

TDK-dolgozat

Máté Meglécz

2024.



Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Artificial Intelligence

Synthetic Facial Image Generation Using DDPM: Enhancing Precision with Attribute Conditioning

Scientific Students' Association Report

Author:

Máté Meglécz

II. évfolyam

Advisor:

András Kalapos,

doktorandusz

dr. Bálint Gyires-Tóth,

egyetemi docens

2024.

Contents

Kivonat	i
Abstract	ii
1 Introduction	1
2 Related Works	3
2.1 Deep Learning-based Generative Models	3
2.2 Controlled Image Synthesis Using TL-GAN	5
2.3 Conditioning Diffusion Models via Attributes and Semantic Masks for Face Generation	6
2.4 Diffusion Model Conditioning Techniques	6
3 Dataset	8
4 Methods	10
4.1 Denoising Diffusion Implicit Model [†]	10
4.1.1 DDIM Sampling	10
4.1.2 DDIM Inversion	11
4.2 Conditional Latent Denoising Diffusion Model [†]	12
4.3 Condition Modification	14
4.4 Proposed Post Conditioning Methods	15
4.4.1 Start Latent Editing	15
4.4.1.1 Latent Vector Retrieval	16
4.4.2 Latent Classifier Guidance	17

4.4.2.1	Intermediate Latent Dataset Creation	19
4.4.2.2	Intermediate Latent Classifier Training	19
4.4.2.3	Algorithm Adaptation	20
4.4.3	Combined Method	21
4.4.4	Attribute Classifier for Evaluation	21
5	Experiments	22
5.1	Exp. 1: Generated Image Quality Evaluation	22
5.2	Exp. 2: Attribute Correspondence Eval.	23
5.3	Exp. 3: Built-In Condition Modification	23
5.4	Exp. 4: Post Conditioning Techniques Eval.	24
6	Results	25
6.1	Exp. 1: Generated Image Quality Evaluation	25
6.2	Exp. 2: Attribute Correspondence Eval.	26
6.3	Exp. 3: Built-In Condition Modifications	27
6.4	Exp. 4: Post Conditioning Techniques Eval.	28
6.4.1	Conditioning on Smiling	29
7	Discussion	31
7.1	Comparison of the Proposed Methods	31
7.2	Comparison with Popular Text-Conditioned Models	32
7.3	Possible Applications	33
7.4	Future Work	33
8	Summary	34
	Acknowledgements	36
	Bibliography	37
	Appendix	40
A.1	Implementation Details	40

A.1.1	Hardware [†]	40
A.1.2	Software [†]	40
A.1.3	VQ-VAE Training Details	41
A.1.4	LDM Training Details	41
A.2	Additional Figures	41

Kivonat

A mélytanulás alapú képgeneráló modellek megjelenése lehetővé tette magas minőségű képek szintézisét szöveges leírások vagy specifikus attribútumok alapján, ezáltal forradalmasítva a kreatív területeket. A szórakoztatástól az orvosi képalkotásig egyre elterjedtebbek ezen modellek, páratlan lehetőségeket kínálva a személyre szabásra és a hatékonyság növelésére.

Ez a dolgozat egy olyan módszert mutat be, amely képes arcképek generálására attribútumok alapján mély generatív modellek segítségével, különösen a Zajtalanító Diffúziós Valószínűségi Modellek (Denoising Diffusion Probabilistic Model, DDPM) felhasználásával. Az attribútum vezérelt arckép szintézis kulcsfontosságú lehet olyan alkalmazások számára, ahol szükséges a pontosság a generálás során. Ilyen alkalmazás például a fantomképek generálása gyanúsítottak azonosítására a bűnüldözésben, valamint egyéb területeken a személyre szabott avatár generálás.

Ellentétben a szöveg alapú diffúziós modellekkel, amelyeknek gyakran nehézséget okoz követni a pontos arc jellemzőket, a javasolt módszer közvetlenül a részletes fix attribútumokra épít, ezzel biztosítva a nagyobb precizitást és következetességet a kimenetben, emellett a megoldás lehetőséget teremt a generált kép iteratív finomításra, ezzel növelve a rugalmasságot a képgenerálásban.

Mivel a bűnüldözési adatbázisokon adatvédelmi szempontok miatt módszeremet nem tudom bemutatni, ezért a megoldást nyilvánosan elérhető adathalmazok, mint például a CelebA-HQ [13] segítségével validálom.

Abstract

The rise of deep learning-based image generation models has revolutionized creative fields by enabling the synthesis of high-quality visuals from textual descriptions or specific attributes. These models are increasingly being utilized in diverse applications, from entertainment to medical imaging, offering unprecedented opportunities for personalization and efficiency.

This research introduces a method for generating synthetic facial images conditioned on specific attributes using deep generative models, particularly Denoising Diffusion Probabilistic Models (DDPM). Attribute-controlled facial image synthesis is crucial for high-precision applications, such as generating facial composites or criminal identification photographs (mugshots) for identifying suspects by the law enforcement, as well as in fields like personalized avatar generation.

Unlike text-based diffusion models, which struggle with accurately representing facial features, the proposed method directly conditions on detailed fixed attributes, ensuring greater precision and consistency in the output and allows iterative refinement on the generated images, providing enhanced flexibility in image synthesis.

Due to privacy concerns with law enforcement datasets, I validate my method using publicly available datasets like CelebA-HQ [13].

Chapter 1

Introduction

Deep learning-based image generation has revolutionized industries in recent years. State-of-the-art solutions are capable of generating high-quality images. The application of these models can range from data enrichment in medical fields to image generation for creative tasks.

Variational autoencoders (VAE) [11] and generative adversarial networks (GAN) [3] have shown promising results in custom image generation tasks but they fall short in terms of balancing image quality, diversity, and stability during training compared to denoising diffusion probabilistic models (DDPM) [7]. VAEs often generate lower-quality images due to their reliance on a probabilistic latent space, which can lead to the loss of fine details. GANs, while capable of producing sharp and realistic images, are prone to issues like mode collapse and require complex adversarial training dynamics, making them challenging to train effectively. DDPMs, on the other hand, achieve superior image quality through a stepwise denoising process that naturally captures the distribution's details without the adversarial challenges, leading to more diverse and stable outputs. [†]

For precise application of these methods in specific tasks, conditioning on some form of input is required, so the user can guide the model towards the desired output. Common model conditioning types include class, attribute, image, or text conditioning. DDPMs excel in such conditioning scenarios due to their flexibility in incorporating conditioning information directly into the diffusion and denoising steps. For example, conditioning can be applied via concatenating input embeddings (e.g., class or text embeddings) to each timestep of the denoising process, or through cross-attention mechanisms [26], which effectively guide the model towards generating outputs that align with the desired conditioning. This allows DDPMs to achieve highly controlled and specific outputs, making them particularly suitable

for tasks like image inpainting, text-to-image generation, and attribute-guided generation. In comparison, conditioning in VAEs or GANs often involves modifying the latent space or using auxiliary classifiers, which can be less effective and harder to integrate consistently across different tasks. [†]

Large pre-trained conditional DDPMs, such as Stable Diffusion (open source) or Dall-E (closed source), are often conditioned on text inputs using some kind of text embedding format (e.g. CLIP [18]). Although conditioning on a natural language description of the desired output is undoubtedly a flexible solution, it often lacks the required precision or knowledge that is needed for some domain-specific tasks, such as the generation of realistic and detailed facial composites. In such cases, although text conditioning offers flexibility, a model trained on specific conditions, such as a binary attribute vector, is more suitable. This approach prioritizes the precision and reliability required for domain-specific tasks over the variable accuracy of prompt-generated samples.

In this work, I train a diffusion model capable of generating high-quality facial images conditioned on fixed attribute labels. Furthermore, I attempt to address the flexibility issue regarding the fixed attribute labels. I experiment with two post conditioning methods for the expansion of the conditioning ability of a trained model. First, I adapt a conditioning technique inspired by TL-GAN [4], then I implement a DDPM conditioning technique called classifier guidance into a Latent Diffusion Model and finally I combine these two solutions. I evaluate the trained model’s image generation and conditioning abilities as well as the performance of the post conditioning techniques.

In Chapter 2 the related works are introduced describing all the necessary knowledge for further reading. Chapter 3 describes the CelebA-HQ [13] dataset’s characteristics, limitations, and application in this work. Chapter 4 delves into the details regarding the methods applied throughout my research, introduces the latent diffusion model, then a method for guiding the model, through editing the start latent from which the sampling happens, the latent classifier guidance method and the trainings required for them. Chapter 5 describes the execution and settings of the experiments for the evaluation of the methods. Chapter 6 reports on my observations and the performances of the proposed methods. In Chapter 7 the results of the techniques are discussed, the limitations of the text-conditioned models compared to my solution are demonstrated, practical application possibilities and future work is described. Finally, Chapter 8 summarizes the content and results of my work.

Note: paragraphs of the work which were phrased or revised by a large language model are marked with the “[†]” symbol.

Chapter 2

Related Works

2.1 Deep Learning-based Generative Models

Generative models are part of machine learning where the task is to learn a representation of the training data so that new data can be created with it. They are usually built on deep neural networks. Their strength is that they can be used to generate realistic “fake” data. They are widely applied in tasks such as text generation, image generation (e.g. for artistic purposes), or data enrichment.

The three most common methods for image generation are the generative adversarial network [3], the denoising diffusional probabilistic model [7] and the variational autoencoder [11].

Variational autoencoders (VAE) [11] consist of an encoder, which compresses input data into a latent space, and a decoder, which reconstructs data from that latent space. VAEs use a variational approach to approximate the posterior distribution of the latent variables, imposing a prior (often Gaussian) to ensure smoothness and continuity in the latent space. This enables VAEs to generate new samples by sampling from the learned latent distribution. A key feature of VAEs is their ability to balance reconstruction accuracy with the diversity of latent variables, making them well-suited for generating data, compressing information, and interpolating between samples.

Generative adversarial network (GAN) models come in many different versions [3], I will only give a general description here. GANs are built from two sub-models, a generator and a discriminator. The generator model is trained to generate new data and the discriminator is trained to decide whether its input is

the data produced by the generator or real data from the training dataset. The two sub-models constantly compete with each other during training, the goal of the generator is to fool the discriminator, and the goal of the discriminator is to recognize the generated data.

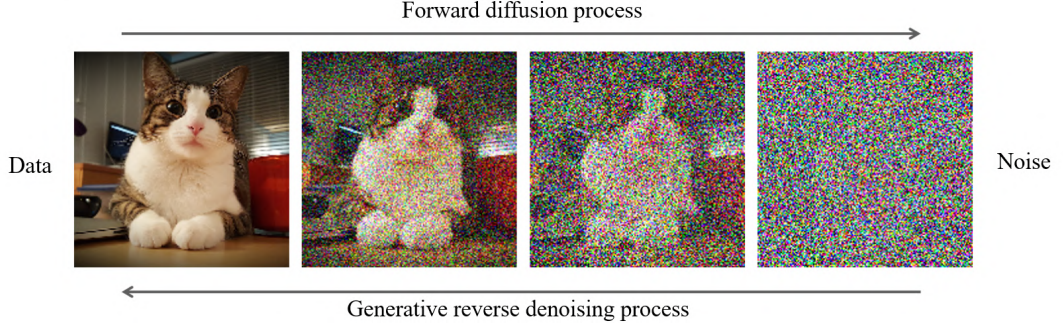


Figure 2.1: The forward and backward processes demonstrated. (source: [15])

Denoising diffusion probabilistic models (DDPM) are currently considered as the state-of-the-art solutions for image generation tasks. Its main goal is to transform a distribution of pixels into another distribution of pixels using a Markov chain. It consists of two main parts, the forward and the backward diffusion process. The U-net model [20] is commonly used as the backbone of the DDPM model.

In the forward process, it gradually adds more and more noise to the image until it is completely unintelligible to the human eye. During this process, the complex and unknown distribution of the training data set is transformed into a simpler latent distribution.

The backward process starts from the “corrupted” image created by the previous forward process. Its goal is to remove the noise from the image step by step so that the original distribution is recovered as closely as possible by the end.

During the forward process, the probability density functions (PDF) of the noise that distorts the images during the steps differ from each other. In the backward process, a deep learning solution is employed to determine the PDF parameters (mean and variance) of the forward steps. The operation of the two processes is illustrated in Figure 2.1.

This solution is significantly more stable during learning and sampling than other approaches for synthesis tasks such as GANs or VAEs.

2.2 Controlled Image Synthesis Using TL-GAN

Transparent Latent-space GAN (TL-GAN) [4], as proposed by Shaobo Guan, offers a novel approach to controlled image synthesis and editing. This work addresses a key challenge in generative modeling: the ability to provide fine-grained control over specific attributes of generated images, such as facial features, expressions, or age. Unlike many previous GAN-based approaches, TL-GAN allows users to interactively adjust features using a single network, without requiring retraining to add new control axes. This efficiency makes TL-GAN particularly advantageous for applications involving iterative creative design and real-time image customization.

†

TL-GAN builds upon Nvidia’s pg-GAN model [10], which is capable of generating high-resolution photorealistic images. By making the latent space “transparent”, TL-GAN discovers meaningful feature directions that correspond to different facial attributes. The core idea is to first train a feature extractor, which is a CNN trained for attribute classification on existing labeled image datasets. Then sample images by the GAN from random noise latent representations and make predictions with the trained CNN on them. Now it is possible to correlate these predicted labels with the initial noise latent vectors from which the classified image was generated by the GAN.

This approach allows the latent space to be intuitively manipulated, enabling continuous control over attributes such as age or gender. Such transparency in the latent space offers a significant improvement over traditional GAN models, which typically lack this level of interpretability and controllability. With a simple linear regression model, the author identifies the vector corresponding to a specific feature. Simply applying this directional vector to the latent representation would guide the sampling model toward generating the feature corresponding to the added vector. However, the simple application of these vectors might result in side effects coming from the natural correlation of facial features, e.g. having a beard is correlated with being a man, thus applying the “beard” vector to a latent vector would result in a woman, might have the side effect of changing the gender of the person. Disentangling correlated feature axes by making them orthogonal to each other solves this problem, reducing possible side effects.

The flexibility of TL-GAN extends to the addition of new tunable features. By training a feature extractor on a labeled dataset and correlating the extracted features with the latent vectors, new feature axes can be defined without retraining

the GAN model itself. This process is highly efficient, takes less than an hour on a single GPU, and allows for rapid expansion of the model’s capabilities.

2.3 Conditioning Diffusion Models via Attributes and Semantic Masks for Face Generation

Lisanti et al. [12] introduces conditioning into Latent Diffusion Models (described later in Section 4.2). Their novel strategy for enhancing diffusion models through multi-conditioning, harnessing cross-attention mechanisms [26] to utilize multiple feature sets is not only capable of generating conditioned images from binary attribute labels but also be conditioned on segmentation masks at the same time for higher control. They demonstrate their work on the CelebA-HQ [13] and Deepfashion [14] datasets.

Their work generates images at a lower resolution and a different conditioning attribute set than mine so it is not directly comparable with my work.

2.4 Diffusion Model Conditioning Techniques

Diffusion models often rely on conditioning information to guide the generative process toward a desired output. Each has its strengths and weaknesses, making them more suitable for different types of tasks and conditioning inputs. [†]

Cross-attention mechanisms [26] are sophisticated tools for incorporating complex, variable-length conditioning information, such as text prompts or detailed attributes, into diffusion models. By dynamically focusing on relevant parts of the conditioning input at each step of the generation process, cross-attention allows for nuanced and flexible guidance of the model, making it particularly well-suited for applications such as text-to-image generation. Although cross-attention is computationally intensive due to the need to compute attention weights for each conditioning input, it significantly enhances the model’s ability to align generated content with desired conditions, leading to high-quality, contextually appropriate outputs. [†]

Time embedding concatenation is a simpler and more computationally efficient conditioning method in diffusion models, particularly useful for scenarios where the conditioning input is fixed or scalar, such as timestep or class label conditioning.

In this approach, an embedding vector representing the conditioning information is directly concatenated or added to the model’s features, typically at each timestep of the diffusion process. This technique is effective for straightforward conditioning, where computational efficiency is critical, and the conditioning does not require complex, dynamic relationships to be learned. Unlike cross-attention, time embedding concatenation offers a static influence on the generative process and is easier to implement, though it lacks the dynamic, nuanced control found in cross-attention.[†]

Text-conditioned diffusion models have emerged as powerful generative tools, enabling the synthesis of high-quality images and other data modalities conditioned on textual prompts. Building upon foundational work in diffusion models for image synthesis, text-conditioned models integrate textual embeddings (often derived from large language models or pre-trained embeddings such as CLIP [18]) to guide the diffusion process, aligning generated outputs with the semantic meaning of the input text. Cross-attention mechanisms are commonly employed in text-conditioned diffusion models to ensure that specific parts of the generated image align well with the text prompts, leveraging the semantic depth of textual embeddings to refine the generation process at each stage.[†]

ControlNet [27] is a specialized conditioning method designed to provide even finer control over diffusion models, particularly for applications that require precise adherence to a desired structure, such as image-to-image translation or controlling generated images with edge maps or pose guidance. ControlNet works by extending the architecture of the diffusion model with a controllable branch, which allows the model to condition its output on additional guidance signals beyond what traditional conditioning techniques can achieve.[†]

Chapter 3

Dataset

For the demonstration and reproducibility of my work, a dataset had to be chosen that is publicly available, has a large enough sample size with high-quality, real, facial images, and is sufficiently labeled for conditional use cases. Facial data by its nature is highly sensitive, so even though the core idea of my work comes from using conditional facial image generation in, for example, law enforcement scenarios, those criminal datasets are not suitable for the evaluation and publication of my work.

CelebA-HQ [13] is a commonly used dataset for the training and evaluation of image generative models. The dataset contains 30,000 high-quality images of celebrity faces, derived from the original CelebA dataset. CelebA-HQ maintains a resolution of 1024×1024 pixels, which makes it particularly well-suited for tasks requiring high visual fidelity. The dataset also features extensive annotations, including 40 binary attribute labels which makes it ideal for conditional use cases, where these attributes can serve as conditioning information for generative tasks.

A benefit of CelebA-HQ is its widespread use in the research community, allowing for a fair comparison of generative models against established benchmarks. Many notable image synthesis models have used CelebA-HQ as a standard dataset for evaluating the effectiveness of their approaches, such as GAN-based models and diffusion models, making it a natural choice for the demonstration and reproducibility of the current work.

However, it is important to note that despite its popularity, CelebA-HQ has its limitations. It is a curated collection of celebrity faces, which means it lacks the diversity seen in real-world populations. This inherent bias should be acknowledged, especially when attempting to generalize results to broader applications, such as the generation of criminal identification photographs (mugshots) mentioned earlier. Therefore, while CelebA-HQ serves as an effective demonstration dataset, it cannot

fully encompass the variability required for highly sensitive applications involving criminal data.

Although the dataset contains 40 attribute labels, they are not necessarily adaptable to other domains. Of the 40 binary attributes, only about half of them can be linked to a person’s natural features such as facial geometry, hair type or color, etc. One could describe the rest of the features as “unnatural” features like the presence of different types of clothing or accessories such as “wearing a hat” or “heavy makeup”. Besides these, there are some attributes that can be linked to expressions (i.e. smiling) or the image’s quality (i.e. blurry).

Another limitations of the dataset is that it lacks crucial natural feature annotations like skin tone or eye color that are required for the accurate description of the target facial image. Besides this, features can be highly correlated with each other (e.g. having gray hair and being not young) or even display a causal relation (e.g. having no beard being true and having a goatee cannot happen at the same time). For further correlations see Figure A.2.1 in the Appendix.

To address these limitations in part, careful consideration has been given to how the dataset’s labels and conditioning attributes are utilized. I have chosen 19 attributes that are descriptive of a person’s natural features and completely excluded accessory or clothing descriptions from the diffusion model conditioning during the training process.

The chosen attributes include gender (Male), age (Young), hair characteristics (Bald, Bangs, Receding Hairline, Black Hair, Blond Hair, Brown Hair, Gray Hair, Straight Hair, Wavy Hair), facial hair features (No Beard, Goatee, Mustache, Sideburns), and additional facial features (Narrow Eyes, Oval Face, Pale Skin, Pointy Nose).

To demonstrate image editing and post conditioning of the model, some non-natural descriptors (e.g. “Smiling”) will also be used due to the limited availability of attribute labels.

The dataset was split into 3 partitions, such as training (27,000 samples), validation (600 samples) and test (2,400 samples) subsets. The train set was used for model training, the validation set for the comparison of models and methods during training, and the test set was left for the independent evaluation of my final solution.

Chapter 4

Methods

This chapter introduces the methods and techniques that are used throughout my work.

4.1 Denoising Diffusion Implicit Model[†]

Denoising Diffusion Implicit Models (DDIM) [24] are a class of generative models that build upon the principles of DDPMs for generating high-quality samples from a given data distribution. DDIMs are particularly known for their efficient and deterministic sampling process, which allows the generation of reproducible samples with fewer diffusion steps compared to DDPMs. By modifying the stochastic nature of the diffusion process to a deterministic one, DDIMs offer a fast, stable, and reproducible alternative for generating images while maintaining the quality of generated samples. [†]

4.1.1 DDIM Sampling

DDIM sampling [24] is a faster, deterministic alternative to the original DDPM sampling, enabling reproducible sample generation with fewer iterations. By parameterizing the reverse diffusion process deterministically, DDIM reduces randomness in each step, achieving efficient sampling with results similar to stochastic DDPMs. While this approach enhances computational efficiency, it may slightly reduce sample diversity due to the absence of stochasticity, making it ideal for scenarios prioritizing speed and consistency without compromising image quality.

4.1.2 DDIM Inversion

DDIM inversion [24] is the process of reversing the sampling steps to recover the original latent representation from a generated image. This technique allows for mapping generated samples back to the latent space, making DDIMs uniquely capable of establishing a two-way mapping between image space and latent space. DDIM inversion is useful in tasks that require a deep understanding of the generative process, such as image editing, reconstruction, and interpretability studies.

The inversion process involves running the deterministic reverse diffusion process backward, which allows the model to approximate the original latent code that produced a given image. By doing so, DDIM inversion opens up possibilities for manipulating existing images by altering their latent codes and regenerating them through the DDIM sampling process.

DDIM inversion is particularly valuable in applications where iterative refinement of generated content is needed, or where the ability to edit specific features of an image is crucial. The deterministic nature of DDIMs ensures that the inversion process is stable, reproducible, and yields reliable latent representations that can be further used for fine-tuning and conditioning the generation. Figure 4.1 shows how the two processes work and DDIM inversion’s ability to create noise representation from the input image and DDIM sampling’s reconstruction capability from the created noise.

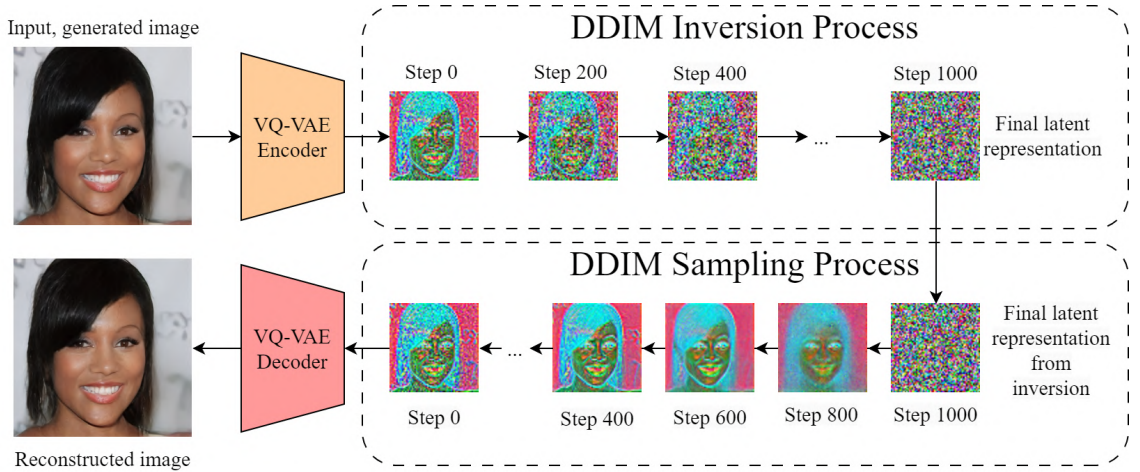


Figure 4.1: DDIM inversion and sampling are demonstrated on a sampled image in my work (VQ-VAE is introduced in Section 4.2). The DDIM inversion process creates a latent noise representation from the original image by adding noise to the image step-by-step. The DDIM sampling process can effectively remove this deterministically added noise to reconstruct the original image.

4.2 Conditional Latent Denoising Diffusion Model [†]

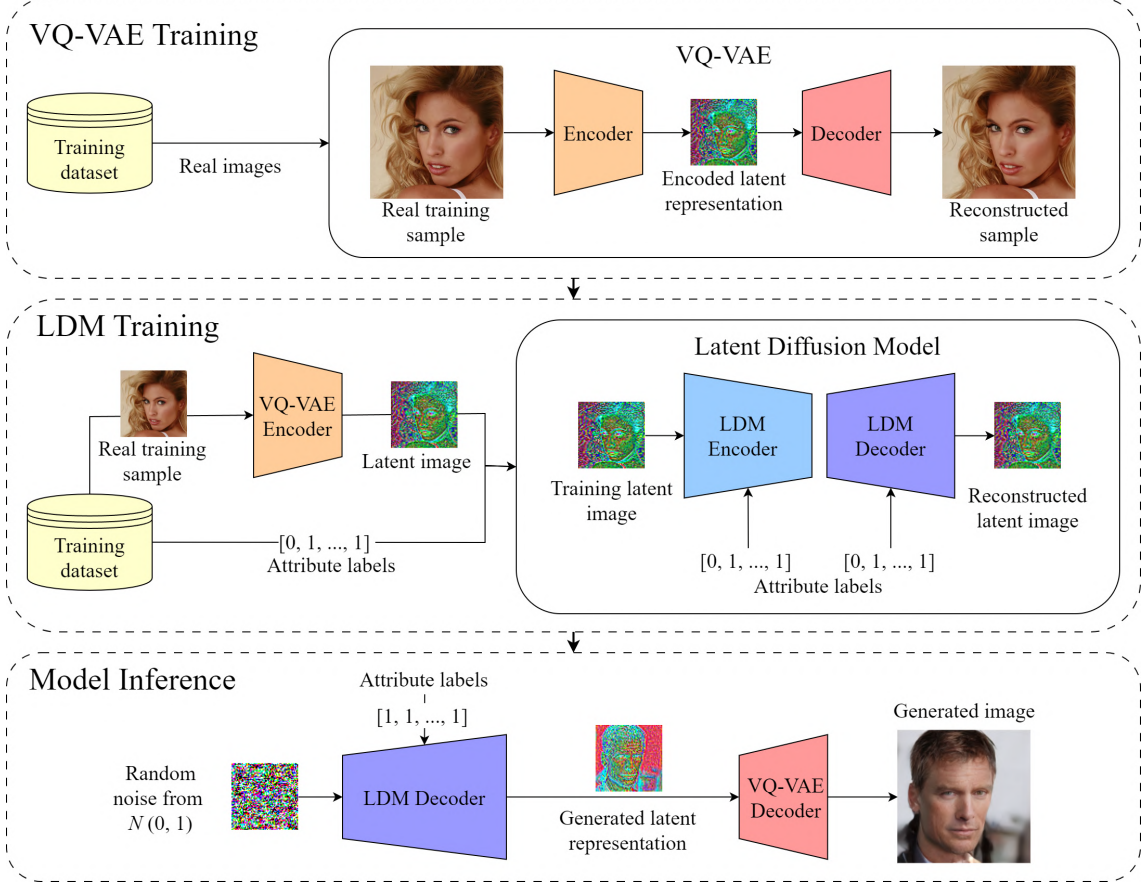


Figure 4.2: Training and inference process for the Conditional Latent Diffusion Model. First, a VQ-VAE model is trained unconditionally to create smaller-sized latent representations from real images. In the second step, the DDPM is trained on the latent representations of the real training images alongside their attribute labels. After both models are trained, random noises are sampled in the resolution of the latent images with the DDPM alongside the desired attribute labels. After the latent representation is created the VQ-VAE decoder creates the final generated image.

In this work, a conditional version of the Latent Denoising Diffusion Model (LDM) [19] was trained. LDMs are particularly effective for generating high-quality images while being computationally efficient for both training and inference.

Latent Denoising Diffusion Models operate on compressed representations of the original images, which significantly reduces the computational burden compared to working directly on high-resolution images. To create these latent representations, an additional model is trained to encode the input images into a lower-dimensional

latent space and then decode these representations back to reconstruct the original images. Variational Autoencoders (VAEs) are commonly used for this purpose. Specifically, a popular choice of VAE architecture in LDM training is the Vector Quantized VAE (VQ-VAE), and this work also follows that approach.

The VQ-VAE compresses the input images into a latent space while preserving key features necessary for high-quality reconstruction, enabling the LDM to focus on learning a diffusion process that generates realistic images from these compressed latents. By conditioning the LDM on additional information, such as class labels or other guidance, the model can generate images that align with specific requirements, leading to greater control and versatility in image generation.

The conditioning is implemented by taking the binary attribute’s vector and embedding it into a fix-sized vector space, creating a lookup table between the attribute vector and the embedding vector representation. The U-net model is conditioned by concatenating this vector representation to the time embedding vector created in each diffusion step.

In this setting, the usage of the cross-attention mechanism would just add a computational overhead, since the conditioning vector is fixed-sized. Another solution for the conditioning could be the adaptation of the Control Net method, however it is aimed toward image-to-image translation task, which would require the embedding of the attribute vectors into the image space. The embedded vector would not contain any spatial information like e.g. a segmentation mask does, and also store the attributes in a much larger embedding than necessary, adding computational overhead.

The dataset used in this work consists of images with a resolution of 1024×1024 . However, using such high-resolution images would result in a latent representation that is too large to manage efficiently. In my experiments, I observed that increasing the number of downsampling blocks in the VQ-VAE led to difficulties in effectively reconstructing the images. To address this issue, I restricted the latent representation to a maximum size of 64×64 , which provided an optimal trade-off between reconstruction quality and computational feasibility, allowing for effective training of the LDM. Specifically, I found that using a maximum of three downsampling blocks was the most effective configuration, which necessitated reducing the input image resolution to 512×512 . This resolution reduction ensured that the latent representation remained computationally manageable while retaining enough image detail to facilitate high-quality generation. Generating 1024×1024 images would still be possible either with further experiments to improve the VAE’s efficiency or with access to greater computational power.

Figure 4.2 demonstrates how the training and inference of the conditional model are executed. Figure 4.3 aims to demonstrate the capability of the trained VQ-VAE model. Implementation details of the LDM are provided in the Appendix in Section A.1 with further samples on Figure A.2.2.

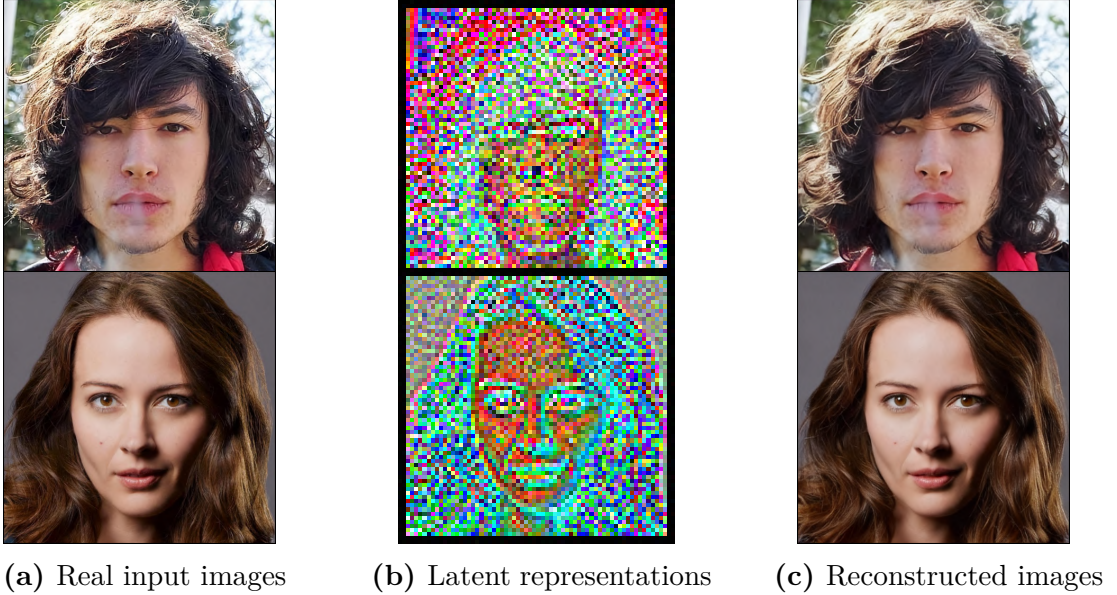


Figure 4.3: Images demonstrating the capability of the VQ-VAE, first latent representations are created from from real images, than the images are reconstructed from the latent representations.

4.3 Condition Modification

In a conditional face generative model, images DDIM sampled from the same noise can be modified by changing the conditioning information. This conditioning allows for precise control over specific attributes of the generated faces, while still retaining a core underlying structure.

The generative process starts with a noise vector, typically sampled from a random distribution (e.g. Gaussian). This noise vector serves as the foundation for generating an image, and without any conditioning, the output would be random but resemble the training data, such as human faces. The key to controlled generation is in the conditioning information, which can guide the model to produce specific desired attributes like hair color, age, or facial expressions. [†]

For example, consider a noise vector used to generate a face conditioned on “young” and “blond hair”. The model will produce an image that fits this description. If the conditioning changes to “old” and “gray hair”, but the noise vector remains the same, the resulting face will still share similarities with the original (due to the shared

noise) but will now appear older with gray hair. This ability to keep certain features consistent while modifying others is what makes conditional generative models so powerful. To ensure consistent and iterative modifications, DDIM sampling is used to eliminate DDPM’s stochastic nature in image sampling.

4.4 Proposed Post Conditioning Methods

The ability to condition a model on attributes that it was not trained to be conditioned on can be beneficial for several reasons.

One common scenario is when there is an unconditioned or text-conditioned pre-trained model, which is trained on millions of images and can generate higher quality images than a model that is feasible to train with data or computational resource constraints. One might take a model like this and would still want to generate condition-specific data.

Another scenario can be when I already have a trained model on some conditions and I want to further condition the trained model on new attributes, for which I am able to acquire labeled data. Even if the new data is labeled for all the model’s built-in conditions plus the new label, I would still need the original training data of our model to be labeled for the new attribute to be able to retrain the model to conditionally generate data according to all the attributes.

Retraining a model or labeling large datasets can be expensive and, time- and resource-intensive, therefore I propose two separate methods for LDMs that can address these issues, with different label and resource requirements.

4.4.1 Start Latent Editing

This method is an adaptation of TL-GAN [4]. In his solution, this technique was used on a pre-trained, unconditional pg-GAN model. In my work, I adapt this method and evaluate it for my conditional LDM model.

This method solves the issue of having a model that is conditioned on a set of attributes, that I would like to extend with new attributes from new data that is labeled not only for the new attributes but also for the model’s built-in conditions. Figure 4.4 shows the overview of the process.

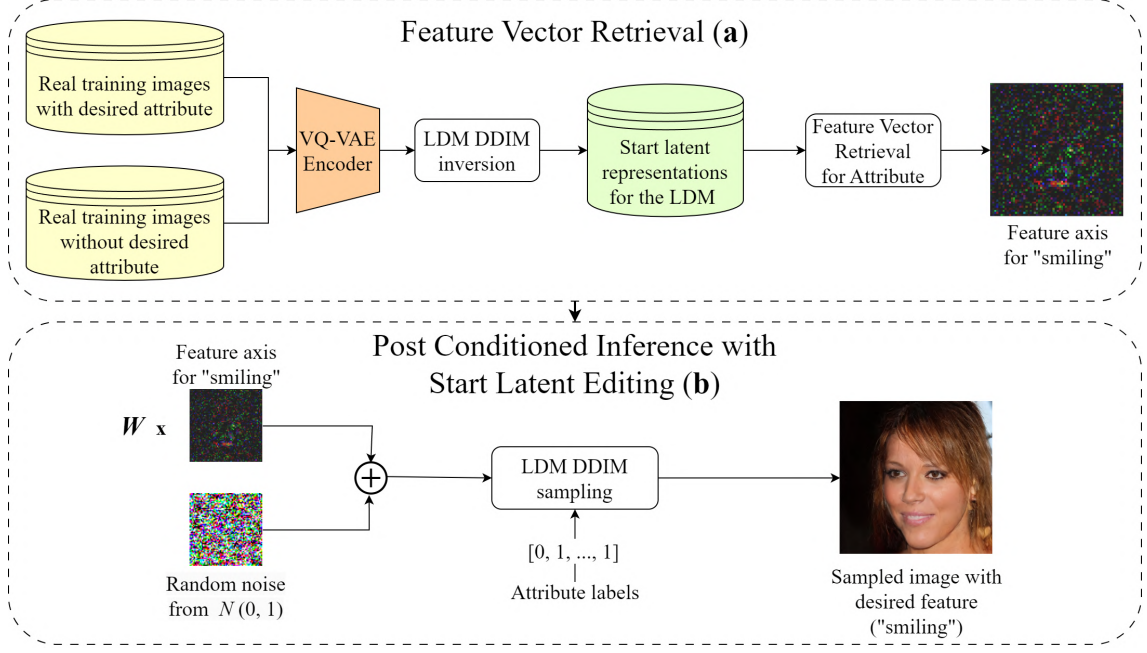


Figure 4.4: The process of retrieving the feature axis for the desired new attribute and its application during the inference of the model. (a) Both images labeled as 1 or 0 for the target attribute are decoded through the VQ-VAE, DDIM inverted with the LDM, and through Linear Regression and linear algebra transformations the feature axis corresponding to the target attribute is retrieved. (b) During the inference, this feature axis is added or subtracted to the initial noise to further condition the generation process.

4.4.1.1 Latent Vector Retrieval

In TL-GAN, the latent vector-attribute matching happens through sampling random noises and predicting the generated image’s attributes with a CNN feature extractor. This technique is not reliable, because of the randomness of an unconditional generative model, it is impossible to tell how many images have to be sampled to yield a sufficient amount of images for a specific attribute and to obtain enough latent vector from which the regressor model can efficiently extract the feature’s latent axes. Furthermore, when using a DDPM as the generative model, the sampling process becomes significantly more time- and resource-intensive compared to GANs, which in turn makes data acquisition slower and more costly.

In my work, I leverage the DDIM inversion to reverse real images into latent representations within diffusion models. Instead of randomly sampling images from the model and labeling them, I can take the already labeled training data and reverse it to create latent representations.

From the final latent representation, I can perform the same technique used by the author of TL-GAN. The algorithm first finds the vector predictive of each individual feature vector in the latent space, this is performed by a linear regression model fitting. Then feature vectors are normalized, distangled, and decorrelated, making them orthogonal to each other. By applying these independent feature vectors with a weight on the first randomly generated noise from which the diffusion model performs the generation process, I expect to be able to condition the model to generate images corresponding to the added feature.

4.4.2 Latent Classifier Guidance

Classifier guidance, introduced by Dhariwal et al. [2], is a technique used to steer diffusion models toward generating samples conditioned on desired properties. In the context of generative models like diffusion probabilistic models, classifier guidance leverages an auxiliary classifier to provide gradients that inform how the sampling process should be modified to generate samples that align with a given label. Essentially, this approach improves sample quality and control over the generated content by adjusting the noise estimates during the reverse diffusion process. [†]

The classifier is trained to predict a label based on partially denoised samples at each diffusion time step. During generation, the gradient of the classifier’s output, with respect to the intermediate sample, is used to modify the noise estimate. By adjusting the strength of the guidance, the model can trade-off between higher sample quality (stronger guidance) and higher diversity (weaker guidance). [†]

Classifier guidance is particularly effective because it allows conditional generation without needing to retrain the underlying diffusion model from scratch. Instead, it utilizes the classifier as a plug-in module that guides generation, making it an efficient way to introduce conditioning into pre-trained diffusion models. However, one challenge is that the need for a separate classifier can add computational complexity during training and inference. [†]

In this work classifier guidance is adapted to work on the latent space and used in the LDM’s DDIM sampling process. This method proposes a solution for adding a new attribute condition to a model from new incoming data. Unlike Start Latent Editing, which needs samples conditioned for all the previous labels, both built-in and post-conditioned, for the addition of every new post-conditioning label, this method only requires samples for the built-in conditions and the desired new attribute.

For every new post-conditioning label a classifier model is trained for the detection of features on the intermediate noise latent images of the LDM.

It is important to note, that the name Latent Classifier Guidance is also used by Shi et al. [22]. Our work shares similarities in using classifier guidance on latent representations of the images, however the used latents and the base model are drastically different. Their work relies on the modification of a semantic subcode, on which a Diffusion Autoencoder [17] is conditioned, while I use a latent diffusion model, optimizing the x_t latent representations of a DDPM. Furthermore, I aim to create a post conditioning method for an already conditioned LDM model. My solution did not take inspiration from their work.

The process of this post conditioning method is visualized and explained in Figure 4.5.

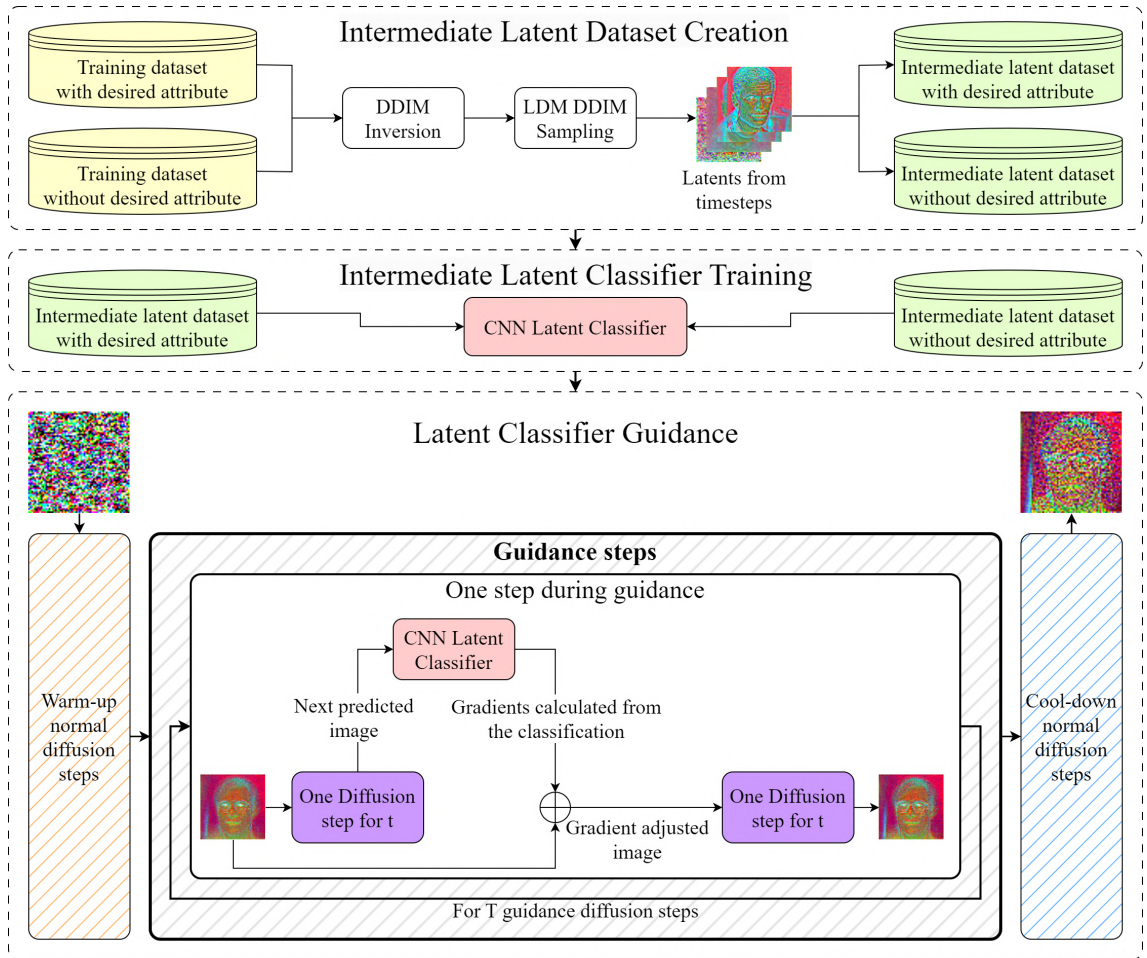


Figure 4.5: The process of classifier guidance as a post conditioning method for LDMs. First a latent image dataset has to be created for the CNN to be able to detect whether the new feature is on the latent or not. The CNN is trained on these latent datasets to classify the images. During the sampling process at every guidance step the next latent is sampled, classified, and the original latent is adjusted with the gradients obtained from the classification. Finally, the next latent is sampled from the gradient adjusted image.

4.4.2.1 Intermediate Latent Dataset Creation

To obtain these intermediate latent representations of the conditional labels there are two possible solutions.

One way to sample images from random noise, save the sampling process’s latents while hoping to obtain images at the end of the sampling that correspond to a label that the LDM was not conditioned on. It is unpredictable how many images would have to be sampled randomly to procure a sufficient amount of samples that fit the target condition.

My solution for this problem is much more computationally efficient. Instead of sampling images randomly, I can use prelabeled images to obtain these latent representations. If a new conditioning label has to be added it is enough to have a dataset of images only labeled on that new condition. These images can be DDIM inverted into a noise-like latent representation, from which the samples can be reconstructed through DDIM sampling. This way the intermediate latents can be obtained from the forward sampling process, which should correspond to the necessary label.

4.4.2.2 Intermediate Latent Classifier Training

The latent classifier is trained on the latent representations in different sampling steps corresponding to a real training image. An equal amount of real images having and not having the desired attribute are sampled randomly from the training dataset. It is important to note that an image’s corresponding latents are very similar especially between following steps since the DDPM works through iteratively removing noise from an image. All of the latents before the final latent, which the VAE decodes, can be considered as a noise-augmented version of that last latent. Although I have a large amount of training latent images (e.g. millions) they are quite similar so during a CNN’s training I have to be especially careful not to overfit the training dataset.

In my work, first I experimented with training a ResNet-18 [5] architecture but it seemed to overfit the training data too quickly thus I have reverted back to a much simpler CNN architecture.

The proposed CNN architecture consists of three convolutional layers with 16, 32, and 64 filters, each using a 3×3 kernel, followed by ReLU activation and 2×2 max-pooling, reducing spatial dimensions to 8×8 feature maps. These are flattened into a 4096-dimensional vector and passed through a fully connected layer with 256

neurons and ReLU activation, leading to a single-neuron output layer for the final classification.

The model is trained for 1 epoch since the latent images corresponding to one real image are highly similar, resulting the model overfitting the training dataset by the end of the second epoch. I have used Adam optimizer with a batch size of 256, a learning rate of 0.001 and Binary Cross Entropy with the prediction and target logits for the loss calculation.

4.4.2.3 Algorithm Adaptation

While the guidance algorithm adds the current gradient vector with a weight to the current latent vector at every step, I have found that finding the optimal weight factor is difficult and the stability of the generation process is highly unreliable, especially if the classifier is also not accurate enough.

In some cases, I have observed sudden spikes in the gradient and in others, the gradient got stuck at 0 with a high loss value, which could mean a local optimum or a plateau in the parameter space. Both observations would lead to the generation of ruined, noisy samples or the lack of expected functionality (i.e. the sample not being properly conditioned). All of this can be a result of a poorly trained latent classifier, but I found that in most cases making modifications to the algorithm can still improve the sampling process.

I have implemented techniques from gradient-based optimization into the guidance algorithm. First I have solved the 0 gradient problem by introducing momentum to the optimization process. Although the application of the method is not necessarily correct since the previous gradients from which the momentum part is created were not computed on the same latent (i.e. it is a different latent since the diffusion model took a step and performed some denoising on the previous latent), I have found that the difference of the latent vectors between steps are so small that the momentum-based method still improves the optimization.

As for the sudden spikes in gradients, gradient clipping was introduced as well as a 3-stage sampling process similar to Hu et al. [8]. The sampling process starts with a warm-up phase, this is beneficial since at the beginning the latents resemble noise more, thus it is more difficult to accurately detect features, which could lead to the classifier producing high loss values leading to the destabilization of the sampling process. The middle stage is where the gradient-based optimization happens. In the third cool-down stage the model is left to refine the samples without the gradient adjustments, in the end, major features are usually already established, or if they

are not it is unlikely to happen, but the gradient adjustments could still ruin the quality of the images so it is better to leave out the gradient modifications.

4.4.3 Combined Method

Finally, in this research, I have combined the two previously introduced post-conditioning methods. First, the feature axis from the start latent editing method is added to the initial noise, then during the sampling process classifier guidance is used to further reinforce that the sampling is guided to the right direction. I expect this method to outperform the previous two by combining their techniques since the two methods can be applied independently at the same time.

4.4.4 Attribute Classifier for Evaluation

For the evaluation of the model’s conditioning ability, I have trained several classifier models. The models are based on pre-trained ResNet-50 [5] models fine-tuned on the training dataset, used in my work, for the classification of the attribute labels. One model is trained for the classification of built-in labels and a separate model for the post conditioning label. They are all trained with the same training hyperparameter settings and are used for the classification of generated samples during the experiments.

I have chosen the ResNet-50 architecture because it has strong classification results on the Imagenet-1k [21] dataset while still being a relatively small model. With the fine-tuning of its pre-trained weights, I only trained the models for 6 epochs, saved the current weights after each epoch, and chose the checkpoint with the smallest validation loss. The models were trained with Adam optimizer, a batch size of 64, learning rate of 0.0001 using Binary Cross Entropy with the prediction and target logit values.

Chapter 5

Experiments

In this chapter, I present a series of experiments designed to evaluate the effectiveness and robustness of the proposed methods across various metrics.

5.1 Exp. 1: Generated Image Quality Evaluation

For evaluation, I used metrics commonly applied in research studies.

Fréchet Inception Distance (FID) [6] measures the similarity between real and generated image distributions by comparing feature embeddings from a pre-trained Inception network. FID models these embeddings as multivariate Gaussians and calculates the Fréchet (Wasserstein-2) distance between them. A lower FID score suggests that generated images are closer to real ones in quality and diversity, making FID more comprehensive than metrics like the Inception Score. [†]

Kernel Inception Distance (KID) [1] also measures the similarity between real and generated image distributions using feature embeddings from a pre-trained Inception network. Unlike FID, KID employs Maximum Mean Discrepancy (MMD) with a kernel function, providing an unbiased estimate that is more reliable for small samples. Lower KID values indicate higher quality and diversity in generated images. [†]

Limitations of the FID and KID Scores. While FID and KID are widely used for evaluation, they come with limitations. These scores are highly domain-specific; for instance, generating flower images produces distinct features and complexity compared to facial images, making cross-domain comparisons unreliable. Even with

the same dataset, models trained at different resolutions may yield inconsistent scores due to resizing distortions to the 299×299 resolution required for the Inception V3 network [25].

Comparing unconditional, conditional, and differently conditioned models adds further challenges. For example, it’s unreasonable to expect similar performance from an unconditional model and a mask-conditioned model, or between models conditioned on different attribute subsets.

Additionally, these metrics require large sample sizes for accurate statistical representation of real and synthetic distributions. The more samples we can use the better we predict the distributions of the real/synthetic data. Unconditional models often require 10-50k samples, which is challenging for conditioned models due to the limited availability of real labeled samples.

Despite these limitations, for the completeness of my work, I report FID and KID scores on my smaller test dataset (2,400 images) and at lower sample sizes to underscore these constraints.

5.2 Exp. 2: Attribute Correspondence Eval.

In my work, I have trained a classifier model for the proper evaluation of the conditional diffusional model’s capabilities. Besides the image quality, it is also important to evaluate the generated images’ correspondence with the conditioning attributes. For this, I use the classifier model described in Section 4.4.4. I have sampled images from the LDM model conditioned on the attribute labels of the test dataset, so the attribute combinations are realistic and the classifier’s metrics on the resulting images are comparable with metrics measured on the real test images. The metrics used to evaluate the success of the conditioning are the classifier’s accuracy, precision, and recall for each individual attribute on the real and generated test data.

5.3 Exp. 3: Built-In Condition Modification

In this experiment, I demonstrate the solution’s editing capabilities using only the labels the model was trained with. For the test dataset’s attributes, an initial noise is DDIM sampled, then attributes are modified and the image is resampled again. This evaluation’s purpose is to demonstrate that while sampling with the same noise with different attribute labels, the samples keep resembling each other.

For measuring the perceptual similarity between two images, I have used the Learned Perceptual Image Patch Similarity (LPIPS) metric [28], which assesses similarity by comparing deep feature representations from a pre-trained network (e.g. VGG [23]) to better capture perceptual differences. For comparison, I measure this metric first on the real test dataset in groups of images, where in each group the metric is calculated between any two images, and the values are averaged. I repeat the same on the generated dataset, where in each group the images are sampled from the same initial noise but with the test dataset’s different conditions. It is important to note that in both scenarios conditioning labels from the test data are distributed the same way between the groups. A lower value means a higher similarity between two images. [†]

5.4 Exp. 4: Post Conditioning Techniques Eval.

Post conditioning techniques are evaluated on the “smiling” attribute with the process shown on Figure 5.1.

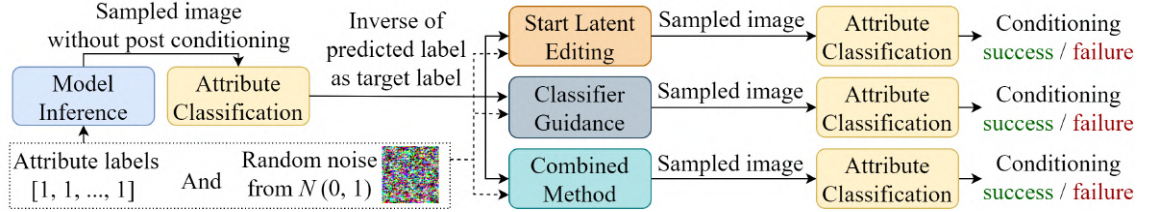


Figure 5.1: The evaluation process for the post conditioning methods.

For each test dataset image, the evaluation starts with DDIM sampling an image with the test dataset’s attributes from a random noise. This image is classified with the help of a ResNet-50 classifier introduced in Section 4.4.4 as having the desired new post conditioning attribute or not having it. My post conditioning target label is going to be the opposite of the classifier’s result. Then the same noise from which the classified image was generated from, is sampled with both of the proposed methods described in Section 4.4 and their combined version to evaluate the possibility to increase robustness with the combination of both techniques. Finally, the three post conditioned images are also classified. The performances of the methods are accessed by their success rate.

Chapter 6

Results

This chapter presents and evaluates the results of each experiment, with a detailed discussion of the findings for each one.

6.1 Exp. 1: Generated Image Quality Evaluation

The final FID score at 2400 sampled images from the test dataset is 25.26, the KID score's mean is 0.0098, and the standard deviation is 0.0028. Lisanti et al. [12] reports a FID score of 8.83 and a KID score of 0.0028 (it is not specified whether it is mean or standard deviation, it is assumed to be mean). It is important to note that they used an image resolution of 256×256 instead of my 512×512 resolution and a larger conditioning attribute set with 40 attributes instead of 19. Since my model is not conditioned on attributes like “hat” or different jewelry types the sampled images have a lower likelihood of having those attributes being randomly sampled from noise, thus the features extracted from the Inception V3 network used for KID and FID scores would also not contain those information, resulting in higher values. Besides this, their solution performs approximately three times better, based on this the model can likely be improved on with the adaptation of their proposed training loss, and the further optimizations and architecture scaling requiring more computational resources.

FID and KID scores are also calculated on the test dataset at different sample sizes, shown in the Appendix on Figure A.2.3. The effect of the number of sampled images, improving the score is demonstrated on the FID score. KID scores do not change drastically with the change in image number, thus being a more robust metric when the dataset size is limited.

6.2 Exp. 2: Attribute Correspondence Eval.

My conditional LDM is also evaluated on the correspondence to the conditioning attributes in detail.

Table 6.1 shows the classifier’s measured metrics, the accuracy, the precision, and the recall for all the built-in attributes that the model was trained on. I report results on each attribute individually and I also show the averaged metrics over all of them.

The classifier model introduced in Section 4.4.4 has an average accuracy of 92.37% and its worst performing attribute is the pointy nose with an accuracy of 69.5%. Upon further examination, of the training dataset and its labels, I have found that identifying pointy noses is also difficult for me. I have included Figure A.2.4 in the Appendix to demonstrate the difficulty in correctly labeling this attribute.

In most cases, the generated dataset shows similar or better results compared to the real one. This can happen for several reasons. On one hand, there is the common issue of human mistakes in the labeling process, so while an image might be labeled as having a certain attribute it is not true in reality and a well-trained classifier labels them correctly while resulting in an incorrect prediction. Even if I assume that the dataset does not contain mistakes, there might be some outlier data between the real data, that the classifier might not have seen similar to, during its training, resulting in an incorrect prediction. However, a generative model might not be that likely to produce such outlier data, since it is trained to capture the generic appearance of a certain attribute and not necessarily the edge cases of those attributes. The model-generated images might better fit the general appearance of an attribute set that it was conditioned on.

The average accuracy and precision on both datasets are within a 1% difference, while there is a 3.36% difference in the recall metric in favor of the generated dataset. From these measurements, it is observable that the generated dataset fits its attribute description similarly to the real dataset.

Lisanti et al. [12] reports only their average attribute conditioning accuracy and does not provide individual attribute accuracies, nor do they include precision and recall metrics. They used a ResNet-18 model instead of a ResNet-50 model, used in this work. They measured an average accuracy of 90.85% on their validation set and 90.53% on the generated samples. Although their model relied on the cross attention mechanism, it produced similar results to my computationally cheaper time embedding concatenation-based solution, conditioning fidelity-wise.

Table 6.1: Comparison of accuracy, precision, and recall for classifier attributes on the real and generated datasets. Results on the real dataset aim to demonstrate the accuracy of the classifier, while results on the generated dataset aim to measure the attribute-following capability of the LDM model.

Attr./Metric	Real Dataset			Generated Dataset		
	Acc.	Prec.	Recall	Acc.	Prec.	Recall
Mean	92.37%	81.34%	70.89%	92.46%	80.34%	74.25%
Male	98.62%	98.03%	97.64%	99.62%	99.22%	99.61%
Young	90.96%	91.28%	97.95%	90.25%	89.72%	99.05%
Bald	99.42%	87.50%	84.00%	99.38%	83.02%	88.00%
Bangs	95.17%	88.70%	84.25%	97.79%	94.66%	93.15%
Receding H.	93.87%	64.20%	53.89%	94.13%	66.46%	54.40%
Black Hair	92.33%	79.14%	86.61%	91.96%	85.39%	74.80%
Blond Hair	94.42%	86.75%	82.00%	95.42%	86.00%	89.52%
Brown Hair	88.12%	82.52%	62.77%	82.17%	59.32%	76.77%
Gray Hair	99.08%	87.18%	85.00%	99.12%	90.41%	82.50%
Straight Hair	85.88%	70.00%	57.06%	85.71%	69.23%	57.26%
Wavy Hair	85.12%	81.19%	79.14%	84.92%	77.04%	85.84%
No Beard	97.50%	98.13%	98.91%	97.71%	97.48%	99.85%
Goatee	97.33%	77.78%	77.78%	97.33%	87.04%	65.28%
Mustache	97.04%	85.19%	53.91%	96.17%	97.37%	28.91%
Sideburns	97.92%	88.11%	85.34%	98.33%	90.37%	88.48%
Narrow Eyes	89.38%	56.79%	47.92%	91.79%	80.54%	41.67%
Oval Face	86.71%	80.72%	18.11%	85.46%	52.48%	60.00%
Pale Skin	96.75%	85.19%	51.11%	95.42%	58.62%	62.96%
Pointy Nose	69.50%	57.12%	43.48%	74.08%	62.00%	62.61%

6.3 Exp. 3: Built-In Condition Modifications

Figure 6.1 shows images sampled from the same initial noise with different conditional attributes, randomly taken from the test dataset’s labels. It is visible that in each row images have a strong resemblance between each other, not only in facial structure and features but also in the background. Besides the attribute labels, the initial generated noise also serves as a way to condition the model.

Table 6.2 reports the similarity values averaged over all groups with a group size of 5 and the standard deviations calculated from the similarity metrics. The mean LPIPS value over the groups, which were sampled from the same noise is half of the groups sampled from different noises, which means a much stronger similarity between them. Although they are sampled from the same noise, the conditioning labels are different, so it is evident that the generated images are going to have dissimilarities



Figure 6.1: In each row images are sampled from the same initial noise with different conditioning labels.

between each other. This big improvement in the LPIPS value indicates that images in the same groups resemble each other in a stronger way than when they are each sampled from different initial noises.

The trained on condition modification with DDIM sampling from the same initial noise serves as a promising way for the iterative editing of the generated images. In practical applications for finding the desired image, several random noises can be sampled with the same condition in a batch and the user can choose by preference. Further samples are demonstrated in the Appendix on Figure A.2.5.

Table 6.2: Comparison of the LPIPS values’ means and standard deviations in scenarios where images in groups are sampled from different noises and where the same noise was used for the generations.

Metrics	Different Noise	Same Noise
LPIPS	0.6052 ± 0.0732	0.3161 ± 0.0684

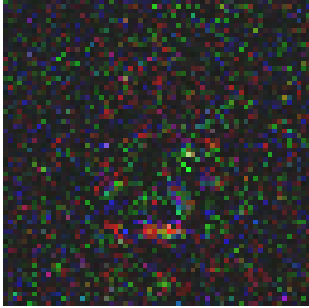
6.4 Exp. 4: Post Conditioning Techniques Eval.

In this section one post conditioning attribute is evaluated by all of the proposed methods.

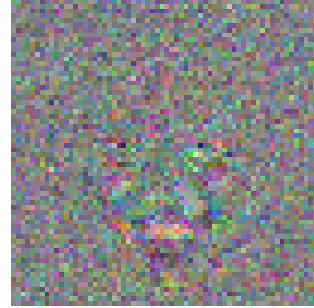
6.4.1 Conditioning on Smiling

For the guidance method, the latent classifier model was trained on 1,800,000 latent images corresponding to 900 real training images where the person is not smiling and 900 where the person is smiling. The classifier model had an 81.20% accuracy on a validation set consisting of 200,000 latent images for 200 real images. For the start latent editing method, 900-900 start latents were used for the extraction of the “smiling” feature axis. For the evaluation, 600 test conditions were sampled 4 times from the same noise. I had to limit the number of images used because sampling the whole test dataset would have meant sampling 9600 images.

In the start latent editing method, the matrix corresponding to the smile feature can be plotted as an RGB image (see Figure 6.2) although it requires scaling for better visualization. Upon closer inspection, higher pixel intensities are observable around the area where the mouth is usually located, which suggests that the latent space and the sampled image have spatial dependency between each other. Increasing the pixel values in the noise around that area could result in the generation of an image with a smile, the same could be true in reverse.



(a) Smile matrix positive values.



(b) Smile matrix normalized values.

Figure 6.2: Comparison of pixel matrices for the smile feature in (a) positive (only positive values are visualized) and (b) normalized forms (normalized over all values).

For the start latent editing, I have used 15 as the weight for the vector. In the classifier guidance method 0.05 was used as the gradient weight with a 0.9 momentum multiplier. The gradients were clipped at -0.05 and 0.05 . The warm-up phase consisted of 300 steps, while the cool-down phase took 200 steps. For the combined method the same parameters were used. I have only experimented on the validation dataset for finding optimal hyperparameters. For the evaluation of the methods, the test dataset was not sampled previously for post conditioning.

Table 6.3 shows the metrics calculated on the sampled images. In all metrics, the combined method has the highest success rate, 56.30% on all targets, while the

guidance method had the poorest performance with 30.85% on all target values. The combined method was especially successful with the removal of smiles on images with a success rate of 66.58%. No ruining effect was observed with either method. A few examples are demonstrated on Figure 6.3, and additional samples are included in the Appendix on Figure A.2.6.

Table 6.3: Comparison of Start Latent Editing, Classifier Guidance, and Combined methods across different metrics for the smile post conditioning. Best values are highlighted.

Metric	Latent Editing	Guidance	Combined
Success when target is 0 (smile removal)	46.36%	40.97%	66.58%
Success when target is 1 (smile addition)	32.03%	22.73%	48.05%
Average success rate	38.42%	30.85%	56.30%

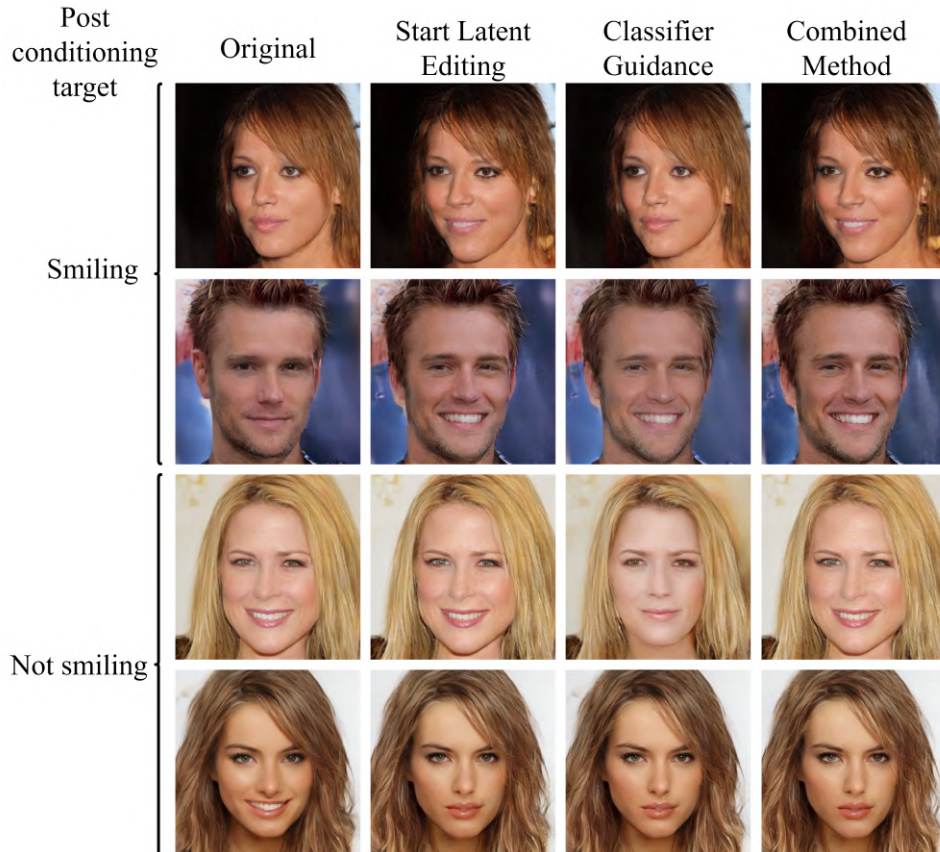


Figure 6.3: In each row images are sampled from the same initial noise with the same conditioning labels, and methods are targeted to add/remove the given attribute from the image.

Chapter 7

Discussion

7.1 Comparison of the Proposed Methods

From all the methods examined in this work, the model’s built-in conditioning drastically outperformed the post conditioning methods’ abilities. As shown in Section 6.2, the images generated from the conditioned LDM showed similar compliance with the conditioning labels as the real test dataset. Images are editable by changing their attribute labels, and with the help of DDIM sampling, they are deterministically sampled, maintaining a strong resemblance between the modified images. However, adding a new conditioning attribute to the model is tedious, as the entire dataset must be labeled for the new label alongside the existing labels, and DDPM training can be computationally expensive, sometimes taking days.

Regarding the post conditioning methods, the start latent editing requires the labels for all of the previous attributes, but it is computationally cheaper since, for this solution, real samples only have to be DDIM inverted, and a simple linear regression is the model behind it.

On the other hand, classifier guidance only needs the labels for built-in features plus the new feature, but it requires more computation due to the DDIM sampling after the DDIM inversion and the training of the latent classifier model. The guidance method often underperformed compared to the other two methods; however, the most reliable solution was the combination of both methods. The higher number of hyperparameters in the guidance method might make it difficult to find a reliable setting with a high success rate on the sampled images.

Their combined method seems to outperform both methods in their post conditioning success rate. Since it uses both solutions, the stronger dataset constraint is present for this method, as it also requires the new data to be labeled on all of the

previous conditioning labels. However, the labeling of the data might be worthwhile since it is by far the most successful method.

7.2 Comparison with Popular Text-Conditioned Models

In recent years, text-conditioned models have become popular for their ability to generate images from textual descriptions. Although these models can generate almost anything that can be described in words and have an image representation, they often lack precision and the ability to control or refine generated images.

I took two text-conditioned models, one open source and one closed source which is available as a service, and tried to generate facial images based on attribute descriptions similar to the labels found in CelebA-HQ. As the open source model, I have used Stable Diffusion V1.5 and as a closed source model, I experimented with Dall-E 3 available from OpenAI’s chat service.

The first problem comes from the embedding of binary attributes into appropriate prompts for the models. I have chosen to create a prompt, wherein the first sentence I describe the style of the image I would like, i.e. “A photorealistic facial image.”, and in the second sentence, I describe the attribute labels with words. For simplicity, I have not included all of the 19 attributes on which my LDM was trained.

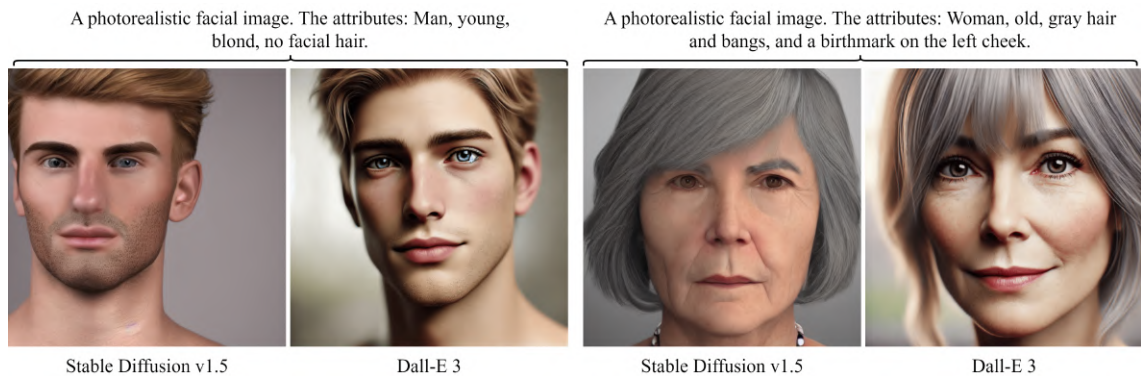


Figure 7.1: Text-conditioned samples from Stable Diffusion v1.5 and Dall-E 3.

The models mostly struggle with generating photorealistic images, as Figure 7.1 shows; they are usually cartoon-like, which can be limiting in some specific tasks. Furthermore, the models completely miss the birthmark specified in one of the prompts, lacking the ability to fine-tune the images. Additionally, because Dall-E is only available as a service, it makes it impossible to iteratively edit images. If asked

to edit the result, the resulting sample was a completely different image, not even resembling the previous one.

I believe that for domain-specific tasks, such as facial image generation for law enforcement, a solution similar to mine is more fitting, as it offers better control over conditioning and image editing, without requiring prompt engineering from the user’s side. Furthermore, there is no style-specific problem, as the model is trained solely on realistic images, so the generated data aims to resemble that style.

7.3 Possible Applications

In image generative tasks where high quality and precision are required, I would suggest training the LDM model with the set of required attributes, as this solution demonstrated the strongest conditioning capabilities. In a practical setting, such as character face or facial composite generation and editing, multiple images can be sampled from different noise inputs to acquire the most fitting description of the required face for attributes that the model is not conditioned on.

In cases where there is labeled data with the unconditioned attribute, especially if that data is also labeled with all of the conditioned attributes, it is possible to expand the set of conditioning attributes with one of the proposed post conditioning methods, although further optimizations are required for them. Additionally, using pre-trained, unconditional LDMs could be a promising application of the post conditioning methods; the primary limitation is finding a pre-trained model suitable for the domain-specific task. [†]

7.4 Future Work

Future work includes improving the image quality generated by the LDM model, both in terms of FID and KID scores, as well as increasing the resolution. Ideally, the model would be capable of generating images at a 1024×1024 resolution. Furthermore, there is room for improvement in the post conditioning techniques, the creation of a more robust solution is necessary. Some potential capabilities remain unexplored, particularly regarding the image editing functionality on real images; these samples could be DDIM-inverted and edited during the resampling process. Additionally, evaluating the performance of post-conditioning methods on pre-trained unconditional models could provide a clearer picture of their capabilities.[†]

Chapter 8

Summary

In this work, I have demonstrated the attribute conditioning abilities of a latent diffusion model capable of generating high-quality facial images at a 512×512 resolution. I have used the CelebA-HQ dataset to train and evaluate my proposed methods with 19 of its conditioning attributes.

First, KID and FID scores were calculated on the test samples generated from the trained model. Although I could not find comparable solutions with the same image resolution and conditioning attribute set, I report a FID score of 25.26, a KID score mean of 0.0098, and a standard deviation of 0.0028 over 2400 samples.

The images generated from the test partition’s conditions did not show any significant difference compared to the real test images when the label correspondence was evaluated. The real data had an average accuracy of 92.37% over all of the attributes averaged, while the generated images had performed 0.09% percent better, which indicates that the LDM has managed to follow the attributes well.

Furthermore, I have evaluated the methods’ ability to modify images with DDIM sampling from the same initial noise. I have demonstrated that images sampled from the same noise tend to keep certain features, structures, and backgrounds when sampled with different conditioning labels, thus creating an efficient way for iterative image modification.

Finally, I adapted and combined the two proposed post-conditioning methods to guide the model toward a target attribute during the sampling process, even for attributes on which the model was not originally trained. I have tested them using the “smiling” condition, targeting the removal/addition of the smile depending on its presence in the image sampled from the initial noise.

First, the method of attribute feature axis retrieval and start latent modification, inspired by TL-GAN, was tested which showed a 38% success rate. The second method, using classifier guidance on the intermediate latents during the sampling process, had a worse performance with a 30.38% success rate and an added computational overhead compared to the first one. However, their compared solution substantially outperformed both of them with a success rate of 56.30%.

These results have shown that post conditioning is possible on conditional latent diffusion models, not just on regular DDPMs sampling through the image space. Further refinement of these techniques could eliminate the need for retraining the base model when adding new conditions, potentially saving days of computational time in some cases.

This work has demonstrated different ways for attribute conditioning in latent diffusion models. Conditioning during training showed high precision with respect to the specified attributes and offers a promising approach for iterative image modification while maintaining resemblance across iterations. Additionally, the experiments with post conditioning an already conditioned model through my proposed methods showed that it is possible to provide additional conditioning to the latent, compressed representations of the high-resolution images in conditional LDMs.

Acknowledgements

I would like to thank my advisors, András Kalapos and Dr. Bálint Gyires-Tóth, for their help and guidance throughout my work.

Additionally, I acknowledge KIFÜ (Governmental Agency for IT Development, Hungary, <https://ror.org/01s0v4q65>) for awarding us access to the Komondor HPC facility based in Hungary.

Bibliography

- [1] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.
- [2] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.
- [3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.
- [4] Shaobo Guan. Generating custom photo-realistic faces using ai, 10 2018. URL <https://blog.insightdatascience.com/generating-custom-photo-realistic-faces-using-ai-d170b1b59255>. Medium.
- [5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [6] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.
- [7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [8] Jijin Hu, Ke Li, Yonggang Qi, and Yi-Zhe Song. Scale-adaptive diffusion model for complex sketch synthesis. In *The Twelfth International Conference on Learning Representations*.

- [9] Morris A Jette and Tim Wickberg. Architecture of the slurm workload manager. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 3–23. Springer, 2023.
- [10] Tero Karras. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.
- [11] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [12] Giuseppe Lisanti and Nico Giambi. Conditioning diffusion models via attributes and semantic masks for face generation. *Computer Vision and Image Understanding*, 244:104026, 2024.
- [13] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.
- [14] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1096–1104, 2016.
- [15] NVIDIA Developer Blog. Improving diffusion models as an alternative to gans - part 1, 2022. URL <https://developer.nvidia.com/blog/improving-diffusion-models-as-an-alternative-to-gans-part-1/>.
- [16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- [17] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10619–10629, 2022.
- [18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

- [19] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- [20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.
- [21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.
- [22] Changhao Shi, Haomiao Ni, Kai Li, Shaobo Han, Mingfu Liang, and Martin Renqiang Min. Exploring compositional visual generation with latent classifier guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 853–862, 2023.
- [23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [24] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.
- [25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [26] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- [27] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.
- [28] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

Appendix

A.1 Implementation Details

The following section introduces the software and hardware environment on which the work was executed.

A.1.1 Hardware[†]

For this project, I utilized the Komondor HPC with Nvidia A100 Graphical Processing Units (GPU), the department’s two A100 GPUs, and my own Nvidia RTX 3060 GPU. On Komondor, single-node training can leverage up to 8 GPUs for multi-GPU training. Across all experiments, model training, and optimizations, the project required approximately 2,500 GPU hours up to this point. Due to its limited 12GB VRAM, the 3060 GPU was used only for classifier training and model inference, as the full models do not fit in 12GB during training. Additionally, the 3060’s slower computation speed makes it less efficient, with a single DDPM sampling taking approximately 120 seconds compared to around 30 seconds on the A100 with 40GB VRAM.

A.1.2 Software[†]

The training for this project was conducted within a Docker container on the Komondor HPC, which uses the Slurm workload manager [9] to schedule and manage compute resources. Running in a Docker environment ensured consistency in software dependencies and configurations across all runs, making the setup portable and reproducible.

PyTorch [16] was the primary framework for model development, utilizing its Distributed Data Parallel (DDP) module to efficiently handle multi-GPU training by distributing data and computations across the GPUs on a single node. This setup al-

lowed for effective scaling when running on Komondor’s 8 GPU nodes, significantly accelerating training times. However, increasing the number of GPUs introduces synchronization overhead, meaning that doubling the number of GPUs does not result in a direct doubling of training speed. The DDP module balances workload distribution and synchronization, but communication between GPUs still adds some delay as the number of GPUs grows.

For experiment tracking and logging, Weights and Biases (W&B) was used. W&B facilitated real-time monitoring of training metrics, hyperparameter configurations, and model checkpoints, providing a comprehensive history of each experiment. This integration streamlined model evaluation and comparison, making it easier to analyze performance across various runs and parameter settings.

A.1.3 VQ-VAE Training Details

The autoencoder was trained to create 3-channel 64×64 latent representations from 512×512 images. This setup required 3 downsampling blocks, with each down-, mid-, and up-block containing 2 layers. The model was trained for 80 epochs with a learning rate of 0.00001 using the Adam optimizer and an 8192 codebook size. Training was conducted on 6 Nvidia A100 GPUs with a batch size of 2 per GPU. Figure A.2.2 aims to demonstrate the capability of the trained model.

A.1.4 LDM Training Details

The LDM model also consists of 3 downsampling blocks, each applying attention. I used 512 as the time embedding dimension, with each block containing 2 layers. The model uses a linear sampling scheduler and was trained for 700 epochs at a learning rate of 0.000001 with the Adam optimizer. Training was conducted on 8 Nvidia A100 GPUs with a batch size of 2 per GPU.

A.2 Additional Figures

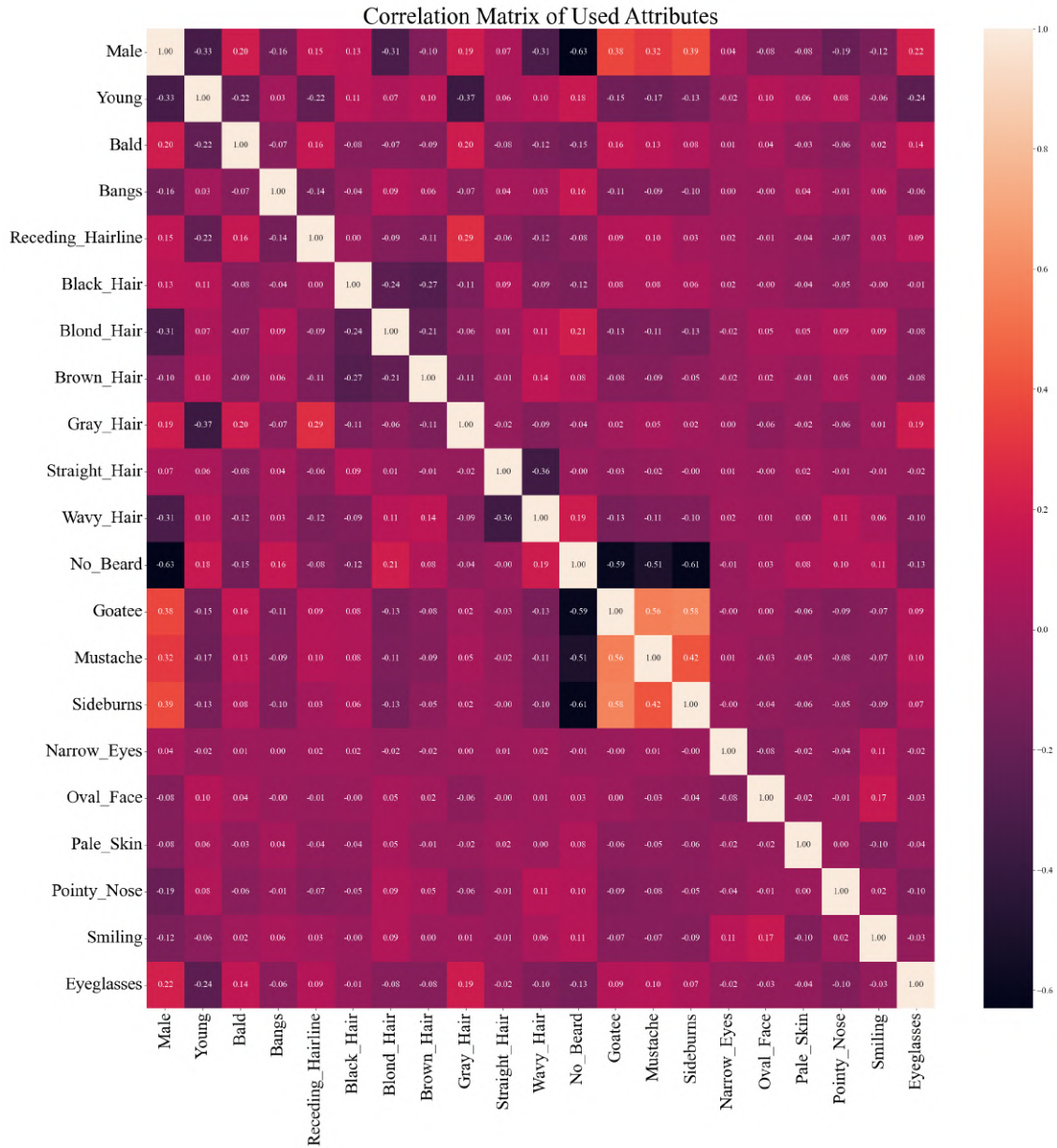


Figure A.2.1: Correlation matrix of the used attributes on the dataset.

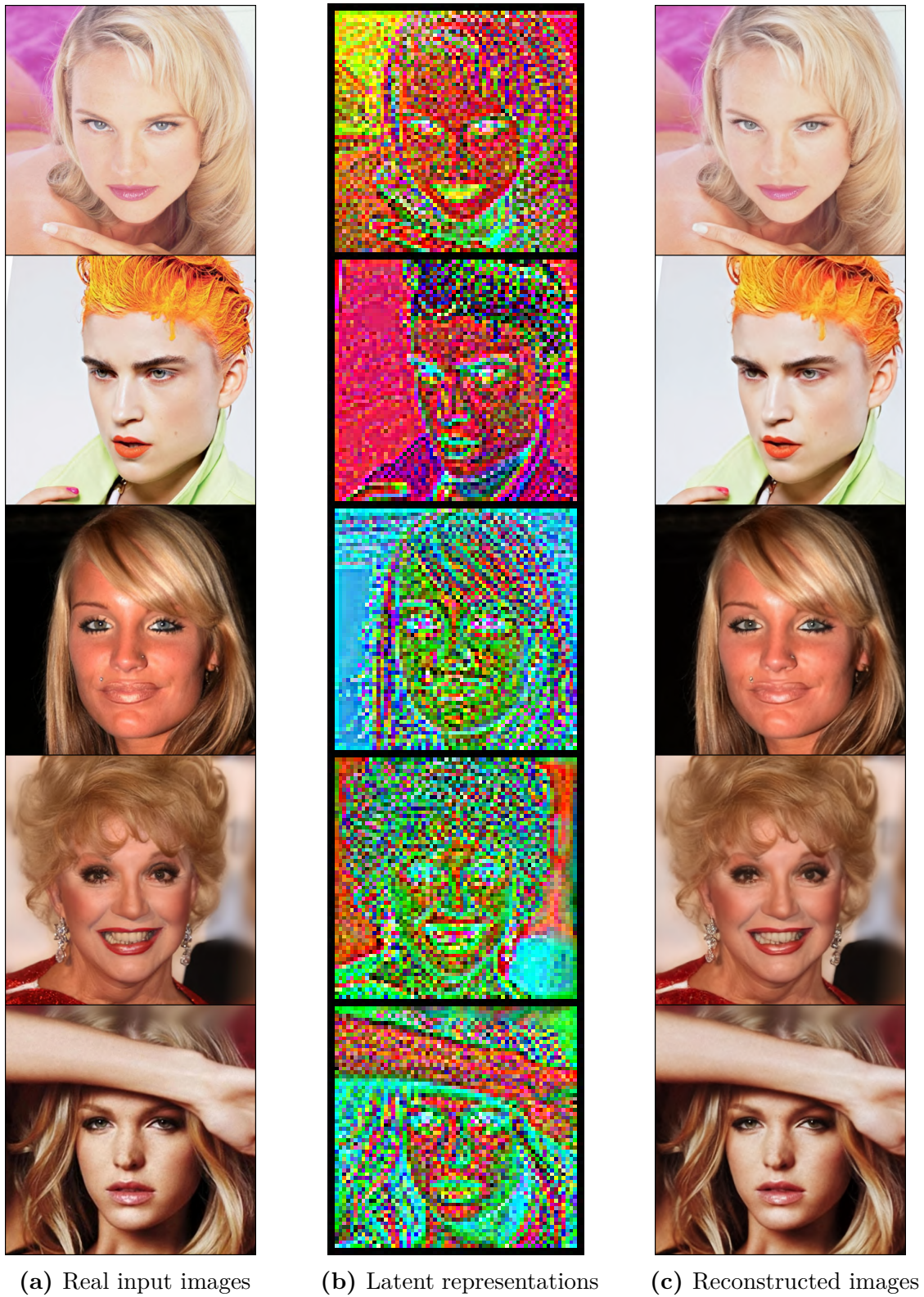
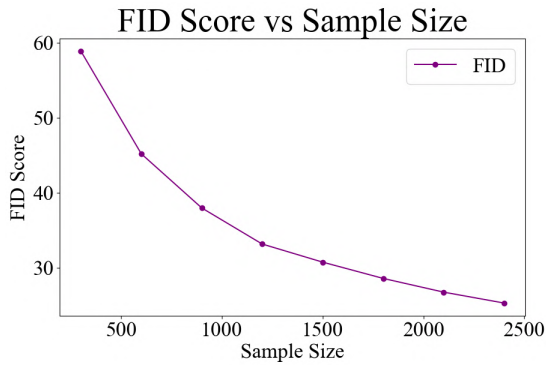
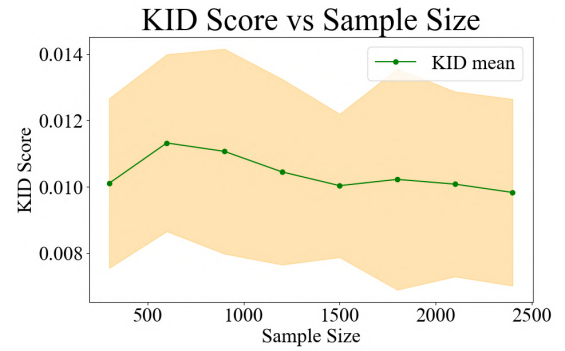


Figure A.2.2: Images demonstrating the capability of the VQ-VAE, first latent representations are created from from real images, than the images are reconstructed from the latent representations.



(a) FID scores calculated on different sample sizes. With the growth of sample size, a reduction in the FID score is observable.



(b) KID scores calculated on different sample sizes. Scores remain relatively stable as the sample size grows, showing stability with smaller sample sizes.

Figure A.2.3: Comparison of FID and KID scores calculated on varying sample sizes.



Figure A.2.4: Images demonstrating the subjectivity of the pointy nose label. Both images are labeled as having a pointy nose.

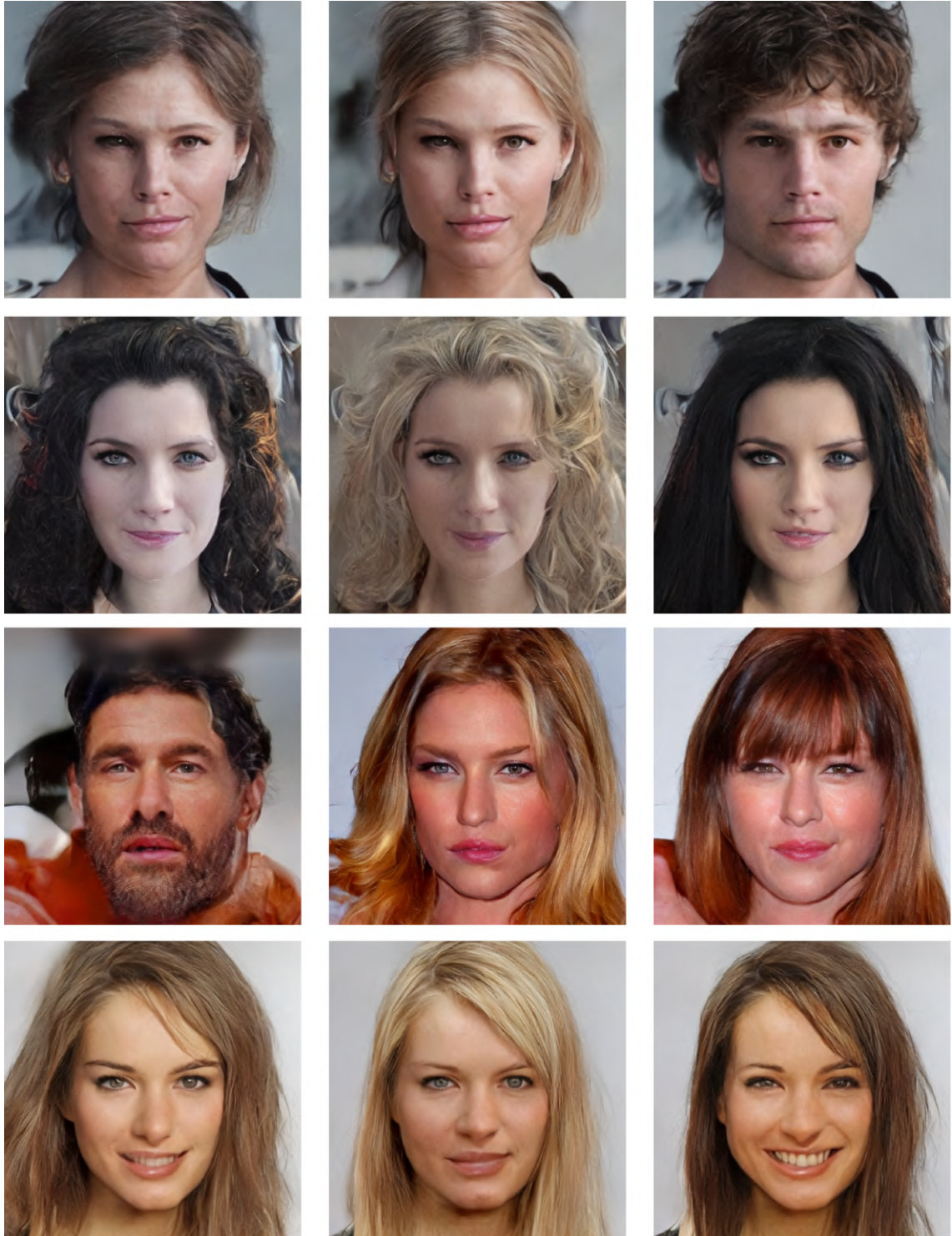


Figure A.2.5: In each row images are sampled from the same initial noise with different conditioning labels.

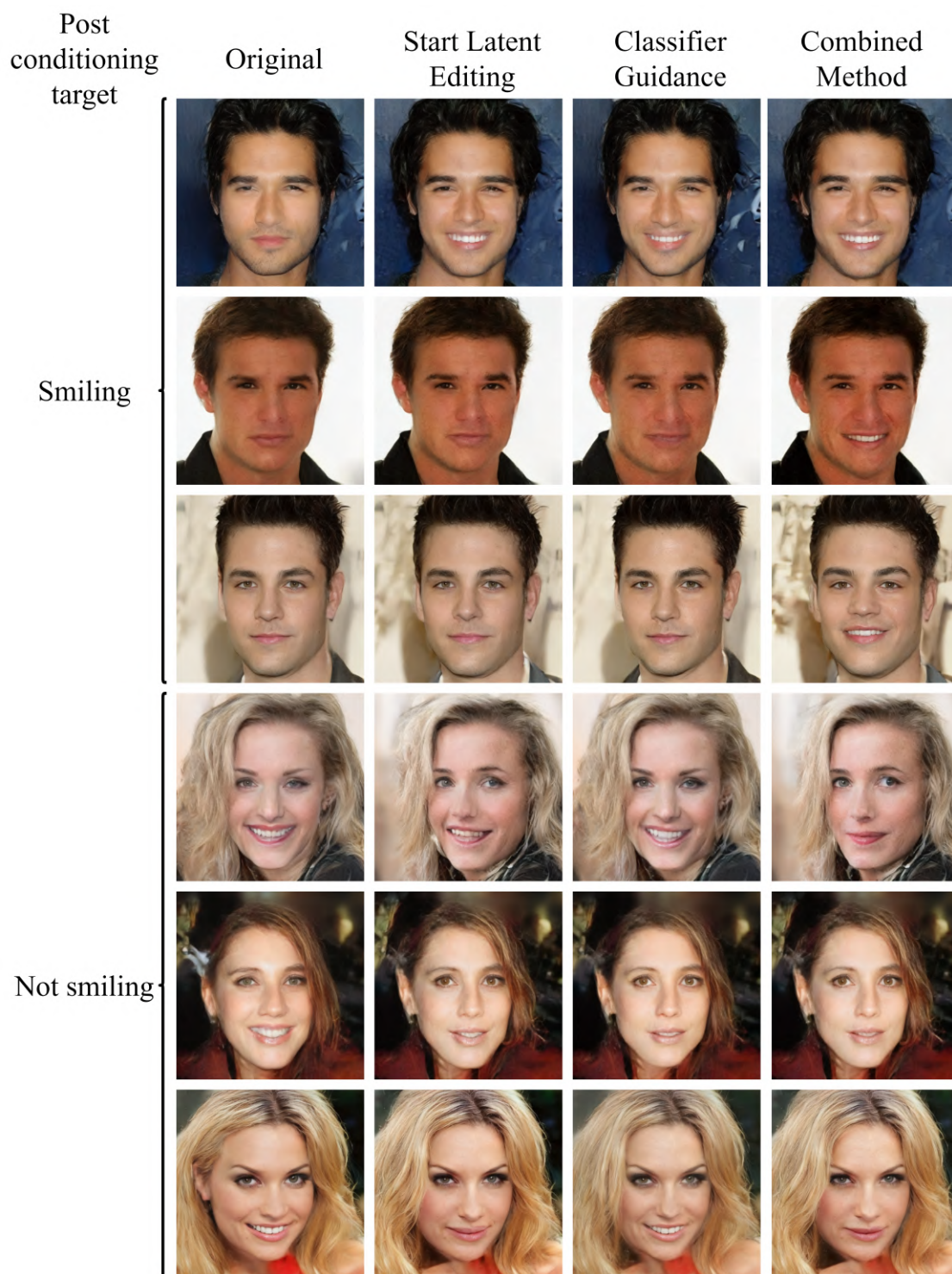


Figure A.2.6: In each row images are sampled from the same initial noise with the same conditioning labels, methods are targeted to add/remove the given attribute from the image.