biased image outputs in SDXL necessitates careful consideration due to its potentially broad societal impact, especially with its usage on advertising and social media. A few papers tried to uncover some of the biases in Stable Diffusion, but fewer tried to propose mitigation strategies on those biases.

3.1 Uncovering Societal Biases on Stable Difussion

The authors in [Bianchi et al., 2023] aim to shed light into the present biases of Text-to-Image generation. They highlight the potential dangers involved in both reproduce and amplify societal stereotypes in these models. They point out that the concept of “whiteness” is often reinforced as ideal, resulting in amplification of gender and racial disparities, just like the concept of “American” is reinforced in multiple generations of different objects. Lastly, the authors make an important observation that so far these models don’t show an effective mechanism to mitigate societal biases, neither are focused on preventing the perpetuation of stereotypes.

In the work of [Wolfe et al., 2023], the societal bias of women objectification is explored in nine text-to-image models. The authors bring the definition of sexual objectification as the effect of “a person’s human characteristics, such as emotions, to be disregarded and the person is treated as a body or a collection of body parts”.

Following this definition, the authors validated through a series of experiments that the less clothes the subject has in generated images, the less emotions are displayed. A concerning bias of sexual objectification derived from biased data scraped from the internet. They also validated that female professionals are more likely to reflect sexual descriptions and that prompts of “X age girl” are 42% of the time sexualized in Stable Diffusion, as opposed to only 9% for prompts of “X age boys”.

3.2 Harmful Content Filtering on Stable Diffusion

A few papers covered interesting mechanisms for safety filters in diffusion models, which can be repurposed for debiasing.

The work of [Rando et al., 2022] exposes the flaws in the unsafe filter of previous version of Stable Diffusion (1.5). They show how the filter can be bypassed to generate sexual

content and how it is ineffective against violence and gore content. The authors took the first step into reverse engineering the filter to document its functionality and pave the way for future improvements.

This original filter is not meant for debiasing the model, but only acts as a mechanism to hide unsafe images. The filter is applied post image generation and uses CLIP latent space to map the present concepts in the image. These concepts are compared against a lookup table carrying unsafe concepts and the cosine similarity is calculated. If the similarity surpasses a defined threshold for each unsafe concept, the image is blurred.

This original mechanism from 2022 is simplistic and it does not offer a flexible mechanism to influence image generation during the diffusion process itself.

The paper Safe Latent Diffusion [Schramowski et al., 2023] proposes an interesting workaround on unsafe concepts in image generation. It comes as a filter applied during inference that prevents Stable Diffusion from generating a list of unsafe concepts, such as “violence”, “blood”, “nudity” or “children”, to name a few. This work is specially interesting as it focuses on the core of the diffusion guiding mechanism: the Classifier-Free-Guidance.

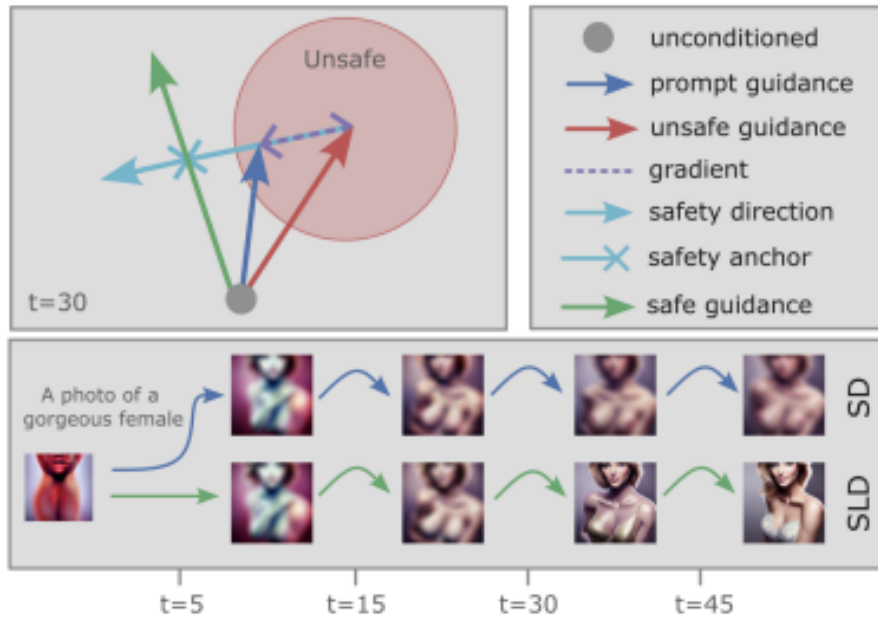


Figure 3.1 – Safety Concept Guidance. Source: [Schramowski et al., 2023].

The authors propose a text embedding representing all the unsafe concepts to be avoided. Then during the denoising loop, where the classifier-free-guidance is applied, the predicted noise is at the same time conditioned towards the text prompt and away from the unsafe concept. This produces promising results without artificially changing the image, as the guidance is done at the very kernel of Stable Diffusion.

This work is applied on the outdated version of Stable Diffusion and originally represents a filter instead of a debiasing mechanism. Nonetheless, it is the main inspiration for

our hypothesis and opens the possibility of repurposing the classifier-free guidance process as a debiasing engine.

3.3 Mitigating Bias on Stable Diffusion

[Friedrich et al., 2023] takes the next step into debiasing Stable Diffusion. They once again show that the models carry intrinsic societal biases originated from biased data randomly scrapped from the internet.

Additionally, they propose a human created lookup table storing all biases for each prompt the authors mapped. They leverage the kernel of diffusion model to shift biases guided by a text prompt, in hopes to guide the model towards fairness concepts.

The caveat present in this approach is on scalability. It requires human evaluation of specific prompts in order to map the present biases that should be suppressed. Moreover, the authors applied an artificial trigger that debiases the generation 50% of the time, but keeps the biased generation for the other 50% of the time, in hopes to reach gender equality. Defining the split percentage creates another factor for human intervention. Lastly, the technique was applied on Stable Diffusion 1.5, which is now outdated.

[Shen et al., 2023] propose first an adjustment to the loss designed in Stable Diffusion in order to guide semantic attributes on the generated images towards a user-defined target distribution. The authors claim it preserves the original semantic information.

Secondly, they propose a finetuning process which considers the improved loss, in order to debias the model. The authors seem to achieve good results, but at the expense of extending the training process of a large network, while also needing to arrange specific debiased data which will support the debiasing finetuning.

At the present moment only a few papers proposed debiasing strategies on diffusion models. Out of those, none of the papers covered SDXL version, leaving room for both improvement in the debiasing strategy, as well as for extending the application to the new version of Stable Diffusion.

4. METHODOLOGY

This Chapter lists all design choices that facilitated our experiments.

4.1 Evaluating Bias Perception from Human Annotations

In order to test the hypothesis in regards to perceived debiasing capabilities, we designed four main stages. This section will described in details how the method was designed, how it was set up and how it was evaluated along the five stages.

4.1.1 Stage 1 - Understanding the Diffusers Library

The first stage was a careful assessment of the current SDXL library, so we could break down the code to isolate the diffusion kernel and Classifier-Free Guidance (CFG) method. The selected implementation of SDXL was extracted from the Diffusers library from Huggingface, found on the GitHub repository at [von Platen et al.,]. This library is composed by several variants of diffusion models, therefore we focused on reproducing the code found on the SDXL text-to-image pipeline.

After studying this pipeline, the structure of the model became clear. The code is composed by auxiliary functions for data preprocessing and the main denoising loop function which is responsible for the actual image generation. This pattern of functions was found for both the *base* and *refiner* subsystems of the model.

The pipeline is displayed in Figure 4.1. It starts by setting the height and width of the image. Next it encodes the text prompt (and negative text prompt if supplied) into an embedded latent representation to guide the denoising loop later. Then the specific denoising timesteps are defined based on the desired number of iterations. The random latent variable is then created, which will serve as a blank canvas for the denoising process to shape the generated image. Moving further, the text embedding and negative text embedding are appended into a single object, giving the text latent representation the expected dimensions.

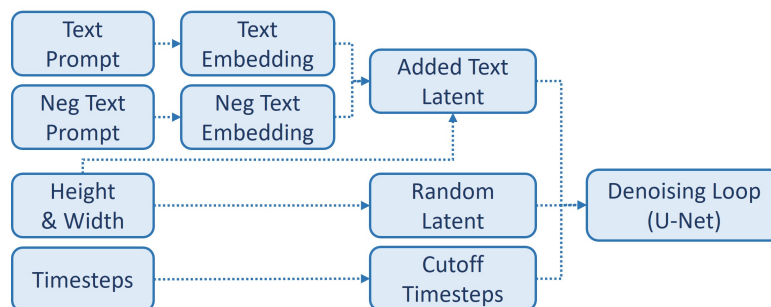


Figure 4.1 – SDXL Preprocessing Graph.

Lastly, a cutoff is defined for the timesteps, considering how much of the image should be generated on the base and refiner subsystems (e.g. given 100 timesteps and a 80%/20% distribution between the base and refiner, 80 steps would be performed on the base subsystem, to then hand-off the latent to the refiner subsystem to resume the last 20 steps). Everything to this point is around preprocessing data for the model.

Finally the denoising loop begins following Figure 4.2. The main network behind the image generation (a U-Net) receives the random latent, the corresponding timestep and the text embeddings. The U-Net then predicts which noise should be removed from this random latent in order to guide it towards the desired concept.

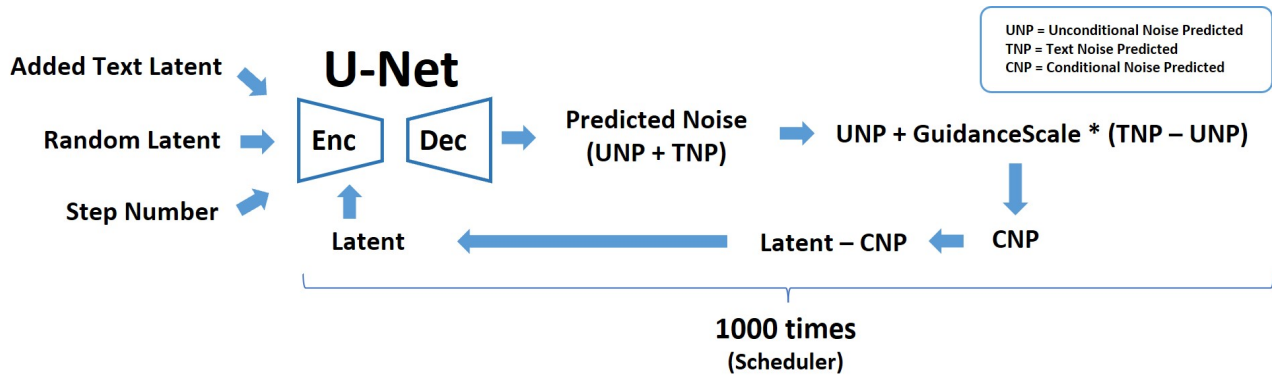


Figure 4.2 – SDXL Denoising Loop Graph.

After that, the predicted noise is separated into the *unconditioned noise* and the *text noise*. The following is the default Classifier-Free Guidance routine: first the difference between the text and unconditional noise is calculated. Then this difference is scaled up by one of the hyperparameters. Lastly, the unconditional noise is added again. This results in a combined noise that will be subtracted from the latent representation.

Once the latent is updated by the combined noise, it serves as an input into the U-Net looping again. The timestep now moves into it's next value. The text embedding is constant to continue guiding the image. This process repeats until all timesteps in the list are covered.

As an intuitive reminder, the predicted noise is deterministic and will depend exactly on:

- The timestep being calculated: typically more noise is removed earlier and less noise is removed later, depending on the scheduler.
- The desired text embedding: will guide the noise removal towards specific dimensions in the data space, thus deining what the predicted noise looks like.
- The random latent: serving a random blank canvas, the initial noise on this latent could start anywhere on the data space. So the U-Net has to be flexible and adapt to different noise distributions.

4.1.2 Stage 2 - Modifying the Diffusers Library

Understanding the applied code was crucial for the second stage. Here we focused on modifying the data preprocessing functions and the denoising loop to incorporate a debiasing element on the Classifier-Free Guidance routine.

The first modification was by adding the bias concept as a textual input and then encoding this concept in to an embedding. Similarly to the default application, the bias concept was tokenized and encoded by the two CLIP models in the base subsystem, then merged into a single object for the correct dimensions.

The prompt embed object was also modified. It was expanded in order to carry the bias concept within the object as depicted in Figure 4.3.

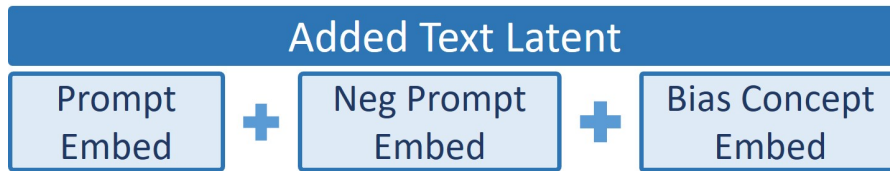


Figure 4.3 – Added Text Latent from expanded Embeddings.

A few additional modifications were performed to ensure the correct dimensions along the preprocessing functions.

The most significant modification was introduced into the denoising loop. Initially, the random latent was extended to match the extended dimensions of the embeddings. Then the U-Net was used to predict the noise as usual.

Because the U-Net received an extended embedding and an extended latent, the dimensions of the predicted noise were also extended. Now the predicted noise had three components: the *unconditioned noise* and the *text noise* like before, but also the *bias concept noise* as a third element.

The Classifier-Free Guidance routine was modified to account for the third noise element and an entirely new routine was implemented as described in Figure 4.4.

In this new routine the *bias concept noise* carries a more complex process if compared to traditional CFG. It's values are scaled and clamped based on new hyperparameters and it builds momentum as the denoising loop progresses. The other two predicted noises (*unconditioned noise* and *text noise*) remain unchanged.

At the end of the routine, a very similar operation happens: the adjusted *bias concept noise* and the *text noise* are subtracted from the *uncondined noise* and scaled up in order to be added into the *unconditioned noise* again. Once more, this results in a combined noise (referred as *conditioned noise predicted*) that will be subtracted from the latent representation.



Figure 4.4 – Modified Classifier-Free Guidance with Bias Concept.

Like before, we restart the denoising loop with the updated latent representation and repeat it until all timesteps on the list are covered. This extended Classifier-Free Guidance implementation was heavily inspired by [Schramowski et al., 2023].

Intuition of CFG as a Debiasing Mechanism

In an attempt to produce a more intuitive explanation, we present Figure 4.5. It is a bi-dimensional data space where we start with a random latent at the blue point. If no guidance was offered during the generation process, let us assume the latent would move towards the yellow point, generating a random concept (a car, a house, a person, etc). As the CFG is introduced, we have the green vector pushing the image towards a specified region on each timestep. The CFG vector creates the curved effect showed by the blue arrows, guiding the generation towards the light green point.

Now we introduce the debiasing concept defined by the red point. This is a point in the data space which identifies a harmful or unwanted concept. As we create a region around this point, we can define an entire region on the data space to be avoided.

The debiasing mechanism pushes the generation (blue) away from the red region if it moves towards the unwanted concept (red). In theory, the benefit is that this process produces a very similar latent in general as it lands on a very close position to what default CFG would be. Still, it has the ability to effectively avoid the unwanted concepts. In practical terms, if the default generated image was of a black-haired child kicking a soccer ball and the avoided concept was the soccer ball, most of the concepts in the image would remain unchanged. Hopefully,



Figure 4.6 – Generated images with our method.

Table 4.1 – Hyperparameters of SDXL & debiased SDXL.

inference steps	50
denoising ratio	0.99
refiner strength	0.01
guidance scale	12
debiasing scale	15000
debiasing threshold	0.025
debiasing warmup steps	7
debiasing momentum	0.5
debiasing beta	0.7

4.1.4 Stage 4 - Setting a Platform to Evaluate Bias Perception

With the pool of images created, we designed a bias perception evaluation. We propose a qualitative assessment of the images performed by humans, considering that evaluating the human perception of bias is the end goal.

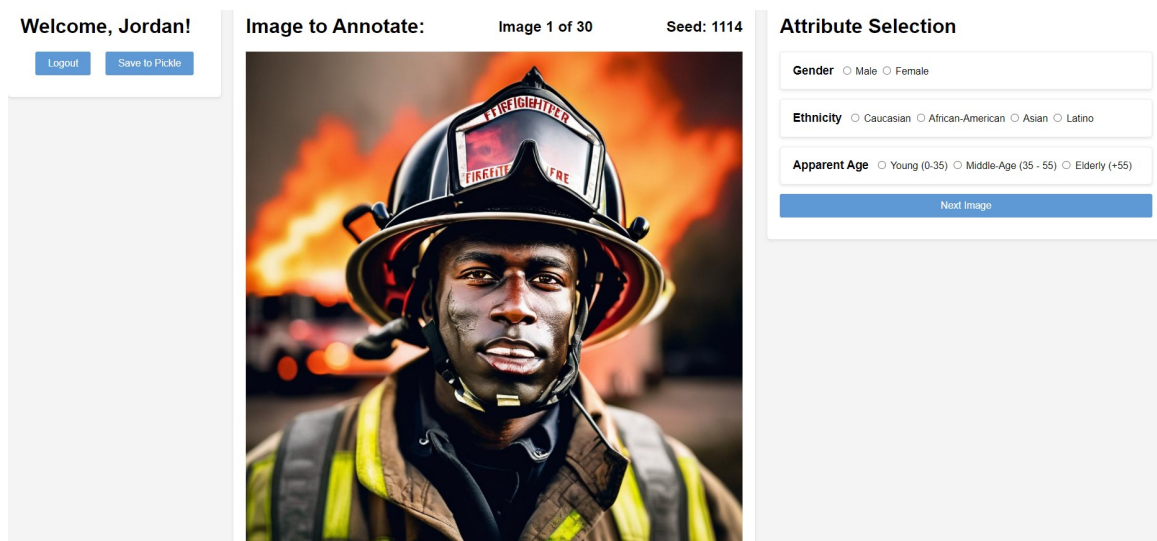
The idea was to annotate the human perception of each image on protected attributes of fairness - specifically on dimensions of gender, ethnicity and apparent age. This exercise included 9 participants to mitigate individual biases.

Next, we required a platform to host the application. It should present one image at a time to be evaluated without any indication of which model was used during generation, which textual concept or which biases were removed.

The platform should also allow the participant to annotate his perceptions of the protected attributes and store his answers for our analysis. Each participant evaluated the pool of 1,200 images, one image at a time.

This platform was developed as an interactive web application. The back-end was developed in Flask to pull all required information locally and manage concurrent sessions, allowing multiple users to evaluate the images in parallel. The front-end application was built using Java Script for custom functionality, CSS for styling the application and HTML as a framework to encapsulate all features. The connection service was deployed on a local network, so each user connected to the network via Secure Shell (SSH) protocol before accessing the application on the browser.

Figure 4.7 shows the front-end application on the browser, providing the user interface for each participant to perform the evaluation. The panel displayed on the right side was able to record the user's perception on the protected attributes. The images were annotated on batches of 30 and once finished, the information was saved back to the server as a Pickle file.



The screenshot shows a web application interface for image annotation. It is divided into three main sections:

- Welcome, Jordan!**: A top-left section with a "Logout" button and a "Save to Pickle" button.
- Image to Annotate:**: A central section displaying "Image 1 of 30" and "Seed: 1114". Below this is a large image of a Black male firefighter wearing a helmet and gear, with a fire in the background.
- Attribute Selection**: A right-hand panel with three radio button groups for selecting attributes:
 - Gender**: ☐ Male ☐ Female
 - Ethnicity**: ☐ Caucasian ☐ African-American ☐ Asian ☐ Latino
 - Apparent Age**: ☐ Young (0-35) ☐ Middle-Age (35 - 55) ☐ Elderly (+55)
 Below these groups is a blue "Next Image" button.

Figure 4.7 – User Interface for Image Annotation.

There were 9 participants on the evaluation and they were diverse in gender and ethnicity. Their age ranged from 19 to 32 years old, so all subjects were relatively young. Each person annotated 1200 images over a period of about four weeks.

4.2 Measuring Consistency of Our Model *versus* Competing Techniques

Given the encouraging results of the initial evaluation (presented in Chapter 5), we moved into a comparison with competitive methods. This exercise allowed us to test the hypothesis that our method was capable of preserving the original elements of the image. This comparison focused on evaluating our model’s capabilities of debiasing an image while staying true to the original model. Furthermore, we compared how consistent our model is to the original SDXL, against other techniques commonly present in the literature.

All techniques were reproduced locally on top of SDXL, which allowed us to be consistent on the computational environment. We also ensured all techniques ran with the same list of random seeds during generation - a list of 200 seeds were leveraged. This enabled a fair comparison between generated images, as the same random latent was used as the starting point for the denoising loop.

In order to enable this comparison, the very first step was to generate 200 images using each of the 5 methods described below.

4.2.1 Original SDXL

This method was implemented as the default version of SDXL and was utilized as a benchmark in the comparison. All other methods were compared to the images generated on this one.

The hyperparameters utilized on this original model were as follows:

- Text Prompt: 'the face of firefighter'
- Denoising Steps: 50
- Usage of Base model: 99% | Usage of Refiner model: 1%
- Guidance Scale: 12

4.2.2 Our Debiased SDXL

As already presented, our method was implemented by modifying the Classifier-Free Guidance present on the original SDXL library. We generated 200 images using our method with the following hyperparameters:

- Text Prompt: 'the face of firefighter'
- Bias Prompt: 'man traits, very manly male man, rough man, man, male, man, male, man, male'
- Denoising Steps: 50
- Usage of Base model: 99% | Usage of Refiner model: 1%
- Guidance Scale: 12

4.2.3 SDXL Negative Prompt

The Negative Prompt method is a very common approach and already available at the SDXL library. At a higher level, the idea sounds similar to our method: it consists of a textual prompt provided to the model, which is used to remove unwanted concepts from the image.

The main difference to our method is how it is implemented. The mechanism behind the Negative Prompt that guides the image generation is simplistic in comparison.

After embedding, the Negative concept is appended to the main Textual concept. Both are then used in the U-Net for the noise prediction, just like what was explained before. The first difference is on the noise expelled from the U-Net. As it is broken down into Unconditional and Textual noises, the Unconditional noise is actually conditioned by the Negative concept. The Textual noise is conditioned by the Textual concept as before.

During the Classifier-Free Guidance process, the Unconditional Noise serves as a base to calculate the final noise that will be subtracted from the latent. This logic results in the latent moving away from the Negative (by subtraction) and towards the Textual concept.

The Negative Prompt mechanism implemented is very different from the modified Classifier-Free Guidance in our model. This method results in images that deviate much more from the original concept, in comparison to our method, losing control capabilities in the data space.

We generated images with Negative Prompt using below hyperparameters:

- Text Prompt: 'the face of firefighter'

- Negative Prompt: 'man traits, very manly male man, rough man, man, male, man, male, man, male'
- Denoising Steps: 50
- Usage of Base model: 99% | Usage of Refiner model: 1%
- Guidance Scale: 12

4.2.4 Human Prompt Engineering

Prompt engineering is commonly defined as the process of leveraging human language in order to guide generative AI towards desired outputs. In essence, it is the process of finding a specific list of words (and the relationship between those words) that will more efficiently produce the desired outcome.

In our comparison the desired outcome consists of diverse generation in regards to protected attributes of gender, ethnicity and age. We explored the prompt that produced more diversity within a few options.

We initially engineered four different prompts and visually analyzed the output images of each option. This initial assessment was done for a batch of 30 images for each prompt engineered.

- Option 1: 'a male or female gender diverse firefighter, focused on the face'
- Option 2: 'the face of firefighter, asian, latino, african-american, caucasian'
- Option 3: 'a single firefighter person, focused on face, female or male'
- Option 4: 'the face of firefighter, female, male, asian, latino, african-american, caucasian'

From these samples, we selected the third option as it ended up generating the most diverse samples. By our evaluation, this method required us to focus on only one protected attribute.

Whenever multiple dimensions of protected attributes were included in the prompt engineering, the outcome was even more biased towards one of the values. If multiple ethnicities were included, the model paid attention to only one of the options in the prompt, then generated that same ethnicity most of the time.

We used the following hyperparameters for Prompt Engineering:

- Text Prompt: 'a single firefighter person, focused on face, female or male'
- Denoising Steps: 50

- Usage of Base model: 99% | Usage of Refiner model: 1%
- Guidance Scale: 12

4.2.5 Lookup Table

This idea consists of manually setting up a table of concepts to be avoided, therefore this can be implemented for debiasing purposes. The concepts are often stored as textual prompts, but can also work when stored as text embeddings.

The process is quite simplistic: the table carries multiple options for each value on protected attributes. For *gender*, we have values of "male" and "female". During inference, the model will use a given prompt, but also lookup for a gender word from this table and sample this word from a uniform distribution.

In our setup, the lookup table was built as a placeholder inside the prompt, which carries the randomly pulled gender: `random.choice(['male','female'])`.

This method can display variations in terms of the concepts included in the table or the distribution (uniform or skewed towards certain concepts if there is reason to it). In our table setup, gender was the only protected attribute and reasonably is represented as a uniform distribution.

The hyperparameters used on this method are described below:

- Text Prompt: 'the face of a ' + `random.choice(['male','female'])` + ' firefighter'
- Denoising Steps: 50
- Usage of Base model: 99% | Usage of Refiner model: 1%
- Guidance Scale: 12

5. EXPERIMENTAL ANALYSES

This Chapter presents the results found for all design choices described in Chapter 4. The discussion of the results is presented in Chapter 6.

5.1 Evaluating Debiasing Capabilities of Our Method from Human Perception

This experiment consisted of evaluating the web platform results. After the 4 stages described in Chapter 4, we had annotations regarding the perception of each participant on each image. This translated in roughly 10 thousand unique perceptions around gender, ethnicity and apparent age present on the images. Half of these perceptions were based on images of the default SDXL and half were based on our debiasing method applied on SDXL. The data was also broken down based on the concepts: firefighter, nurse and business leader.

After loading all the annotations and performing data validation, we were able to summarize the protected attributes in default SDXL and our debiasing method on SDXL. We measured, for instance, how many images of firefighters were perceived as males *versus* females on the default model and on our modified model. The same was repeated for each concept and each protected attribute.

The result of this experiment is presented as a percentage distribution comparing human perception on the original code of SDXL and our modified code of SDXL for debiasing. The results are split between textual concepts and protected attributes.

The textual concept and all hyperparameters for each evaluation was the same on both models. Our model also included the bias concept to be avoided, which naturally is not included on the original model. The results are presented in Sections 5.1.1, 5.1.2 and 5.1.3.

5.1.1 Firefighter

Here we generated images of firefighters using the textual concept "the face of firefighter". After some experimentation on our model, the bias to be avoided was defined as "man traits, very manly male man, rough man, man, male, man, male, man, male", with the intention to encapsulate the bias of what a "man" is for the model.

There were 1645 samples evaluated from the original model and 1641 samples from our debiased model. Tables 5.1, 5.2 and 5.3 show the percentage distributions for each protected attribute:

Gender	Original	Debiased
Female	0.182	25.289
Male	99.818	74.711

Table 5.1 – Original SDXL vs. our method: Genders on firefighter prompt.

Ethnicity	Original	Debiased
African American	12.644	15.478
Asian	0.122	0.244
Caucasian	82.310	78.184
Latino	4.924	6.094

Table 5.2 – Original SDXL vs. our method: Ethnicity on firefighter prompt.

Apparent Age	Original	Debiased
Elderly	0.426	2.986
Middle Age	84.985	69.653
Young	14.590	27.361

Table 5.3 – Original SDXL vs. our method: Apparent age on firefighter prompt.

5.1.2 Nurse

On this second evaluation we generated images of nurses using the textual concept "the face of a nurse" and we defined the bias to be avoided as "very womanly female woman, woman, female, woman, female, woman, female, woman, female", with the intention to encapsulate the bias of what a "woman" is for the model.

There were 1645 samples evaluated from the original model and 1631 samples from our debiased model. Tables 5.4, 5.5 and 5.6 show the percentage distributions for each protected attribute:

Gender	Original	Debiased
Female	98.419	75.720
Male	1.581	24.280

Table 5.4 – Original SDXL vs. our method: Genders on nurse prompt.

5.1.3 Business Leader

The third evaluation was focused on images of business leaders using the textual concept "the face of a business leader". The bias to be avoided was defined as "an old man,

Ethnicity	Original	Debiased
African American	19.891	26.609
Asian	0.122	2.207
Caucasian	71.776	54.629
Latino	8.212	16.554

Table 5.5 – Original SDXL vs. our method: Ethnicity on nurse prompt.

Apparent Age	Original	Debiased
Elderly	2.857	4.905
Middle Age	51.915	32.250
Young	45.228	62.845

Table 5.6 – Original SDXL vs. our method: Apparent age on nurse prompt.

caucasian, male, man, man, man, male, man, male, man, manly male, man traits", now with the intention to encapsulate the biases of what a "caucasian elderly man" means for the model.

On this evaluation we explored a combination of multiple biases being avoided at the same time in order to evaluate how our model performed.

There were 1648 samples evaluated from the original model and 1640 samples from our debiased model. Tables 5.7, 5.8 and 5.9 show the percentage distributions for each protected attribute:

Gender	Original	Debiased
Female	2.245	11.829
Male	97.755	88.171

Table 5.7 – Original SDXL vs. our method: Gender on business leader prompt.

Ethnicity	Original	Debiased
African American	1.214	17.622
Asian	2.367	32.561
Caucasian	93.143	21.098
Latino	3.277	28.720

Table 5.8 – Original SDXL vs. our method: Ethnicity on business leader prompt.

Apparent Age	Original	Debiased
Elderly	80.886	1.463
Mid Age	18.325	46.159
Young	0.789	52.378

Table 5.9 – Original SDXL vs. our method: Apparent age on business leader prompt.

5.2 Our Method’s Consistency to SDXL *versus* Competing Techniques

The first part of this experiment was to generate a list of 200 images for each of the 5 variations of SDXL we analyzed. The images were grouped by seed and displayed in a grid for qualitative analysis, as can be seen on Figure 5.1 showing a few samples.



Figure 5.1 – Samples of the generated grid of images.

Additional samples can be found on attachment A.1 on the generated grid.

Once all 1000 images were generated (being 200 from each method) we started the comparison to understand how consistent each method is to the original model.

A visual qualitative comparison was performed in order to understand nuances on how each method affected the semantics, shapes or colors present on the images. This analysis revealed important indications of how our method performs against current alternatives.

This comparison can followed the layout displayed on Figure 5.2. An expanded list of samples is presented on chapter 5.



Figure 5.2 – Images generated from each method using the same seed.

5.2.1 Average of Individual Similarity

Next we evaluated the methods in a quantitative way. For this analysis we leveraged the cosine similarity method, measuring how similar each debiasing method is to the original SDXL.

We computed the cosine similarity between original SDXL and each method, on each of the 200 seeds. This created a list of 200 similarity measurements for each of the 4 methods. Finally, we aggregated the measurements as an average and as a standard deviation for each of the 4 methods, resulting in 2 quantitative values to compare them.

Average of Individual Similarity - Results

We computed the individual cosine similarity for each method against the original SDXL and calculated the averages and standard deviations displayed on Table 5.10.

On this evaluation we are looking for which method is the most consistent to the original model. The higher the average similarity value, the better. In the opposite logic, the lower the standard deviation, the better.

Method	Average	Standard Deviation
Our Method	95.155	0.017
Negative Prompt	85.801	0.055
Prompt Engineering	84.266	0.062
Lookup Table	84.181	0.049

Table 5.10 – Average and Standard Deviation of Individual Similarities

5.2.2 Similarity of Average Image

Our next comparison measured how similar an average image of each method would be from an average image of the original model.

We computed an average pixel representation of the 200 images for each method, which resulted in Figure 5.3. Then by measuring the cosine similarity as before, we evaluated if the average pixel space was also consistent to the results found on the previous measurement.

Similarity of Average Image

The similarity based on an average image of each method was driven by the grid displayed on Figure 5.3. The similarity results are displayed on Table 5.11.

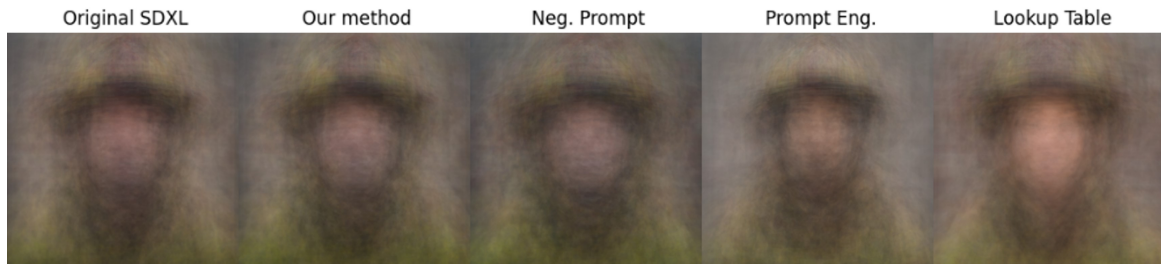


Figure 5.3 – Average of all 200 images in pixel space from each method.

Method	Cosine Similarity
Our Method	99.928
Negative Prompt	99.682
Prompt Engineering	99.659
Lookup Table	99.274

Table 5.11 – Cosine Similarity Based on Average Image

5.2.3 Feature Extraction via ResNet-50

Another quantitative evaluation was performed, now focusing on similarity of semantic features rather than pixel values.

This measurement leveraged a pretrained ResNet-50 network that ran through the 1000 images, extracting individual semantic representations for each of them. This created a list of 200 extracted features for each method, that were used to compute similarity.

The features of each image in the 4 debiasing methods were compared to the features of the original SDXL, using cosine similarity. This process converted the list of features into a list of similarities.

Lastly, all similarities were grouped by each method by computing it's average and standard deviation.

Feature Extraction via ResNet-50

Lastly, we measure similarity on this semantic representation consisting of extracted features. The average values and standard deviations are displayed on Table 5.12.

Method	Average	Standard Deviation
Our Method	95.682	0.015
Negative Prompt	90.975	0.047
Prompt Engineering	86.973	0.070
Lookup Table	87.914	0.052

Table 5.12 – Cosine Similarity of Extracted Features between Default and Debias Methods

5.3 Autonomous Bias Detection Engine on SDXL

This last experiment was an extension of the hypothesis. Our previous evaluations presented results (as discussed on Chapter 5) that drove us to further exploration.

The goal of this exercise was to explore an automated method to detect biases on the original model for a given protected attribute, such as gender.

This task was not yet presented on the literature for Stable Diffusion, to the best of our knowledge.

The first validation was to ensure that the tokenization and embedding processes in the SDXL were deterministic. This was important as they are the foundation of the latent conditioning. If those tokens and embeddings can be used to indicate the bias, the indication needs to be consistent given the same hyperparameters.

For this exercise, we ran the CLIP network with fixed hyperparameters and random seeds. After comparing the generated tokens, the same values were found every time, regardless of the seeds used. The same deterministic behavior was validated on the embeddings.

5.3.1 Autonomous Bias Detection Engine on SDXL

This method was designed around the embeddings used on SDXL. We explored the protected attribute of gender.

First we fixed an embedding representation of a man and an embedding representation of a woman, based on the text prompts of "person is a man" and "person is a woman", respectively.

Those served as a reference of two opposite concepts in the gender dimension. We understand that in many dimensions both embeddings will have similarities, considering both represent human beings. Still, even the smallest differences between them can be leveraged for a bias selector.

Next, we defined a regular implementation of SDXL with the hyperparameters below:

- Text Prompt: "the face of a business leader"
- Guidance Scale: 12
- Denoising Steps: 50
- Usage of Base model: 99% | Usage of Refiner model: 1%
- Batch Size: 1
- Number of Images: 1

As the SDXL starts by generating the embedding of "the face of a business leader", we captured that embedding and computed it's cosine similarity against the "man" and "woman" representations.

This measurement produced two similarity values, of which the highest value could indicate the bias direction in the gender dimensions.

This algorithm acts as a dynamic bias detection mechanism. Once the bias is identified, we can trigger our modified SDXL debiasing method and indicate which gender representation should be avoided.

Finally, by resuming the image generation process we can produce the debiased image with no human intervention on the bias identification. We repeated this process for multiple seeds and the results were consistent.

At first, this might seem like a process that would simply flip genders: a male on the original model would become a female and *vice-versa*. In reality, there is nuance on the modified Classifier-Free Guidance method in such a way that concepts which are biased towards "male" would suffer a delicate influence in order to generate more "females" and concepts biased towards "female" would generate more "males".

That is: regardless of the original bias direction, the inference would converge towards a uniform distribution between the genders. Hardly a uniform distribution itself, but moving towards it.

Lastly, there are nuances to this detection mechanism which can be vastly explored. We see this experiment as a very first step in SDXL autonomous bias detection.

The experiment around an automated bias selector leveraged the cosine similarity to identify the direction of the bias in the gender dimension.

There were multiple random seeds implemented on this validation. All seeds generating business leaders identified the **male** bias. The embedding similarity towards "male" was of 98.9549% and similarity towards "female" was of 98.9233%.

All seeds generating firefighters also identified the **male** bias. The embedding similarity towards "male" was of 98.7668% and similarity towards "female" was of 98.7252%.

All seeds generating nurses identified the **female** bias. The embedding similarity towards "male" was of 98.9820% and similarity towards "female" was of 99.0012%.

6. DISCUSSION

Our initial hypothesis covered two main ideas around the usage of Classifier-Free Guidance for debiasing mechanisms. The first idea was that this method was able to consistently debias the generated images. The second idea was that the generated images should remain as close as possible to the original model, except for the biased concept.

6.1 Evaluating Human Bias Perception

The first idea was initially validated through a visual comparison of a few samples and then the validation was scaled up with the web application described on previous chapter. By comparing the percentages derived from human perception, we found a strong indication that our method is capable of reducing unwanted biases on generated images.

On the "firefighter" evaluation we notice a substantial improvement on gender equality from 0.1% females on original model to 25.2% females on our method. On "nurses" we observed a very similar improvement on the opposite direction of gender, increasing male representation from 1.5% to 24.2%. Lastly on "business leaders" we notice a smaller improvement on female representation, but still relevant, from 2.2% to 11.8%.

To our surprise, other protected attributes were improved alongside the gender. The most noticeable improvements happened on:

- The "business leader" ethnicity attribute: originally 93% concentrated on Caucasian individuals and debiased to a 32.5%, 28.7%, 21% and 17.6% distribution on Asian, Latino, Caucasian and African American respectively.
- The "business leader" apparent age attribute: originally 80% concentrated on elderly individuals and debiased into a 52.3%, 46.1% and 1.4% distribution on young, middle age and elderly individuals.
- On "firefighters" the concentration on Caucasian middle aged individuals was reduced as well and other categories were improved.
- On "nurses" the concentration on Caucasian individuals was reduced and other categories were improved.
- On all evaluations we notice more equally distributed ethnicities and gender.

From these results we understand that some concepts seem to be intertwined with each other. By deviating from the concept of "man" on the "firefighter" evaluation, we deviated from the concept of "Caucasian". By deviating from the concept of "woman" on the "nurse"

evaluation we also deviated from the concept of "Caucasian". This lead us to an understanding that both genders in the model are associated with Caucasian people. As we explored a combination of concepts for debias on the "business leader" evaluation (both in gender "male" and ethnicity "Caucasian") we saw an even stronger improvement on ethnicity.

The first part of the hypothesis was supported by our results, leading to acceptance that our method consistently reduces bias on generated images.

The second part of the hypothesis was validated through human comparison between images from the original model *versus* our method, in order to spot which elements of the image were altered and how those elements changed.

We observed minor changes in the background of images, textures or logos. Still, all the core shapes and colors were kept very close to the original model.

Figures 6.1, 6.2 and 6.3 display a few samples to exemplify semantic consistency from our method. On the top row are displayed the images generated by the original SDXL model and on the bottom row are the images generated using our method - both generated with the same seed.

In the case of firefighters on Figure 6.1, we can observe the consistency of background elements such as fire, trucks or fog, as well as consistency on the shape of clothes and colors. The main changes on the image are in relation to the gender and ethnicity, which was the desired outcome.

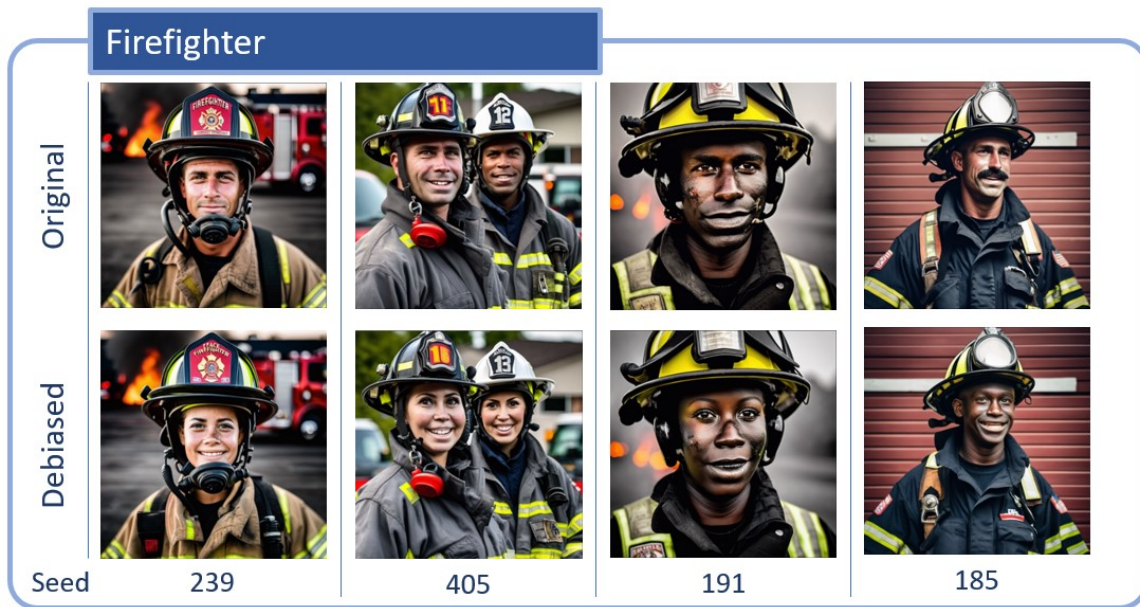


Figure 6.1 – Semantic changes between original SDXL and our method for firefighter concept.

For nurses on Figure 6.2, we can observe very similar background, even though some elements are altered such as chairs or hospital beds, but general shapes and colors are kept. The clothes and body pose are consistent, just like the angle, lightning and camera conditions on the picture.

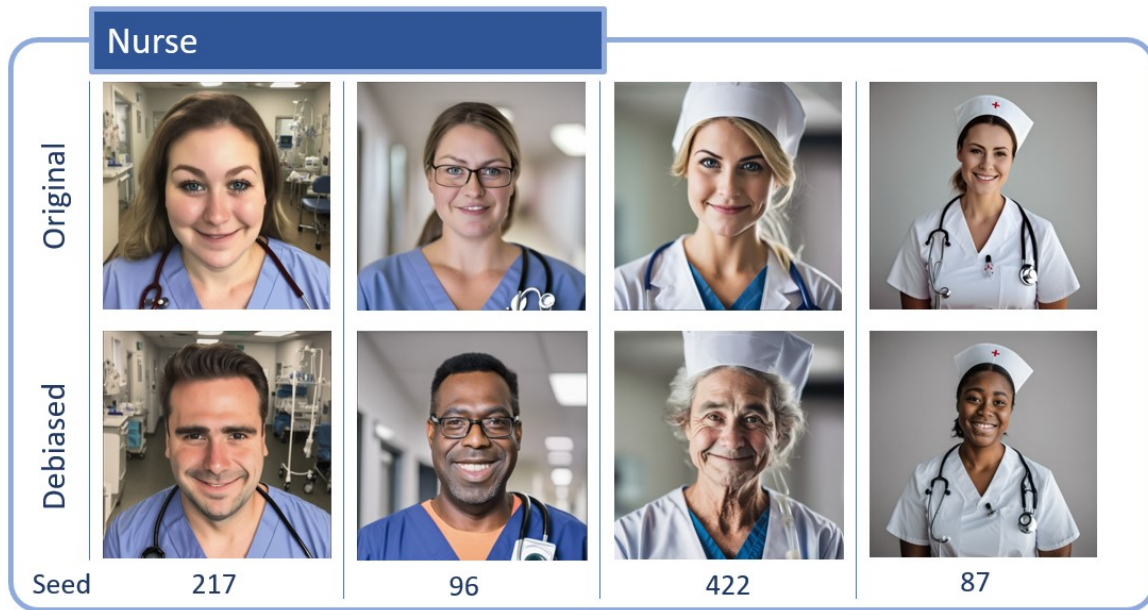


Figure 6.2 – Semantic changes between original SDXL and our method for nurse concept.

Business leaders on Figure 6.3 presented a very similar consistency with the original model. Specially clothes, body poses, picture lightning and angle were barely altered on the analyzed samples.



Figure 6.3 – Semantic changes between original SDXL and our method for Business Leader concept.

The results from this comparison seem to indicate our model's ability to generate images which keep most of the original elements from the SDXL model. The changes are significant on the unwanted biased concept, but much smaller on unrelated concepts. This successfully supports the second part of our hypothesis, leading us to accept it.

6.2 Quantitative Similarity Experiment

This experiment produced a valuable measurement, in theory free of potential human biases. Additionally, the results support the same conclusions drawn on the prior evaluations. That is: our model indeed keeps a high level of consistency to the original SDXL, modifying the images as little as needed to mitigate the bias. Moreover, our model presents the highest fidelity to original SDXL among other alternatives.

Starting by the generated grid as in Figure 5.1, it becomes evident how our model keeps the vast majority of the elements in the same position, shape, colors, lightning conditions and even expressions. The modifications driven out of the other methods often change completely the original image, revealing the lack of control over the latent representations.

When it comes to the modified elements of the image, our model is precise on which biases need to be adjusted. We can observe the desired diversity in gender, ethnicity or age on specific human traits. The other methods seem to introduce a bias in the opposite gender direction, rather than mitigate it.

Moving into the calculated similarities, our model presented significant improvement over other methods. The average similarity of each image (95.15%), presented in Table 5.10, was superior by almost 10 percentage points against other methods, supporting the consistency driven by our approach. The 0.017 standard deviation in similarity of our method was more than 3 times lower than other methods, once more supporting the consistency.

On the results of average images from Table 5.11, we observe similar magnitudes on the percentages. This is expected once the average pixel values in the image will converge to the same colors and general shapes of a given concept. In other words, the nuances are mostly discarded during this analysis. Still, we wanted to evaluate if even in the average pixel space our model was also more similar than the other methods.

We gladly found that our model hit a 99.93% similarity, indicating once more the higher level of control over how the images are modified. This result can be understood as our method’s ability to modify only 0.07% of the average image and still effectively mitigate the bias.

The last part of this experiment was around the features extracted via a ResNet-50 presented on Table 5.12. These features can represent an embedding of the semantic concepts. As most of the semantics in the image should stay the same, high similarities indicate a more efficient bias mitigation.

Our method overcame the alternatives by 5 percentage points on the second best method and by 9 percentage points on the least effective method. The standard deviation in similarity of semantic features was on 0.015, essentially 3 times less than the second best method. These results also support the conclusions from all prior evaluations around the consistency of our method.

In a summary, the first evaluation supported the first part of the hypothesis that our method can effectively mitigate biases, as indicated by human perception. While this second experiment measured and supported the second part of the hypothesis, that our model maintains a high level of control during the debiasing process and alters only necessary elements.

6.3 Bias Selector Experiment

Lastly, we took an extra step into the third exercise. This bias selector worked autonomously and effectively understood which gender bias should be avoided in each of the concepts explored.

The similarity values between the biases left a narrow decision boundary, as described in section 5.3 of chapter 5. The values are very close and that could be the result of a process that is not yet optimized. Ideally we want to see values further apart to create a larger decision boundary, but the selector still works if the precision includes at least 3 decimal places.

So far we explored a bias selector in the embedding space. This same concept could be implemented at the core of the denoising process, exploring bias similarity in the latents. That is left for future work on this research.

7. CONCLUSION AND LIMITATIONS

In a world that is rapidly adopting generative models into our daily life, images are no exception. Image generation models are freely available on the internet and being used to feed a multitude of applications, such as social media, advertising or entertainment.

This leads into an overflow of biased images landing into the internet across the next few years. If we reasonably assume that these biased images will likely be part of the training dataset for new models, at worst we potentially snowball the bias effect and at best we perpetuate our current biases.

This work is a first step into leveraging native mechanisms of Diffusion Models with the goal of making image generation more fair.

Our method facilitates a short-term debiasing solution as it can be easily applied during inference. This solution does not extend the training process and is robust to handle the current biases originated from datasets used to train large text-to-image models.

In results we demonstrated how efficiently we can debias the concepts, while preserving the capabilities of the model. We improved the control over the latent space, which resulted in a much more precise modification of the elements into pixel space. We understand that our delicate influence to guide the generation process presents more value than brute force methods applied at once.

We presented quantitative analysis and qualitative analysis supporting our initial hypothesis that the most efficient way to control the bias on diffusion models consists of working at the core of the denoising process. Specifically, we showed how our method is superior in comparison to current alternatives in terms of image consistency and how effective it is as a debiasing mechanism.

Extending the usage of our method, it can also be used as mechanism to influence the image. Much like the purpose of the Negative Prompt, but producing more consistent results to the original concept. To extend this feature, the user simply needs to indicate the concept to be avoided as a text input, just like the Negative Prompt method. In terms of usage, very easy to replace the current method.

Finally, we demonstrated how our method can leverage an automated bias selector in order to achieve a truly autonomous debiasing mechanism - a capability which no other method proposes so far. This automated selector can debias any protected attribute as long as the references are configured in the model. That is: once we provide examples of concepts (such as "male" and "female"), the model has a reference to what can be found on the gender dimension and can autonomously measure similarity to detect which bias is present. The same can be done for ethnicity by providing multiple options such as "caucasian", "latino", "african-american", "asian" and so on.

7.0.1 Limitations

The main cost of our method is that it slows inference in about 40%. This is due to the fact that our method needs to predict an additional noise on the denoising process - the *bias concept noise* - which is essentially another run on the U-Net.

This method demonstrated good results, but it is limited to our scope of diffusion models, in specific applied to Stable Diffusion. Other generative text-to-image models based on VAEs or GANs will not support this approach. In contrast, Diffusion Models have recently become the predominant model being applied for image generation. While Stable Diffusion is in the front line of open source research and application, popular paid models like Dall-E 3 present on OpenAI's ChatGPT and MidJourney also leverage diffusion models in the background.

At last, we understand that this research paves the way for a robust and autonomous debiasing application, but there are optimizations not yet implemented during our work that could potentially improve the inference time and usage of our method.

REFERENCES

- Balaji, Y. Nah, S. Huang, X. Vahdat, A. Song, J. Zhang, Q. Kreis, K. Aittala, M. Aila, T. Laine, S. Catanzaro, B. Karras, T. and Liu, M.-Y. (2023). ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*.
- Bianchi, F. Kalluri, P. Durmus, E. Ladhak, F. Cheng, M. Nozza, D. Hashimoto, T. Jurafsky, D. Zou, J. and Caliskan, A. (2023). Easily accessible text-to-image generation amplifies demographic stereotypes at large scale. *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*.
- Fawcett, T. and Provost, F. (2013). *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*. O’Reilly Media.
- Friedrich, F. Brack, M. Struppek, L. Hintersdorf, D. Schramowski, P. Luccioni, S. and Kersting, K. (2023). Fair diffusion: Instructing text-to-image generation models on fairness. *arXiv preprint arXiv:2302.10893*.
- Goodfellow, I. Bengio, Y. and Courville, A. (2016). *Deep Learning*. MIT Press.
- Goodfellow, I. Pouget-Abadie, J. Mirza, M. Xu, B. Warde-Farley, D. Ozair, S. Courville, A. and Bengio, Y. (2014). Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.
- Ho, J. Jain, A. and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Proceedings of NeurIPS 2020*.
- Karras, T. Laine, S. and Aila, T. (2018). A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*.
- Karras, T. Laine, S. Aittala, M. Hellsten, J. Lehtinen, J. and Aila, T. (2019). Analyzing and improving the image quality of stylegan. *arXiv preprint arXiv:1912.04958*.
- Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114v11*.
- McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*.
- Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.
- Nichol, A. Q. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. *Proceedings of the 38th International Conference on Machine Learning*.

- Podell, D. English, Z. Lacey, K. Blattmann, A. Dockhorn, T. Müller, J. Penna, J. and Rombach, R. (2023). Sd-xl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*.
- Rando, J. Paleka, D. Lindner, D. Heim, L. and Tramèr, F. (2022). Red-teaming the stable diffusion safety filter. *arXiv preprint arXiv:2210.04610*.
- Rezende, D. J. Mohamed, S. and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.
- Rombach, R. Blattmann, A. Lorenz, D. Esser, P. and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. *arXiv preprint arXiv:2112.10752*.
- Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*.
- Schramowski, P. Brack, M. Deiseroth, B. and Kersting, K. (2023). Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Shen, X. Du, C. Pang, T. Lin, M. Wong, Y. and Kankanhalli, M. (2023). Finetuning text-to-image diffusion models for fairness. *arXiv preprint arXiv:2311.07604*.
- Song, J. Meng, C. and Ermon, S. (2020). Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.
- Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Proceedings of NeurIPS 2019*.
- Tan, P.-N. Steinbach, M. and Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley Professional.
- Vahdat, A. Xiao, Z. and Kreis, K. (2021). Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*.
- von Platen, P. Patil, S. Lozhkov, A. Cuenca, P. Lambert, N. Rasul, K. Davaadorj, M. and Wolf, T. Diffusers: State-of-the-art diffusion models.
- Wolfe, R. Yang, Y. Howe, B. and Caliskan, A. (2023). Contrastive language-vision ai models pretrained on web-scraped multimodal data exhibit sexual objectification bias. *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*.
- Zhou and Chellappa (1988). Computation of optical flow using a neural network. *Proceedings of the 1988 IEEE International Conference on Neural Networks*.

ATTACHMENT A – Grid of Images for each Method

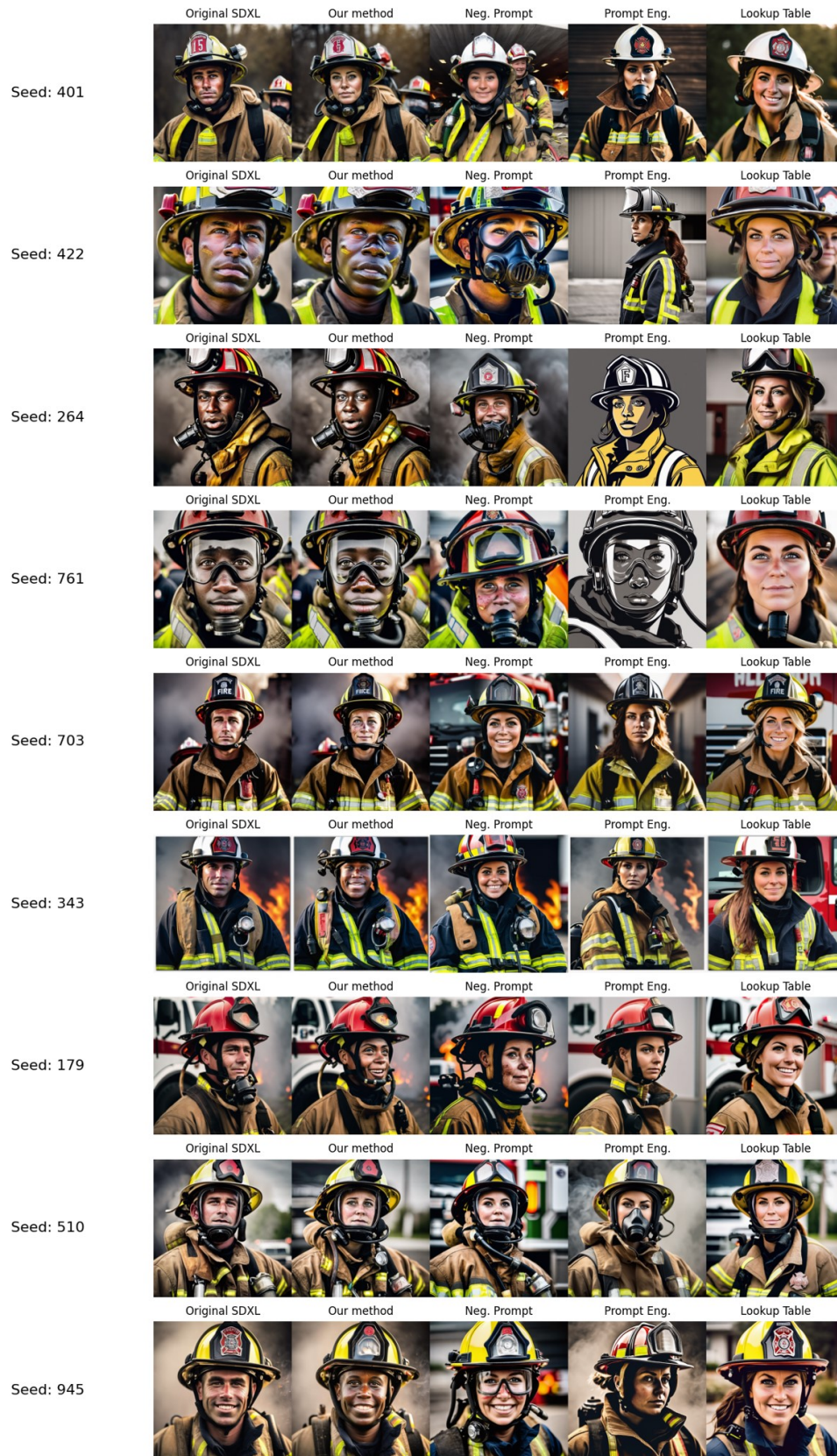


Figure A.1 – Samples of the generated grid of images.



Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria de Pesquisa e Pós-Graduação
Av. Ipiranga, 6681 – Prédio 1 – Térreo
Porto Alegre – RS – Brasil
Fone: (51) 3320-3513
E-mail: propesq@pucrs.br
Site: www.pucrs.br