AUTOMATED IMAGE GENERATION

(CCDS24-0163)

Hill Seah Wen Qi

College of Computing and
Data Science 2024

NANYANG TECHNOLOGICAL UNIVERSITY

AUTOMATED IMAGE GENERATION
(CCDS24-0163)

Submitted in Partial Fulfillment of the
Requirements for the Degree of Bachelor of
Computing (Hons) in Data Science and
Artificial Intelligence of the Nanyang
Technological University

by

Hill Seah Wen Qi

College of Computing and
Data Science 2024

22nd March 2025

Project Supervisor: Associate Professor Lu Shijian
Examiner: Associate Professor Lin Guosheng
Academic Year: 2024/2025

Abstract

Multi-conditional image generation aims to synthesize images that satisfy diverse conditions, such as textual descriptions, segmentation masks, and landmark constraints. Current training-free approaches, which rely on off-the-shelf and open-source pre-trained networks to provide guidance, perform well for single conditions but fail to capture the complex interdependencies among multiple conditions. The purpose of this project is to develop a novel framework that overcomes these limitations by effectively modelling the interactions between conditions. To achieve this, the project analyzes existing methods and introduces an innovative design: a time-independent approximated energy guidance function enhanced with interaction modelling. This function captures non-linear and complex dependencies, guiding an iterative denoising process to progressively refine the generated images. Experimental results indicate that our approach outperforms existing techniques, producing images that are both coherent and condition-consistent. In conclusion, the framework not only resolves key challenges in multi-conditional image synthesis but also provides a basis for future research, with recommendations to further explore adaptive energy functions for even broader applicability.

Acknowledgements

This project is standing on the shoulders of giants and would not have been possible without the foundational knowledge, resources, and inspiration provided by the authors and researchers whose works have been referenced in this project. Their pioneering contributions have laid the groundwork for advancements like mine. Hence, I would like to extend my deepest gratitude to them for their help in the completion of this project.

Additionally, I am extremely grateful to my project supervisor Professor Lu Shijian, and my student mentor Xu Muyu for their guidance and assistance in this endeavor, making this journey a memorable and fulfilling experience.

Table of Contents

Table of Tables

Table of Figures

1. Introduction

1.1 Problem Statement

Multi-conditional image generation has emerged as a critical area in computer vision, where the goal is to generate images that simultaneously satisfy multiple user-defined constraints.

Training-free approaches are rapidly gaining popularity due to advantages over training-required methods, as they eliminate the need for extensive datasets and computationally expensive training processes, making them faster to deploy, more cost-effective, and easier to adapt to new tasks or domains.

For single-condition image generation, training-free approaches often leverage off-the-shelf and open-source pre-trained networks to estimate the distance between an intermediate image and the condition, guiding the iterative denoising processes [3]. However, extending these techniques to multi-condition image generation presents challenges. Unlike single-condition tasks, where adhering to a single constraint is sufficient, multi-conditional generation requires optimizing multiple constraints simultaneously. This often leads to conflicts, trade-offs, or failures (e.g., aligning text descriptions while maintaining geometric consistency). Without explicit training, these methods struggle to balance constraints, resolve conflicts, and adapt to complex, real-world multimodal scenarios.

Specifically, training-free methods often struggle with effectively handling the complexity and interdependence of multiple conditions. A core issue lies in the independent optimization of conditions, where most approaches treat each guidance signal separately under the assumption of independence—an oversimplification that neglects the semantic interactions between conditions, resulting in incoherent or conflicting outputs [6]. Additionally, without a learned fusion mechanism, many training-free methods face conflicting guidance signals when different conditions pull the generation in different directions, degrading visual fidelity [8]. This limitation is further compounded by the lack of joint condition satisfaction mechanisms, where current approaches struggle to enforce coherence when conditions interact in non-trivial ways, particularly in scenarios where satisfying one constraint may impact the fulfillment of another [21].

Hence, this necessitates the development of new training-free strategies to manage multi-condition interdependencies, dynamically resolve conflicts, and adjust constraints among conditions to ensure robust, high-quality image outputs that satisfy all specified requirements.

1.2 Objective

This project aims to overcome the limitations of training-free multi-conditional image generation by integrating an approximated time-independent energy guidance function with effective interaction modelling within the diffusion model denoising process. This approach enables the dynamic handling of multiple, potentially conflicting conditions, ensuring coherent, high-quality image synthesis that satisfies all specified constraints. Moreover, it adheres to a fully training-free paradigm, allowing new conditions to be incorporated without requiring retraining. Ultimately, the goal is to enhance adaptability and stability in complex, real-world multimodal scenarios without requiring retraining.

We will evaluate our different proposed models to identify the best-performing approach based on qualitative results, such as visual fidelity and consistency in constraint adherence, as well as quantitative metrics like Fréchet Inception Distance (FID) and constraint-specific distances. This will allow us to comprehensively assess the balance between image quality and the successful integration of multiple conditions.

1.3 Scope

Given GPU memory constraints, running larger diffusion models with general image generation capabilities (such as Stable Diffusion or ControlNet) was infeasible. These models require substantial Video Random-Access Memory (VRAM) to store high-dimensional tensors during the iterative denoising process, leading to out-of-memory errors on our available hardware. Consequently, we opted for a smaller pre-trained unconditional human face diffusion model for facial image generation.

The conditioning inputs for our multi-conditional image generation experiments were selected to provide a diverse and complementary set of information as shown in Table 1 below:

Condition	**Description**
Face ID	Encodes identity-specific facial features in a numerical representation, ensuring consistency in facial appearance
Sketch	Provides a structural outline of the subject, capturing overall shape and contours.
Landmark	Defines spatial key points (e.g., eyes, nose, mouth) to enforce geometric accuracy in facial features.

Segmentation Map	Specifies region-based attributes, guiding the model in differentiating facial parts and background elements.
Text	Offers high-level semantic descriptions, providing flexible and interpretable guidance for image generation.

Table 1. Scope of supported conditions

However, the results and contributions of this project remain translatable to general image generation, as the methods and frameworks developed are model-agnostic and condition-agnostic, and can thus be applied to larger models. Additionally, the principles of interaction-aware image generation explored in this project are not limited to facial images but can be generalized to broader contexts, enabling scalability to more complex image generation tasks.

2. Related Work

2.1 Conditional Score Based Diffusion Models

For unconditional score-based diffusion models (SBDM) operating on score theory, its goal is to learn and estimate a time-dependent score function $\nabla_{x_t} \log p(x_t)$ that guides the denoising phase of a noisy image x_t to x_{t-1} at time step t during the iterative denoising process. The denoising formula is denoted as follows:

$$x_{t-1} = \left(1 + \frac{1}{2}\beta_t\right) x_t + \beta_t \nabla_{x_t} \log p(x_t) + \sqrt{\beta_t}\epsilon \tag{1}$$

where β_t is a hyperparameter and $\epsilon \sim N(0,1)$ represents random Gaussian noise [16].

For conditional diffusion, a corrective gradient $\nabla_{x_t} \log p(c|x_t)$ is added to the denoising formula Equation (1) to guide x_t to a hyperplane in the data space that aligns with the condition c [16], resulting in the following formula:

$$x_{t-1} = \left(1 + \frac{1}{2}\beta_t\right) x_t + \beta_t \nabla_{x_t} \log p(x_t) + \nabla_{x_t} \log p(c|x_t) + \sqrt{\beta_t}\epsilon \tag{2}$$

Training-required methods often retain the time-dependent nature of the corrective gradient $\nabla_{x_t} \log p(c|x_t)$, learning it through approaches like classifier training. In contrast, training-free methods aim to approximate the corrective gradient using time-independent functions.

2.2 Energy Diffusion Guidance

One alternative method to model the corrective gradient $\nabla_{x_t} \log p(c|x_t)$, would be to use an energy function as follows:

$$p(c|x_t) = \frac{e^{-\lambda\varepsilon(x_t,c)}}{z} \tag{3}$$

where $z = \int_{c\in C} e^{-\lambda\varepsilon(x_t,c)}$ represents a normalizing constant, λ represents the positive temperature constant and $\varepsilon(x_t, c)$ represents an energy function measuring the similarity between a given condition c and a noisy image x_t [9].

The energy function value decreases as the similarity between c and x_t increases, reaching the value zero when c and x_t are perfectly similar. Thus, the corrective gradient can be remodelled to energy guidance as $\nabla_{x_t} \log p(c|x_t) \propto -\nabla_{x_t}\varepsilon(x_t, c)$ [9].

The final denoising formula that incorporates energy guidance into the denoising formula Eq. (2) is as follows:

$$x_{t-1} = \left(1 + \frac{1}{2}\beta_t\right)x_t + \beta_t\nabla_{x_t}\log p(x_t) + \sqrt{\beta_t}\epsilon - p_t\nabla_{x_t}\varepsilon(x_t, c) \tag{4}$$

where p_t represents the learning rate of the energy guidance term.

2.3 Approximating Time-Dependent Energy Guidance with Time-Independent Distance Functions

To obtain the energy guidance function, most training-required methods revolve around training classifiers to calculate a time-dependent distance measuring function $D_\phi(c, x_t, t)$ to approximate it, where ϕ represents the trained parameters of the classifier [5]. This is particularly problematic as it is extremely difficult to find an existing pre-trained model for the noisy image x_t to ensure training-free. To circumvent this, we can estimate the time-dependent energy guidance with time-independent distance functions through a series of approximations.

Unlike time-dependent networks, time-independent functions for measuring distances in clean data, x_0, are widely accessible [20]. Open-source pre-trained models, such as those for classification, text encoding, segmentation, and face identification, are commonly available and highly effective for working with clean images.

First, we can approximate the time-dependent distance function $D_\phi(c, x_t, t)$ with the time-independent distance function $D_\theta(c, x_0)$ where θ represents the pre-trained parameters, as follows:

$$D_\phi(c, x_t, t) \approx E_{P(x_0|x_t)}\left[D_\theta(c, x_0)\right] \tag{5}$$

and this is reasonable because if the noisy image x_t is close to condition c , then the corresponding clean image x_0 should also be close to c [20].

Next, we need to approximate a clean image x_0 corresponding to an intermediate noisy image x_t for each time step t as follows:

$$x_{0|t} \approx E[x_0|x_t] = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t + (1 - \bar{a}_t)s(x_t, t)\right) \tag{6}$$

where $\sqrt{\bar{\alpha}_t} = \prod_{i=1}^{t}(1 - \beta_i)$ and $s()$ is the pre-trained score estimator [3].

Finally, we can combine the results to approximate the time-dependent energy guidance function with a time-independent distance function provided by the condition-specific open-source model. For a singular condition c, we get $\varepsilon(x_t, c) \approx D_\theta\left(c, x_{0|t}\right)$ [20].

3. Proposed Methodology

3.1 Multi-Conditional Training-Free Energy Guidance

3.1.1 Multi-Conditional Energy Guidance Model – No Interaction Modelling

A primitive approach in modelling a multi-conditional energy guidance formula would be to extend the singular-conditional energy guidance formula $\varepsilon(x_t, c) \approx D_\theta\left(c, x_{0|t}\right)$ as a weighted sum of the different distance functions for each of the respective conditions as follows:

$$\varepsilon(x_t|c_1, c_2, \cdots c_n) \approx \sum_{i=1}^{n} \lambda_i D_i\left(c_i, x_{0|t}\right) \tag{7}$$

where $D_i\left(c_i, x_{0|t}\right)$ represents the distance between condition, c_i, and the approximated clean image at time step t, $x_{0|t}$, computed by a pre-trained network i that is specific to condition c_i, and λ_i represents the weighting factor of $D_i\left(c_i, x_{0|t}\right)$.

For example, if c_i represents a text condition, then the pre-trained network i could be a CLIP embedding model, and $D_i\left(c_i, x_{0|t}\right)$ could be a Euclidean distance value between the CLIP embedding of c_i and $x_{0|t}$.

However, the primary limitation of this primitive model lies in its reliance on the naive assumption that all conditions are mutually independent and non-conflicting. Consequently, it struggles to generate high-quality images under diverse conditions, particularly when those conditions exhibit complex, non-linear interdependencies.

3.1.2 Improved Multi-Conditional Energy Guidance Model – Interaction Modelling

To overcome the primary limitation and improve the multi-conditional energy guidance model, we can account for the interactions between all combinations of conditions as follows:

$$\varepsilon(x_t|c_1, c_2, \cdots c_n) \approx \sum_{i=1}^{n} \lambda_i D_i(c_i, x_{0|t}) + \sum_{i \neq j}^{\eta} \lambda_{ij} \phi_{ij}(x_{0|t}, c_i, c_j) \qquad (8)$$

where $\phi_{ij}(x_{0|t}, c_i, c_j)$ represents a function that models the interactions between conditions $c_i\ and\ c_j$, and the approximated clean image $x_{0|t}$ in their respective spaces, and λ_{ij} represents a weighting factor of the interactions between $x_{0|t}, c_i, c_j$.

3.2 Interaction Models

To maintain the training-free nature of image generation in this project, we deliberately avoided interaction modelling methods that require training, such as attention mechanisms, graph-based models, bilinear models, latent factor models, or any other training-dependent neural networks. These methods demand additional training time and data, which would undermine the core advantages of the training-free conditional image generation framework.

3.2.1 Simple Similarity Measures

A straightforward approach to interaction modelling is by simply computing the similarities between the different conditions with a chosen similarity metric. Here, we propose trying 3 different similarity metrics as follows:

1. Euclidean distance,

$$\phi_{ij}(x_{0|t}, c_i, c_j) = \|\mathrm{c_i} - \mathrm{c_j}\| \qquad (9)$$

2. Cosine similarity,

$$\phi_{ij}(x_{0|t}, c_i, c_j) = \frac{c_i \cdot c_j}{\|c_i\| \, \|c_j\|} \qquad (10)$$

3. Pearson correlation,

$$\phi_{ij}(x_{0|t}, c_i, c_j) = \frac{\mathrm{cov}(c_i, c_j)}{\sigma(c_i) \cdot \sigma(c_j)} \qquad (11)$$

In general, these simple similarity measures are quick, interpretable, and computationally efficient interaction modelling in tasks where the relationships between features are simple or linear.

3.2.2 Polynomial Functions

Polynomial functions can typically be used to model interactions by expanding features into a higher-dimensional space. They are particularly useful in capturing for complex and non-linear relationships [15].

Using polynomial functions to model the interaction terms between each possibly dependent condition $c_i$and c_j, we get the model as follows:

$$\phi_{ij}(x_{0|t}, c_i, c_j) = \left(D_i(c_i, x_{0|t}) \cdot D_j(c_j, x_{0|t}) + k\right)^{\rho} \tag{12}$$

where ρ represents the degree of the polynomial and k represents a constant that controls the flexibility of the polynomial.

3.2.3 Sigmoid Functions

Sigmoid functions are another popular method for modelling interactions between inputs, particularly for capturing non-linear relationships [10]. They are frequently used in the activation functions in neural networks and can model complex dependencies between conditions. It is especially useful in situations where the interactions between conditions exhibit saturating behavior.

Using sigmoid functions to model the interaction terms between each possibly dependent condition $c_i$and c_j, we get the model as follows:

$$\phi_{ij}(x_{0|t}, c_i, c_j) = \tanh\left(\alpha \cdot D_i(c_i, x_{0|t}) \cdot D_j(c_j, x_{0|t}) + m\right) \tag{13}$$

where α represents the scaling factor that controls the sensitivity to the input distance and m represents a bias term.

3.2.4 Gaussian Kernels

The Gaussian kernel is a radial basis function (RBF) that is widely used in machine learning to measure similarity between inputs in a smooth and interpretable manner, defined as

$$G(x\ , y) = \ e^{-\frac{\|x-y\|^2}{2\sigma_i^2}} \tag{14}$$

The Gaussian kernel computes a similarity score between two inputs x and y, where σ is a hyperparameter standard deviation term that controls the sensitivity to differences in input. Its smooth exponential decay enables effective modelling of nuanced relationships in high-dimensional feature spaces [13].

Using Gaussian kernels to model the interaction terms between each possibly dependent condition $c_i$and c_j, we get

$$\phi_{ij}\left(x_{0|t}, c_i, c_j\right) = G\left(c_i, x_{0|t}\right) \cdot G\left(c_j, x_{0|t}\right) \cdot G\left(c_i, c_j\right) \tag{15}$$

where $G\left(c_i, x_{0|t}\right)$ and $G\left(c_j, x_{0|t}\right)$ are Gaussian Kernels capturing the individual effects and $G\left(c_i, c_j\right)$ is the Gaussian Covariance Factor, defined in Equation (14), that captures the dependency between conditions c_i and c_j.

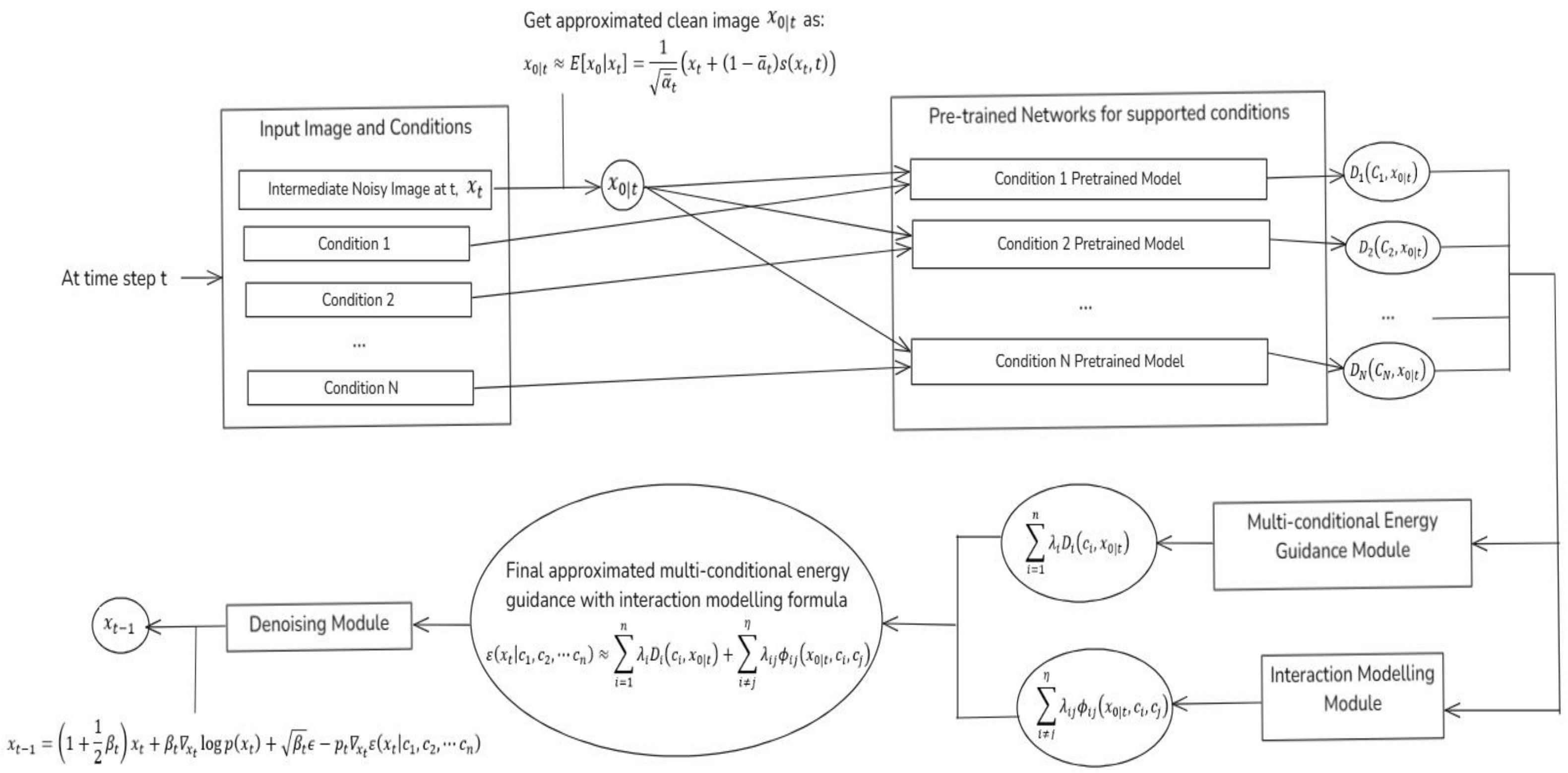
The final multi-conditional energy function model with interaction-aware Gaussian kernels is as follows:

$$\varepsilon(x_t|c_1, c_2, \cdots c_n) \approx \sum_{i=1}^{n} \lambda_i D_i\left(c_i, x_{0|t}\right) + \sum_{i \neq j}^{\eta} \lambda_{ij} G\left(c_i, x_{0|t}\right) \cdot G\left(c_j, x_{0|t}\right) \cdot G\left(c_i, c_j\right) \tag{16}$$

3.3 Configuration

The configuration of our framework enables training-free multi-conditional image generation by guiding the denoising process in diffusion models with time-independent energy functions. Instead of training a new model, we used pre-trained condition-specific networks to extract conditional information and compute distances, which are combined into an approximated energy function. This energy function then dynamically guides the denoising process, ensuring the final image satisfies multiple conditions while maintaining high fidelity and coherence.

This modular approach allows seamless integration of new conditions by simply incorporating an appropriate new pre-trained network, making the framework training-free, highly flexible and scalable for various conditional image generation tasks. Figure 1 below illustrates the full framework.



Get approximated clean image $x_{0|t}$ as:
$x_{0|t} \approx E[x_0|x_t] = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t + (1-\bar{\alpha}_t)s(x_t,t))$
Input Image and Conditions
Intermediate Noisy Image at t, x_t
Condition 1
Condition 2
...
Condition N
At time step t
$x_{0|t}$
Pre-trained Networks for supported conditions
Condition 1 Pretrained Model
Condition 2 Pretrained Model
...
Condition N Pretrained Model
$D_1(C_1, x_{0|t})$
$D_2(C_2, x_{0|t})$
...
$D_N(C_N, x_{0|t})$
Multi-conditional Energy Guidance Module
$\sum_{i=1}^{n} \lambda_i D_i(c_i, x_{0|t})$
Interaction Modelling Module
$\sum_{i \neq j}^{\eta} \lambda_{ij} \phi_{ij}(x_{0|t}, c_i, c_j)$
Final approximated multi-conditional energy guidance with interaction modelling formula
$\varepsilon(x_t|c_1, c_2, \cdots c_n) \approx \sum_{i=1}^{n} \lambda_i D_i(c_i, x_{0|t}) + \sum_{i \neq j}^{\eta} \lambda_{ij} \phi_{ij}(x_{0|t}, c_i, c_j)$
Denoising Module
x_{t-1}
$x_{t-1} = \left(1 + \frac{1}{2}\beta_t\right) x_t + \beta_t \nabla_{x_t} \log p(x_t) + \sqrt{\beta_t}\epsilon - p_t \nabla_{x_t} \varepsilon(x_t|c_1, c_2, \cdots c_n)$

Figure 1. Architecture of Framework for our proposed denoising process

3.3.1 Pre-trained Models

3.3.1.1 Base Unconditional Diffusion Model

Model	**Purpose**
Unconditional Human Face Diffusion Model [11]	Base unconditional face generation model trained on the CelebA-HQ dataset, that generates random faces without conditions. To test the proposed training-free multi-conditional energy guidance models.

Table 2. Pre-trained diffusion models used.

3.3.1.2 Condition Extracting Model

Recall the proposed multi-conditional energy guidance model in Equation (8). To ensure training free, we use pre-trained models specific to the condition to calculate $D_i(c_i, x_{0|t})$.

Hence, for each unique condition type i, a specific model $f_i(x)$ was used to extract the given conditional information in c_i and in the approximated clean image $x_{0|t}$ as $f_i(c_i)$ and $f_i(x_{0|t})$ respectively, and $Distance_i()$ is a pre-determined distance function between $f_i(c_i)$ and

$f_i(x_{0|t})$.

Concretely,

$$D_i(c_i, x_{0|t}) = Distance_i\left(f_i(c_i), f_i(x_{0|t})\right) \tag{17}$$

The specific pre-trained model $f_i(x)$ for each unique condition type i is shown below in Table 3.

Model $f_i(x)$	**Purpose**	$\boldsymbol{Distance_i}()$
Open-source Face Segmentation Network [19]	Supports the Segmentation map condition. Generates a facial Segmentation map of the image and the conditional image.	Euclidean distance
Open-source Landmark Extractor Network [2]	Supports the landmark condition. Generates a facial landmark of the image and the conditional image.	Euclidean distance
Open-source Face Identification Network [4]	Supports the facial ID condition. Generates a Segmentation map of the image and the conditional image.	Euclidean distance
Sketch [18]	Supports the sketch condition. Generates a sketch of the image and the conditional image.	Euclidean distance
CLIP image encoder [12]	Supports textual condition. Encode the image and text condition into the same CLIP feature space.	Euclidean distance

Table 3. Pre-trained condition-specific models used to extract distance information between given condition and image.

3.3.2 Experiments

3.3.2.1 Multi-Condition Combinations

For multi-conditional image generation, we classified conditions into similar groups based on the type of information they provide. Conditions within the same group convey similar features or representations of the image, so testing them together would be contradictory, redundant and meaningless.

Group	**Conditions**	**Description**
Structural Representation	Face ID, Sketch	Defines the specifics of the shape and structure of the facial subject
Spatial Feature	Landmark, Segmentation Map	Describes locations of key points and regions
Semantic Description	Text	Provides high-level conceptual information

Table 4. Conditional groups.

Thus, for testing, we would avoid combining conditions from the same group but evaluate all possible combinations across different groups to ensure diverse and meaningful testing. For example, testing multi-condition combinations like (Face ID, Landmark, Text), (Sketch, Landmark, Text), (Face ID, Segmentation Map, Text), and (Sketch, Segmentation Map, Text).

3.3.2.2 Experimental Setup

For each of the Multi-Conditional Energy Guidance Models proposed in Sections 3.1 and 3.2, we generated a set of 100 images for each multi-condition combination outlined in Section 3.3.2.1. To evaluate the quality and accuracy of the generated images, we measured key metrics under each condition, including condition-specific distances and Fréchet Inception Distance (FID), and took the average values. This setup enables a comprehensive comparison of different conditioning strategies and their impact on generation performance.

3.3.2.3 Hyperparameter Optimization

Due to GPU limitations which resulted in long image generation times**,** hyperparameter tuning was conducted using a grid search over a predefined set of values. This approach ensures a systematic exploration of key parameters while maintaining computational feasibility. The search space for each hyperparameter and the reasonings for them are defined as follows:

Hyperparameter	**Search Space**	**Reasoning**
λ_i	(By ratio, $r_{ij} = \frac{\lambda_i}{\lambda_j}$) 1, 10, 100, 1000	These values span different orders of magnitude to balance guidance strength.
λ_{ij}	(By ratio, $r_{i1j1i2j2} = \frac{\lambda_{i1j1}}{\lambda_{i2j2}}$) 1, 10, 100, 1000	Keeping a similar range ensures meaningful weight adjustments for interaction terms.
ρ	1, 2, 3, 4, 5	The polynomial degree should remain low

		to prevent excessive complexity and overfitting.
k	0.5, 1, 5, 10, 15	This controls polynomial-modeled interaction terms; a range from small to moderate values helps test expressiveness.
α	0.05, 0.1, 0.5, 1, 2	The scaling factor for sigmoid-modeled interactions should remain small to maintain stability in optimization.
m	0.5, 1, 5, 10, 15	Similar to k, this adjusts the strength of sigmoid-based interactions.
σ	0.5, 0.8, 1	A reasonable range for standard deviations in Gaussian kernels, balancing smoothness vs. sharpness in interaction modelling.

Table 5. Search space and reasonings for each hyperparameter

4. <u>Results</u>

4.1 Hyperparameters

After evaluating visual and quantitative results through grid search, the hyperparameter configurations in Table 6 below were found to produce the most coherent and acceptable images within our resource constraints. While this configuration may not be globally optimal, it serves as a reasonable choice for our experiments. The subsequent results in Sections 4.2 and 4.3 are based on these hyperparameters.

Hyperparameter	**Value**	**Description**
λ_i	$\lambda_{Text} : \lambda_{Parse} : \lambda_{Landmark} : \lambda_{ID} : \lambda_{Sketch}$ = 1000 : 1 : 1000 : 1000 : 10	Weighting factor for $D_i(c_i, x_{0\|t})$
λ_{ij}	$\forall i, j, \lambda_{ij} = 1$	Weighting factor for $\phi_{ij}(x_{0\|t}, c_i, c_j)$
ρ	3	Polynomial degree for polynomial-modelled interaction terms
k	1	Constant for polynomial-modelled interaction terms
α	1	Scaling factor for sigmoid-modelled interaction terms
m	1	Constant for sigmoid-modelled

		interaction terms
σ	0.5	Standard deviation for gaussian-kernel-modelled interaction terms

Table 6. Configuration of hyperparameter values.

4.2 Quantitative Results

To compare the optimized multi-conditional energy guidance models—with and without interaction modelling—as well as to determine the best interaction modelling approach, we evaluated two key metrics:

1. Average Condition-Specific Distance (Euclidean Distance) – Measures how well each generated image adheres to its specific conditioning inputs.
2. Average FID Score – Measures the overall quality and realism of the generated images.

We computed these metrics across all multi-conditional combinations for a set of 100 generated images each. This ensures a fair comparison of the models in terms of both fidelity to conditions and visual quality.

4.2.1 Comparison between Interaction Models

	Text		**Facial ID**		**Segmentation Map**		**Landmark**		**Sketch**	
Interaction Models	**FID**	**Distance**	**FID**	**Distance**	**FID**	**Distance**	**FID**	**Distance**	**FID**	**Distance**
None (Primitive Model)	112.721	21.812	158.014	84.122	129.301	2331.756	139.192	17.342	119.087	328.919
Euclidean Distance	162.891	34.921	187.928	132.271	146.027	2737.189	135.928	18.911	129.139	367.716
Cosine Similarity	130.475	24.572	144.371	73.490	118.163	2147.203	128.305	21.380	115.978	317.192
Pearson Correlation	148.907	29.230	148.039	83.211	131.394	2721.823	141.283	16.101	131.371	395.908
Polynomial	86.192	17.988	102.102	71.519	91.027	2174.201	95.318	15.283	91.378	292.830
Sigmoid	102.391	21.695	117.920	68.355	101.273	2271.521	127.311	16.309	122.202	322.337
Gaussian Kernel	74.189	13.353	86.346	58.745	77.523	1824.910	81.681	14.892	89.209	248.293

Table 7. Results of the interaction models.

As shown in Table 7, multi-conditional energy guidance models with appropriate interaction modelling significantly outperformed primitive models without interaction modelling. This was evident from the lower average FID and distance values across the conditions.

Conversely, the table also highlighted that inappropriate or insufficient interaction modelling can lead to worse results. For example, some models using Euclidean distance, cosine similarity, or Pearson correlation coefficient for interaction modelling generally exhibited higher average FID and distance values compared to models without any interaction modelling.

Notably, Table 7 demonstrated that methods capable of capturing complex and non-linear interactions, such as Polynomial, Sigmoid, and Gaussian kernels, consistently outperformed simpler methods that only capture linear relationships. Among these, Gaussian kernels stood out as the most effective, yielding the lowest average FID and distance scores across all conditions.

4.2.2 Baseline Comparison

To further validate our results, we compared our proposed approximated time-independent energy function guidance, using the best interaction model (Gaussian kernel), with the baseline model TediGAN. TediGAN is a multi-modal image generation framework that enables text-guided synthesis through StyleGAN inversion and visual-linguistic similarity learning [17]. It also supports image synthesis from diverse inputs, including segmentation maps, landmarks, and sketches [17], which aligns with our scope.

	Text		**Facial ID**		**Segmentation Map**		**Landmark**		**Sketch**	
Interaction Models	**FID**	**Distance**	**FID**	**Distance**	**FID**	**Distance**	**FID**	**Distance**	**FID**	**Distance**
TediGAN [17]	71.901	12.844	82.839	59.012	81.239	2073.721	86.959	19.371	98.201	287.839
Our Model	74.189	13.353	86.346	58.745	77.523	1824.910	79.681	14.892	89.209	248.293

Table 8. Comparison Between Approximated Energy Function Guidance with the Best Interaction Model vs. Baseline (TediGAN)

From Table 8, our proposed approximated time-independent energy guidance with Gaussian kernel interaction modelling outperforms TediGAN in the segmentation map, landmark, and sketch conditions, achieving lower FID and conditional distances. However, it falls short in the text and facial ID conditions, where TediGAN attains lower FID and conditional distances. Despite this, our framework demonstrated competitive performance against a state-of-the-art image generation model, highlighting its potential for integration into existing diffusion models to enhance their effectiveness.

4.3 Qualitative Results

To demonstrate the effectiveness of our proposed model, in this section, we present qualitative results from the best-performing variant, approximated time-independent energy guidance with Gaussian kernel interaction modelling for multi-conditional image generation. Specifically, we compared images generated with and without interaction modelling, followed by a comparison between our best-performing model and the baseline, TediGAN.

4.3.1 Interaction Modelling Comparison

Visually, the images produced with interaction modelling exhibit improved visual fidelity and greater consistency in adhering to multiple constraints, demonstrating the benefits of capturing condition interdependencies.

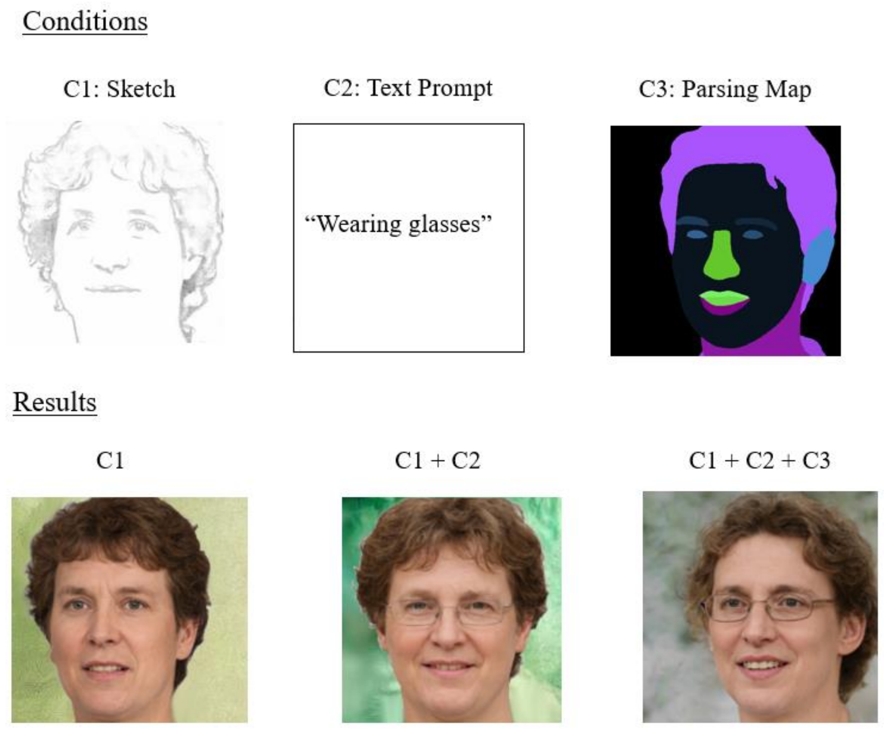


Figure 2. Illustration of Sequential Multi-Conditional Image Generation by our model

Figure 2 shows a sequence of image generation to better illustrates our model's ability to integrate multiple conditions progressively (Sketch → Sketch + Text Prompt → Sketch + Text Prompt + Segmentation Map), demonstrating how our proposed framework is able to add additional constraints refine the output while preserving visual coherence.

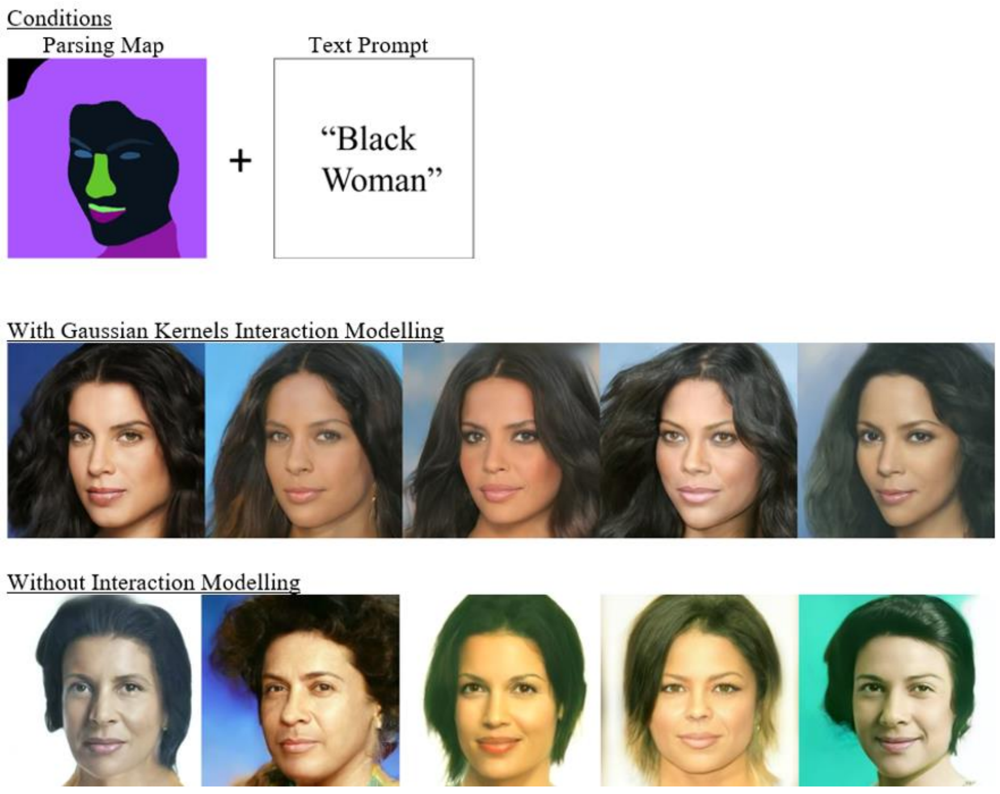

Conditions
Parsing Map
Text Prompt
+
"Black Woman"
With Gaussian Kernels Interaction Modelling
Without Interaction Modelling

Figure 3. Segmentation Map + Text Prompt Multi-Conditional Image Generation Result

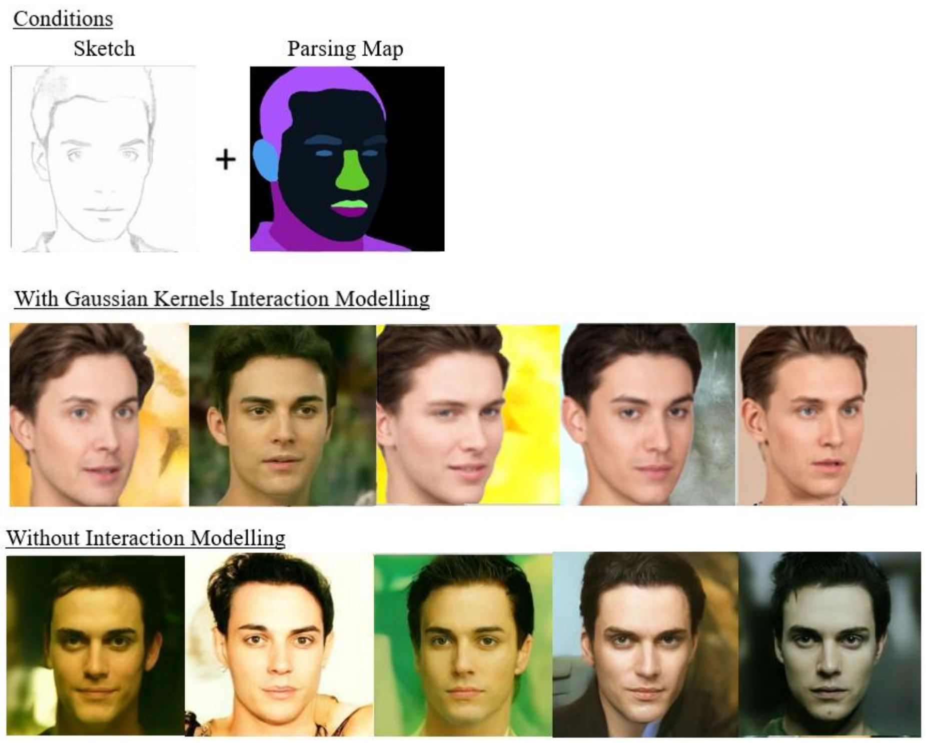

Conditions
Sketch
Parsing Map
+
With Gaussian Kernels Interaction Modelling
Without Interaction Modelling

Figure 4. Sketch + Segmentation Map Multi-Conditional Image Generation Result

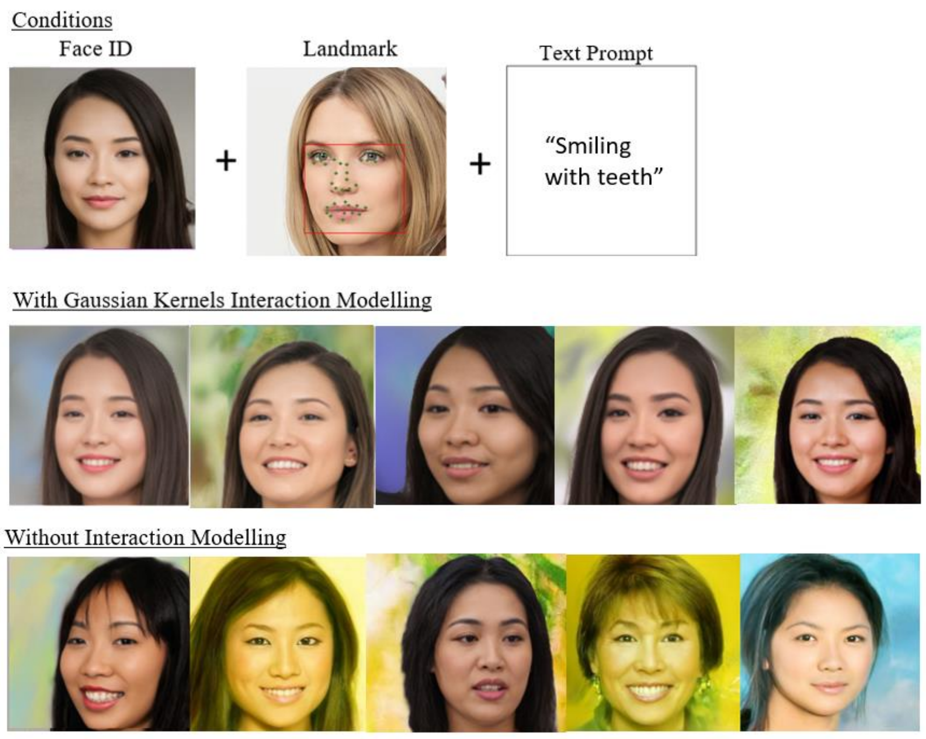

Conditions
Face ID
Landmark
Text Prompt
"Smiling with teeth"
+
+
With Gaussian Kernels Interaction Modelling
Without Interaction Modelling

Figure 5. Face ID + Landmark + Text Prompt Multi-Conditional Image Generation Result

4.3.2 Baseline Comparison

A comparison between images generated by our best-performing model and the baseline, TediGAN, revealed visually comparable results, with our model demonstrating slightly improved coherence and fidelity.

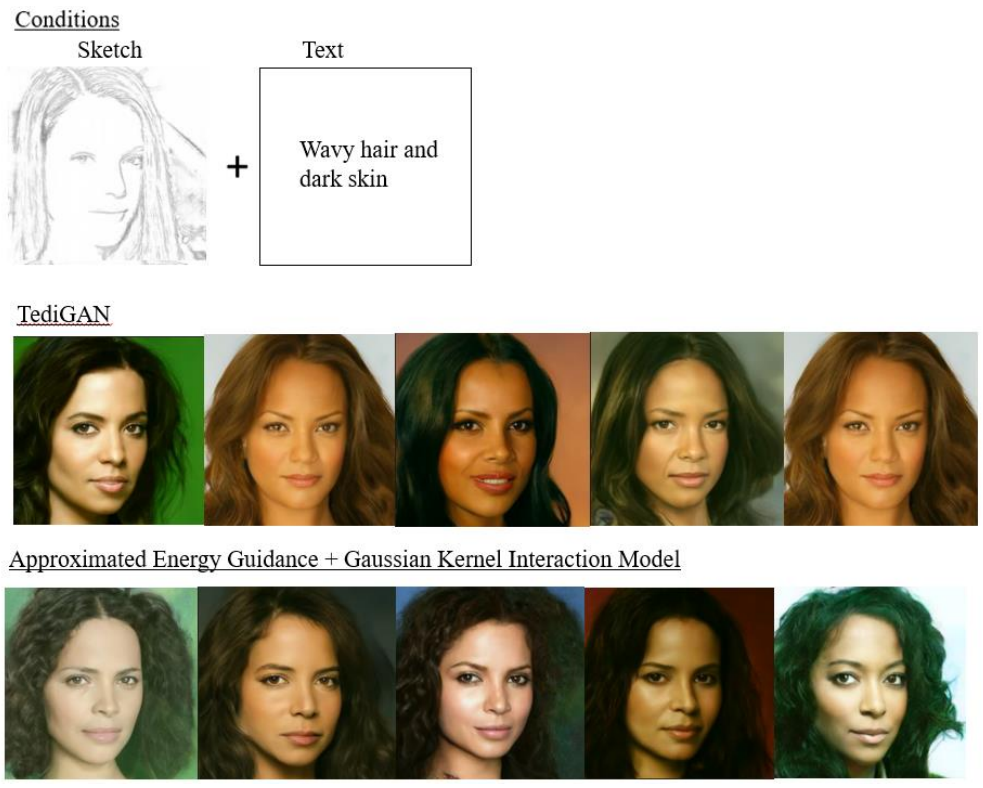

Conditions
Sketch
Text
Wavy hair and dark skin
+
TediGAN
Approximated Energy Guidance + Gaussian Kernel Interaction Model

Figure 6. Sketch + Text Prompt Baseline Comparison

5. Evaluation

5.1 Models

Based on the observed quantitative results in Section 4.2.1, we observed that modelling interactions between potentially dependent conditions is crucial for achieving significantly better outcomes in multi-conditional image generation. However, the choice of interaction modelling approach was equally critical, as inappropriate modelling can lead to worse results than omitting interaction modelling altogether.

In this section, we attempt to evaluate the suitability of various approaches to interaction modelling between diverse conditions in the context of multi-conditional image generation, analyzing the potential reasons behind the observed results

Model	**Pros**	**Cons**
Simple Similarity Measures (Euclidean Distance, Cosine Similarity, Pearson Correlation)	Simple and computationally efficient.	Fails to capture non-linear dependencies. Poor performance in high-dimensional spaces.
Polynomial Functions	Captures non-linear interactions. Flexible with adjustable degrees.	Sensitive to parameter tuning. Risk of overfitting with higher-degree polynomials. May exhibit abrupt changes for higher-degree terms, making optimization more challenging. Also ensures localized influence

		but may over-penalize small deviations due to the lack of squared distance in their formulation.
Sigmoid Functions	Captures non-linear interactions. Effective for sigmoidal or threshold-like dependencies. Useful in situations where the interactions between conditions exhibit saturating behavior.	Sensitive to parameter tuning. Limited expressiveness for highly complex interactions. Not symmetric and may introduce biases based on the direction of interactions.
Gaussian Kernels	Excellent for capturing complex, non-linear relationships without requiring explicit feature engineering [7]. Provides a smooth and differentiable measure of similarity, which is crucial for stable optimization and gradual alignment of conditions [13]. Robust and adaptable across diverse data distributions. Radial symmetry ensures that similarity depends only on the distance between conditions, not their orientation or scale, making it ideal for pairwise interactions. [7]. The exponential decay ensures that only closely related conditions strongly influence the output, minimizing the impact of irrelevant or conflicting conditions [14]. This focuses on the	Computationally more expensive than simpler methods. Requires careful bandwidth parameter selection.

	model’s attention on relevant, closely aligned conditions, preventing the overemphasis on distant or irrelevant conditions can lead to artifacts. Flexible as the bandwidth parameter σ allows fine-grained control over how much dissimilarity is tolerated before the kernel value drops significantly [13]. Adjusting σ enables the model to adapt to the specific nature of each condition.	

Table 9. Evaluation the pros and cons of each proposed interaction-modelling methods in the context of multi-conditional image generation tasks

5.2 Experimental Limitations

The image generation process was highly time-intensive due to GPU resource constraints, particularly when exploring multiple hyperparameter combinations. As illustrated in Equation (8), the number of possible configurations for interaction model-specific hyperparameters, along with the weighting factors for each condition, was extremely large. As a result, we acknowledge that our proposed grid search optimization method and the defined search space in Section 3.3.2.3 may not have been the most efficient approach for hyperparameter tuning. Consequently, the configurations presented in Section 4.1 may not represent the most optimal settings.

5.3 Future Work

Aligned with the project's focus on enhancing training-free, multi-conditional image generation, the following suggestions for future work could further improve or address the limitations of the proposed framework.

5.3.1 Anisotropic Gaussian Kernels

Given that the gaussian kernel shows the most promise, we aim to further enhance its performance. Currently, the standard gaussian kernel proposed in Equation (14) can be written as:

$$K(x,y) = e^{-\sum_{i=1}^{d} \frac{(x_i - y_i)^2}{2\sigma_i^2}} \tag{18}$$

where σ_i is assumed to be fixed across all dimensions i, reducing the hyperparameter search space to address GPU limitations and minimize computational complexity. However, with the fixed σ_i, the standard gaussian kernel assumes equal influence in all directions, but some conditions (e.g., shape vs. color) influence different aspects of the image.

An improvement to this would be to introduce anisotropic kernels, which allow varying scaling across different feature directions with separate σ_i values for each dimension i as follows:

$$K(x,y) = e^{-\frac{(x-y)^T \Sigma^{-1} (x-y)}{2}} \tag{19}$$

where Σ is predefined diagonal covariance matrix with varying σ_i values based on the perpetual importance of the different dimensions. This flexibility enables the kernel to capture the anisotropic nature of images, where different features (dimensions) may have different importance or units, requiring different levels of smoothness or variance along each axis [1]. Consequently, this modification can potentially add improvements to shape, spatial, and color constraints by ensuring conditions affect only relevant aspects of image features.

5.3.2 Dynamic and Adaptive Weight Adjustment

Given that the weighting factors account for much of the hyperparameter search space, a potential improvement is to implement dynamic, adaptive weight adjustment to reduce reliance on exhaustive optimization. One approach involves computing the gradient norms of each condition’s energy term and applying a softmax function to obtain normalized weights that reflect real-time energy contributions. For interaction terms, the similarity between gradients can indicate whether conditions are synergistic or conflicting, allowing for corresponding adjustments in their weights. This process enables automatic and efficient scaling of both condition weights and interactions, thereby mitigating the need for labor-intensive hyperparameter tuning in high-dimensional spaces.

Our initial attempts to incorporate dynamic weight adjustment revealed several critical challenges. In early iterations, the gradient signals were extremely noisy, which compromised the stability of the softmax-based normalization of energy contributions. This instability hindered our ability to reliably adjust weights in real time. Moreover, the significant variability in the scales across different conditions made it difficult to establish robust thresholds for both individual conditions and their interaction terms. As a result, the dynamic adjustments did not consistently reflect the true energy dynamics, limiting the overall effectiveness of our approach. These findings underscore the inherent complexity of automating weight scaling in such systems and highlight the need for more refined strategies to mitigate noise and scale disparities.

5.3.3 Extension to other generative tasks

Another promising avenue for future work is to extend interaction modelling in energy guided functions to other training-free multi-conditional generative tasks, such as video synthesis and 3D reconstruction.

In video synthesis, the guided energy function can be adapted to capture temporal interactions between consecutive frames. By modelling these interactions, the system can ensure smooth transitions, maintain motion coherence, and manage dynamic content effectively. This approach enables the generator to produce sequences where each frame is consistent with its neighbors, preserving the continuity and flow necessary for realistic video content.

Similarly, for 3D reconstruction, interaction modelling can be employed to capture spatial dependencies across multiple viewpoints. Here, the energy function integrates cues from different perspectives to enforce geometric and photometric consistency. This allows for the generation of volumetric representations that accurately reflect the underlying 3D structure of the scene without the need for additional training, paving the way for more flexible and efficient 3D content creation.

6. <u>Conclusion</u>

In conclusion, this work successfully addresses the limitations of training-free multi-conditional image generation in handling multi-conditional dependencies. We propose integrating interaction modelling, particularly Gaussian kernels, with approximated time-independent energy guidance functions to enhance generation quality. The proposed approach not only improves the qualitative aspects of generated images but also delivers significant quantitative gains, demonstrating its effectiveness in overcoming conventional challenges in image synthesis.

The validity of these results is supported by notable improvements in key evaluation metrics, confirming that interaction-based energy modelling effectively refines generated outputs, especially when compared to the baseline, TediGAN, a state-of-the-art image generation network. However, certain limitations remain, such as potential trade-offs in computational efficiency and adaptability to highly complex constraints. Additionally, GPU limitations may lead to suboptimal hyperparameter configurations and hinder the testing of larger, more generalized generative models. Further research is needed to evaluate the method's generalizability across diverse datasets and real-world applications.

Future work could focus on optimizing computational efficiency, extending applicability to more complex generative models, and dynamically integrating additional constraints and weighting parameters. Exploring alternative kernel methods may further enhance adaptive scaling across different feature directions. This study lays a solid foundation for advancements in training-free multi-conditional image generation, opening avenues for more robust generative applications such as video synthesis and reconstruction.

7. Project Schedule

This section provides an overview of the entirety of the project schedule, detailing how the final-year project was systematically divided into phases, with each phase further broken down into specific objectives. The project was carefully planned across the designated timeline to ensure efficient progress from inception to completion.

In the original project schedule submitted on 31st August 2024, Table 11 in Appendix B, the research focus was to evaluate various image generation neural networks (GANs, VAEs, and Diffusion models) and reimplement the best-performing variation. However, at that stage, the project direction was still exploratory, and the final research topic had yet to be solidified.

Following discussions with project mentors, the initial scope was deemed too simplistic and lacking in originality to make a meaningful contribution to the field of image generation. After the research and literature review phase, the focus of this automated image generation project shifted to “Enhancing Training-Free Multi-Conditional Image Generation through Approximated Energy-Based Guidance and Interaction Modelling in Diffusion Model Denoising”. Thus, significant revisions were made to refine the research focus and enhance its impact.

The finalized project schedule, presented in Table 10 below, is the result of a comprehensive revision after the project's completion. It now accurately reflects the actual timeline, segmented into distinct periods with corresponding objectives, ensuring a detailed and precise alignment with the project lifecycle. This revised schedule provides a clear, systematic view of the research and development process, capturing the true progression of the project.

Time Period	Objective

August 2024	**<u>Research and Literature Review Phase</u>**
	Research and Literature Review on:
	1. Neural Networks for Image Generation
	  a. GANs
	  b. Variational Autoencoders
	  c. Diffusion Models
	    i. Unconditional Score-based diffusion models
	    ii. Conditional Score-based diffusion models
	2. Performance Metrics for image generation
	  a. FID, IS, KID, PPL
	3. Learning Deep Learning Frameworks
	  a. TensorFlow
	  b. PyTorch

September 2024	**<u>Refinement of Research Scope Phase</u>**
	1. Identify potential research topics
	  a. Benchmarking different image generation neural networks (Initial).
	  b. Enhancing training-free image generation (Chosen).
	2. Research training-free image generation approaches
	  a. Classifier Guidance
	  b. Classifier-Free Guidance
	  c. Approximated Energy-Based Guidance
	  d. Latent Editing
	  e. Diffusion Inversion
	3. Assessment of training-free image generation approaches to work on
	  a. Classifier Guidance
	    i. Requires a separate pre-trained classifier, which may not generalize well across conditions.
	  b. Classifier-Free Guidance
	    i. Less effective when multiple constraints need to be enforced simultaneously.

	c. Approximated Energy-Based Guidance (Chosen) d. Latent Editing i. Requires additional optimization steps at inference, making it slow. e. Diffusion Inversion i. Can be computationally expensive and may introduce artifacts. 4. Research on approximated energy guidance. a. Time independent approximations for energy guided functions with pre-trained networks. 5. Research on the pain points of training-free image generation. a. To increase the accuracy of the Approximated Multi-Conditional Energy Guidance. 6. Finalize research topic a. "Enhancing Training-Free Multi-Conditional Image Generation through Approximated Energy-Based Guidance and Interaction Modelling in Diffusion Model Denoising." 7. Set up the project repository and define an initial experimental plan.
October 2024	**Preliminary Experiments and Search Phase** 1. Experimenting with pre-trained networks for the primary image generation model. a. Due to the limitations of my personal GPU, the denoising process for larger models like StableDiffusion and ControlNet either took too long to complete or resulted in memory exhaustion. b. Chose to use a smaller, pre-trained face generating network 2. Evaluate different pre-trained networks for condition-specific guidance (condition-specific extracting models). a. Dependent on the type of conditions to be experimented and the scope of images able to be generated by the primary image generation model. b. Chosen conditions: Face Identity, Landmark, Segmentation Map, Sketch, Text. c. Find pre-trained networks for the chosen conditions 3. Experiment with different formulas for approximating single-

	conditional energy-based guidance with time-independent distance functions.
November 2024	**<u>Multi-Conditional Energy Function Approximation & Initial Model Integration Phase</u>** 1. Develop and test different methods for approximating multi-conditional energy-based guidance with time-independent approximated distance functions. a. Initial model - extension of single-conditional energy-based guidance via weighted sum. 2. Integration of approximated multi-conditional energy-based guidance into denoising function. 3. Conduct initial qualitative evaluations to check the impact of individual condition constraints. 4. Begin logging results and debugging performance issues a. To find the optimal ratio of weighting factors of condition-specific distance guidance functions for different condition combinations.

December 2024	**<u>Interaction Modelling for Multi-Condition Dependencies Phase</u>**
	1. Research and implement different training-free interaction modelling techniques to capture dependencies between multiple conditions.
	    a. Weighted Aggregation
	    b. Euclidean Distance
	    c. Pearson Correlation
	    d. Cosine Similarity
	    e. Polynomial Kernels
	    f. Sigmoid Kernels
	    g. Gaussian Kernels
	2. Integration of interaction modelling into multi-conditional energy-based guidance function.
	3. Experiment with how different conditions influence each other and refine the weighting factors of the interaction terms.
	4. Perform initial qualitative evaluations to determine the reasonable search space for the hyperparameters of each interaction modelling technique, before conducting hyperparameter optimization through grid search.

January 2025	**<u>Model Optimization & Performance Evaluation Phase</u>** 1. Optimize all combinations of hyperparameters with grid search. 2. Conduct extensive evaluations using quantitative metrics (FID, constraint-specific distances). 3. Compare results with existing training-free image generation baselines. 4. Compare the different interaction modelling techniques to find the best. 5. Identify failure cases and areas for improvement. 6. Identify potential extension areas for the project. 7. Start drafting key findings and insights. 8. Interim Report Writing. 9. Submission of Interim Report – 27th January 2025
February 2025	**<u>Documentation & Final Experiments Phase</u>** 1. Code refactoring (Submission preparation). 2. Final Deployment. 3. Conduct final experiments to confirm findings. 4. Prepare figures, graphs, and tables for results visualization. 5. Final Report Writing a. Document methodology, experimental design, results, and key conclusions. b. Include all required deliverables mentioned in the report guide.
March 2025	**<u>Final Report Submission</u>** 1. Continue Final Report Writing 2. Thorough checks and ensure all formats are correct 3. Submission of Final Report – 24th March 2025
April 2025	**<u>Amended Final Report</u>** 1. Final amendments to the final report for submission. 2. Creation of slides for oral presentation. 3. Creation of script for oral presentation. 4. Submission of Amended Final Report – 18th April 2025
May 2025	**<u>Oral Presentation</u>** 1. Continue creation of slides for oral presentation. 2. Continue creation of script for oral presentation. 3. Rehearsals for oral presentation

	4. Oral Presentation – 9th May 2025

Table 10. Finalized Project Schedule

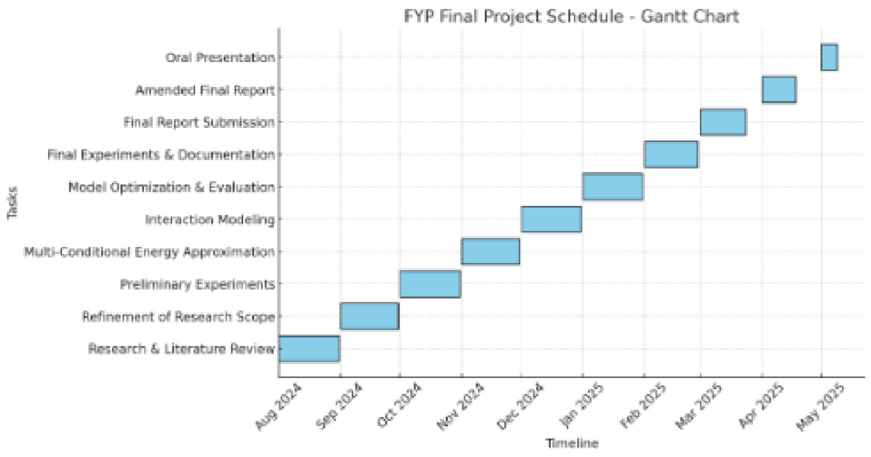


Figure 7. Gantt Chart of Finalized Project Schedule

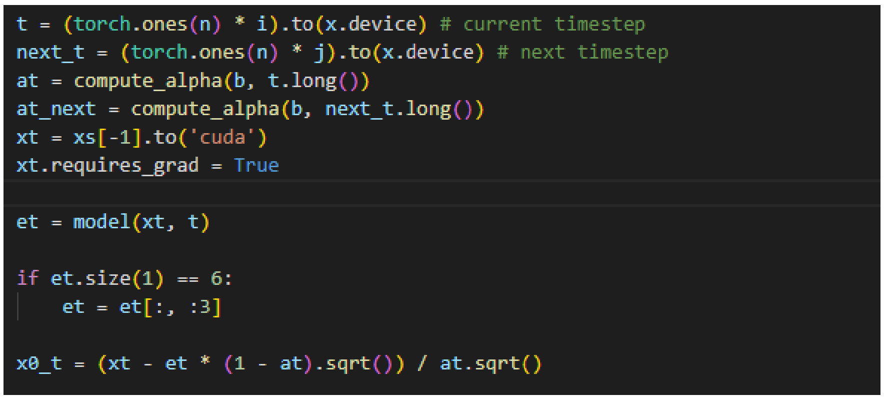
8. Appendices

8.1 Appendix A – Code Snippets

This appendix provides key code snippets that were instrumental in the implementation of the project. The following sections contain excerpts from core scripts of the proposed framework, including the time-independent approximated time energy function and the various interaction models.

8.1.1 Approximated clean image from an intermediate noisy image

This code snippet depicts the approximation formula to get a clean image x_0 corresponding to an intermediate noisy image x_t for each time step t as in Equation (6) [20]. This is crucial for the calculation of distance between the given condition and the intermediate image as the pre-trained condition networks only work well on cleaned images.

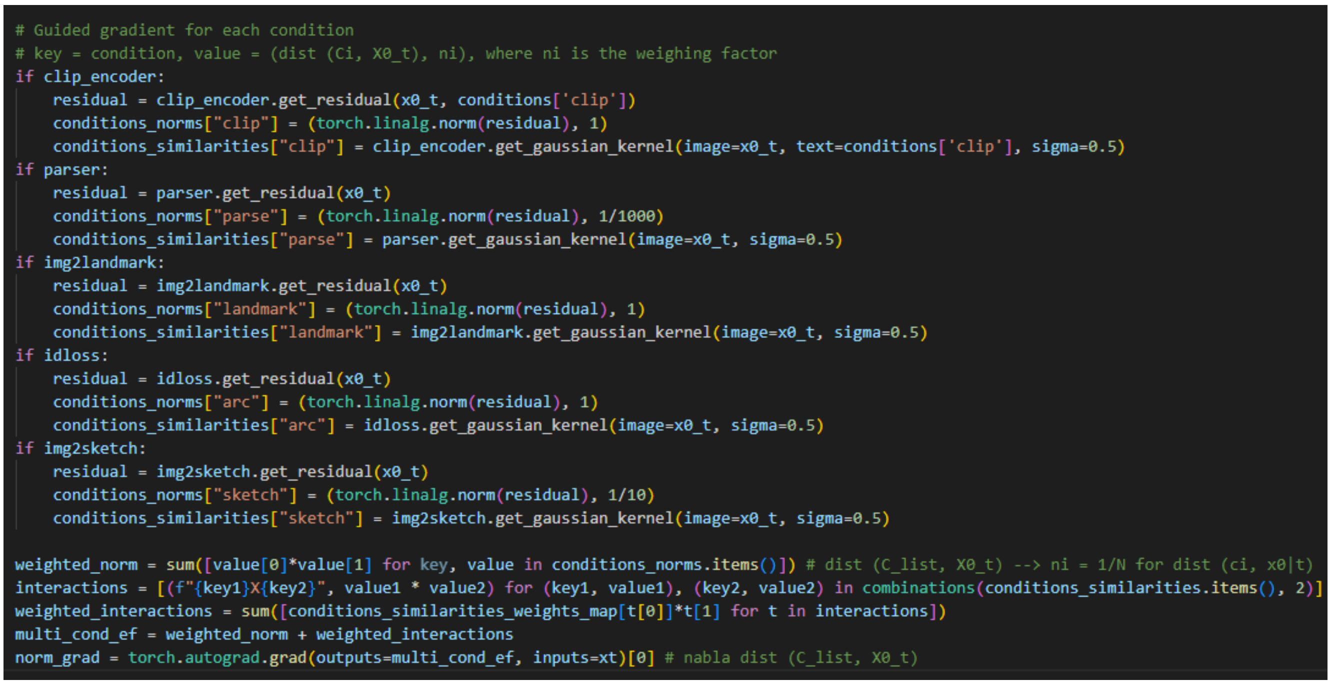


```
t = (torch.ones(n) * i).to(x.device) # current timestep
next_t = (torch.ones(n) * j).to(x.device) # next timestep
at = compute_alpha(b, t.long())
at_next = compute_alpha(b, next_t.long())
xt = xs[-1].to('cuda')
xt.requires_grad = True

et = model(xt, t)

if et.size(1) == 6:
    et = et[:, :3]

x0_t = (xt - et * (1 - at).sqrt()) / at.sqrt()
```

Figure 8. Approximated clean image from intermediate noisy image

8.1.2 Multi-Conditional Energy Guidance with Interaction Modelling

This code snippet depicts the algorithm for calculating the multi-conditional energy guidance with interaction modelling as in Equation (8).

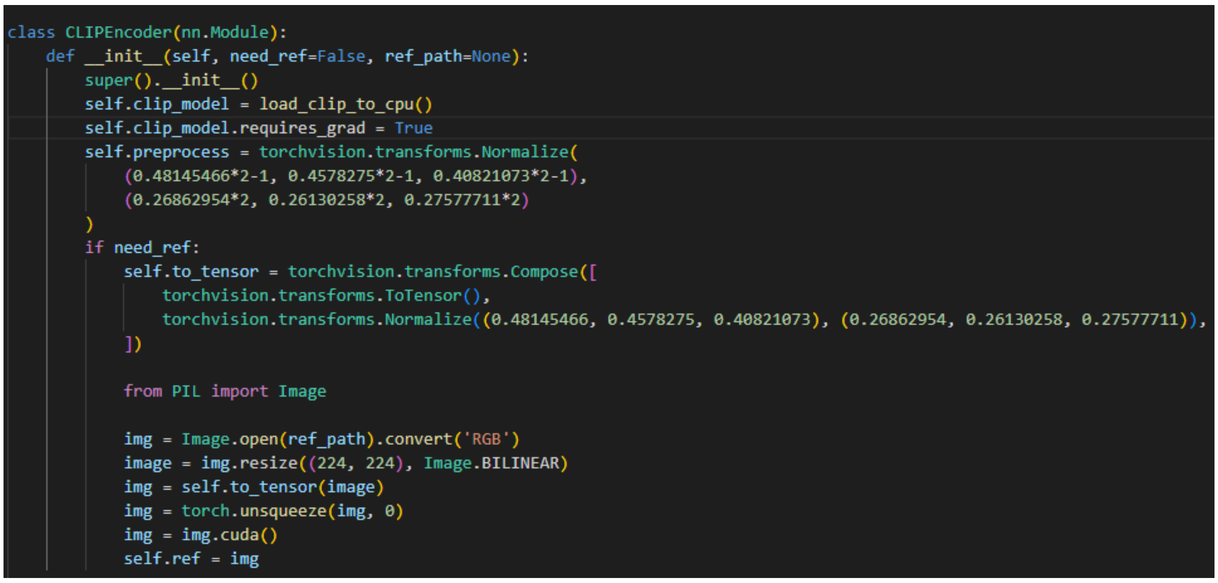


```
# Guided gradient for each condition
# key = condition, value = (dist (Ci, X0_t), ni), where ni is the weighing factor
if clip_encoder:
    residual = clip_encoder.get_residual(x0_t, conditions['clip'])
    conditions_norms["clip"] = (torch.linalg.norm(residual), 1)
    conditions_similarities["clip"] = clip_encoder.get_gaussian_kernel(image=x0_t, text=conditions['clip'], sigma=0.5)
if parser:
    residual = parser.get_residual(x0_t)
    conditions_norms["parse"] = (torch.linalg.norm(residual), 1/1000)
    conditions_similarities["parse"] = parser.get_gaussian_kernel(image=x0_t, sigma=0.5)
if img2landmark:
    residual = img2landmark.get_residual(x0_t)
    conditions_norms["landmark"] = (torch.linalg.norm(residual), 1)
    conditions_similarities["landmark"] = img2landmark.get_gaussian_kernel(image=x0_t, sigma=0.5)
if idloss:
    residual = idloss.get_residual(x0_t)
    conditions_norms["arc"] = (torch.linalg.norm(residual), 1)
    conditions_similarities["arc"] = idloss.get_gaussian_kernel(image=x0_t, sigma=0.5)
if img2sketch:
    residual = img2sketch.get_residual(x0_t)
    conditions_norms["sketch"] = (torch.linalg.norm(residual), 1/10)
    conditions_similarities["sketch"] = img2sketch.get_gaussian_kernel(image=x0_t, sigma=0.5)

weighted_norm = sum([value[0]*value[1] for key, value in conditions_norms.items()]) # dist (C_list, X0_t) --> ni = 1/N for dist (ci, x0|t)
interactions = [(f"{key1}X{key2}", value1 * value2) for (key1, value1), (key2, value2) in combinations(conditions_similarities.items(), 2)]
weighted_interactions = sum([conditions_similarities_weights_map[t[0]]*t[1] for t in interactions])
multi_cond_ef = weighted_norm + weighted_interactions
norm_grad = torch.autograd.grad(outputs=multi_cond_ef, inputs=xt)[0] # nabla dist (C_list, X0_t)
```

Figure 9. Algorithm for multi-conditional energy guidance with interaction modelling

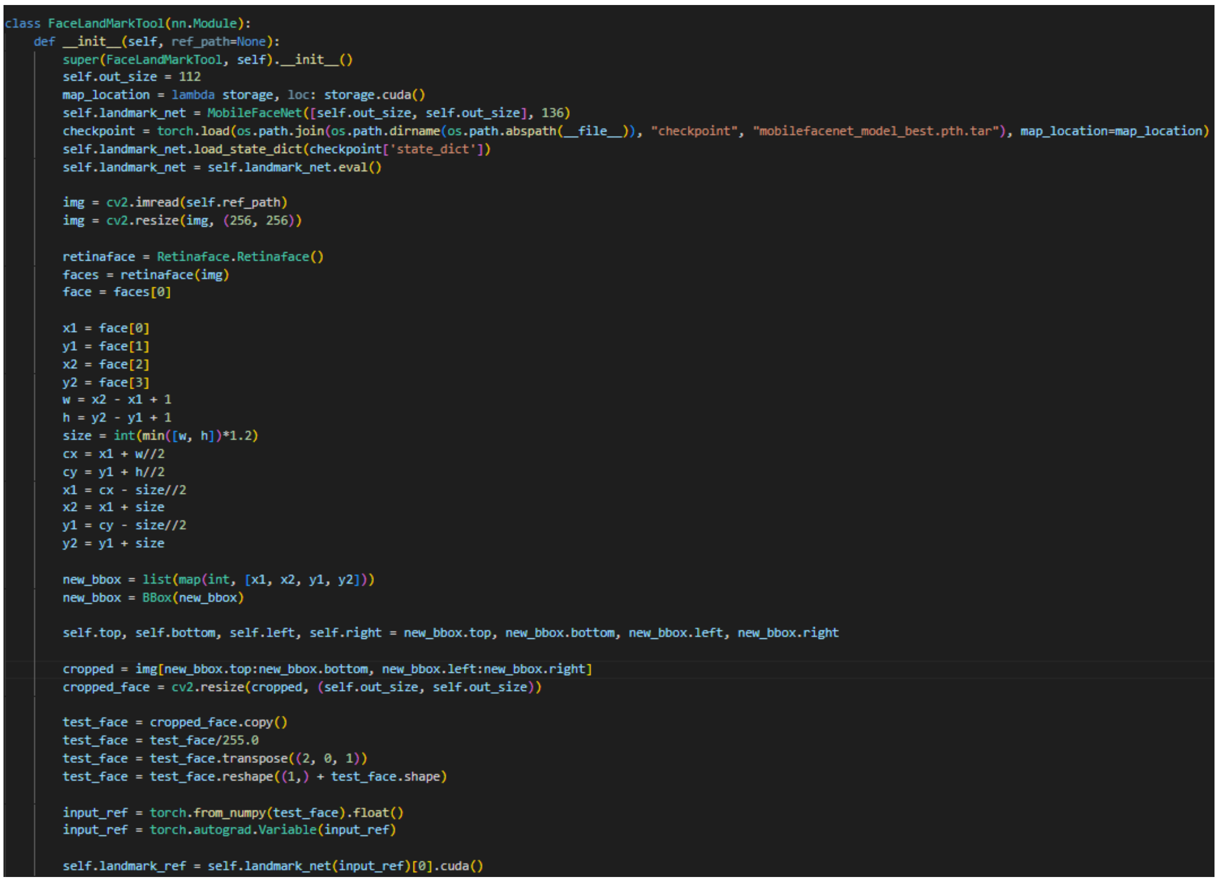
8.1.3 Condition-Specific Classes

The following code snippets (Figures 10–14) illustrate the implementation of condition-specific classes: Text, Facial Landmark, Facial Segmentation Map, Face ID, and Sketch. Each class is initialized with its respective pre-trained model and a specified instance of the condition as an image. Additionally, these classes include functions for interaction models and conditional distance calculations. Approximated clean images can be passed into these functions to compute the required information.



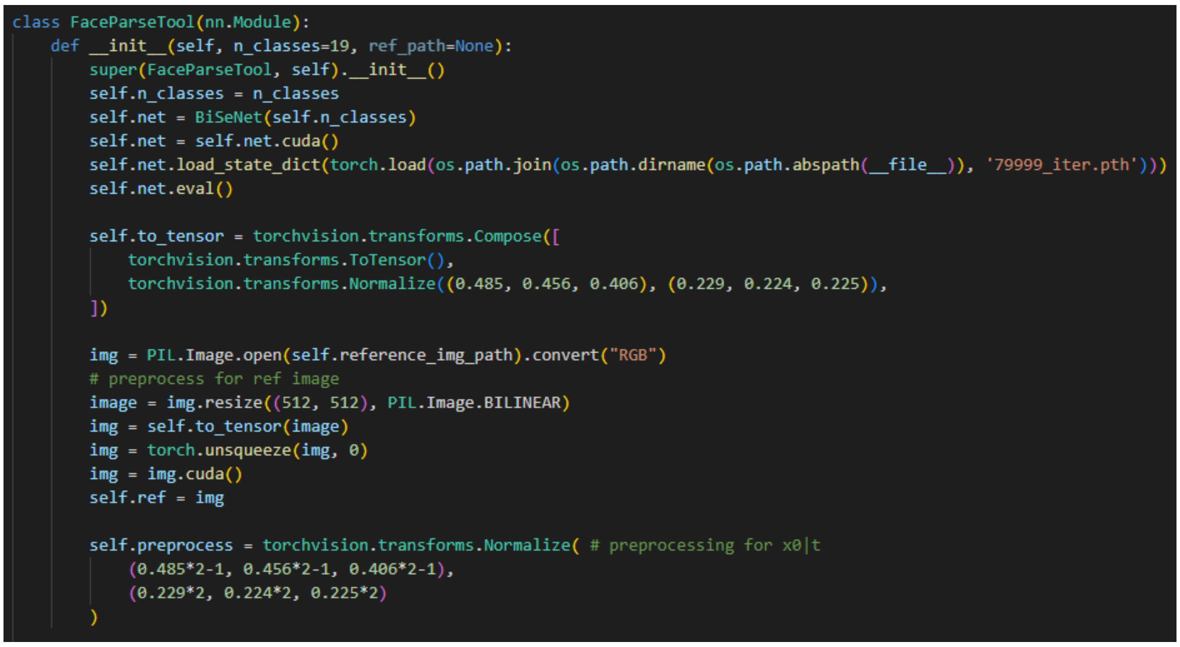
```python
class CLIPEncoder(nn.Module):
    def __init__(self, need_ref=False, ref_path=None):
        super().__init__()
        self.clip_model = load_clip_to_cpu()
        self.clip_model.requires_grad = True
        self.preprocess = torchvision.transforms.Normalize(
            (0.48145466*2-1, 0.4578275*2-1, 0.40821073*2-1),
            (0.26862954*2, 0.26130258*2, 0.27577711*2)
        )
        if need_ref:
            self.to_tensor = torchvision.transforms.Compose([
                torchvision.transforms.ToTensor(),
                torchvision.transforms.Normalize((0.48145466, 0.4578275, 0.40821073), (0.26862954, 0.26130258, 0.27577711)),
            ])

            from PIL import Image

            img = Image.open(ref_path).convert('RGB')
            image = img.resize((224, 224), Image.BILINEAR)
            img = self.to_tensor(image)
            img = torch.unsqueeze(img, 0)
            img = img.cuda()
            self.ref = img
```

Figure 10. Text condition class



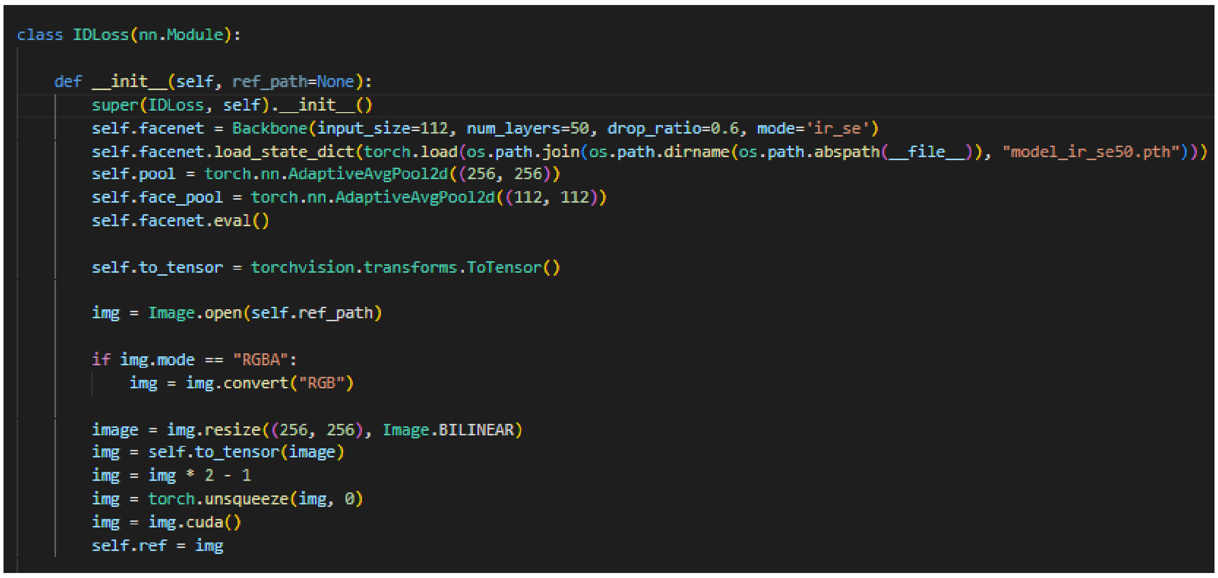
```python
class FaceLandMarkTool(nn.Module):
    def __init__(self, ref_path=None):
        super(FaceLandMarkTool, self).__init__()
        self.out_size = 112
        map_location = lambda storage, loc: storage.cuda()
        self.landmark_net = MobileFaceNet([self.out_size, self.out_size], 136)
        checkpoint = torch.load(os.path.join(os.path.dirname(os.path.abspath(__file__)), "checkpoint", "mobilefacenet_model_best.pth.tar"), map_location=map_location)
        self.landmark_net.load_state_dict(checkpoint['state_dict'])
        self.landmark_net = self.landmark_net.eval()

        img = cv2.imread(self.ref_path)
        img = cv2.resize(img, (256, 256))

        retinaface = Retinaface.Retinaface()
        faces = retinaface(img)
        face = faces[0]

        x1 = face[0]
        y1 = face[1]
        x2 = face[2]
        y2 = face[3]
        w = x2 - x1 + 1
        h = y2 - y1 + 1
        size = int(min([w, h])*1.2)
        cx = x1 + w//2
        cy = y1 + h//2
        x1 = cx - size//2
        x2 = x1 + size
        y1 = cy - size//2
        y2 = y1 + size

        new_bbox = list(map(int, [x1, x2, y1, y2]))
        new_bbox = BBox(new_bbox)

        self.top, self.bottom, self.left, self.right = new_bbox.top, new_bbox.bottom, new_bbox.left, new_bbox.right

        cropped = img[new_bbox.top:new_bbox.bottom, new_bbox.left:new_bbox.right]
        cropped_face = cv2.resize(cropped, (self.out_size, self.out_size))

        test_face = cropped_face.copy()
        test_face = test_face/255.0
        test_face = test_face.transpose((2, 0, 1))
        test_face = test_face.reshape((1,) + test_face.shape)

        input_ref = torch.from_numpy(test_face).float()
        input_ref = torch.autograd.Variable(input_ref)

        self.landmark_ref = self.landmark_net(input_ref)[0].cuda()
```

Figure 11. Facial landmark condition class



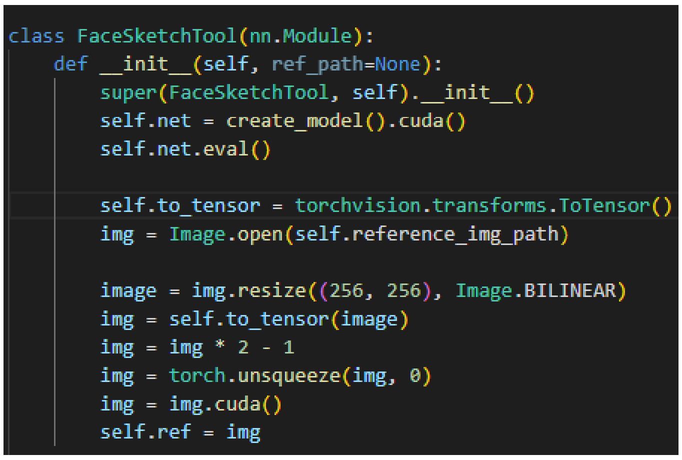
```python
class FaceParseTool(nn.Module):
    def __init__(self, n_classes=19, ref_path=None):
        super(FaceParseTool, self).__init__()
        self.n_classes = n_classes
        self.net = BiSeNet(self.n_classes)
        self.net = self.net.cuda()
        self.net.load_state_dict(torch.load(os.path.join(os.path.dirname(os.path.abspath(__file__)), '79999_iter.pth')))
        self.net.eval()

        self.to_tensor = torchvision.transforms.Compose([
            torchvision.transforms.ToTensor(),
            torchvision.transforms.Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225)),
        ])

        img = PIL.Image.open(self.reference_img_path).convert("RGB")
        # preprocess for ref image
        image = img.resize((512, 512), PIL.Image.BILINEAR)
        img = self.to_tensor(image)
        img = torch.unsqueeze(img, 0)
        img = img.cuda()
        self.ref = img

        self.preprocess = torchvision.transforms.Normalize( # preprocessing for x0|t
            (0.485*2-1, 0.456*2-1, 0.406*2-1),
            (0.229*2, 0.224*2, 0.225*2)
        )
```

Figure 12. Facial segmentation map condition class



```python
class IDLoss(nn.Module):

    def __init__(self, ref_path=None):
        super(IDLoss, self).__init__()
        self.facenet = Backbone(input_size=112, num_layers=50, drop_ratio=0.6, mode='ir_se')
        self.facenet.load_state_dict(torch.load(os.path.join(os.path.dirname(os.path.abspath(__file__)), "model_ir_se50.pth")))
        self.pool = torch.nn.AdaptiveAvgPool2d((256, 256))
        self.face_pool = torch.nn.AdaptiveAvgPool2d((112, 112))
        self.facenet.eval()

        self.to_tensor = torchvision.transforms.ToTensor()

        img = Image.open(self.ref_path)

        if img.mode == "RGBA":
            img = img.convert("RGB")

        image = img.resize((256, 256), Image.BILINEAR)
        img = self.to_tensor(image)
        img = img * 2 - 1
        img = torch.unsqueeze(img, 0)
        img = img.cuda()
        self.ref = img
```

Figure 13. Facial ID condition class

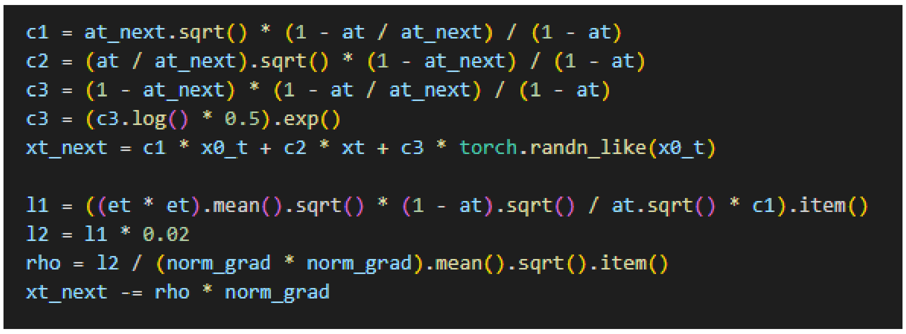


```python
class FaceSketchTool(nn.Module):
    def __init__(self, ref_path=None):
        super(FaceSketchTool, self).__init__()
        self.net = create_model().cuda()
        self.net.eval()

        self.to_tensor = torchvision.transforms.ToTensor()
        img = Image.open(self.reference_img_path)

        image = img.resize((256, 256), Image.BILINEAR)
        img = self.to_tensor(image)
        img = img * 2 - 1
        img = torch.unsqueeze(img, 0)
        img = img.cuda()
        self.ref = img
```

Figure 14. Sketch condition class

8.1.4 Final Denoising Formula

This code snippet depicts the final denoising formula with energy guidance as in Equation (4) [20].

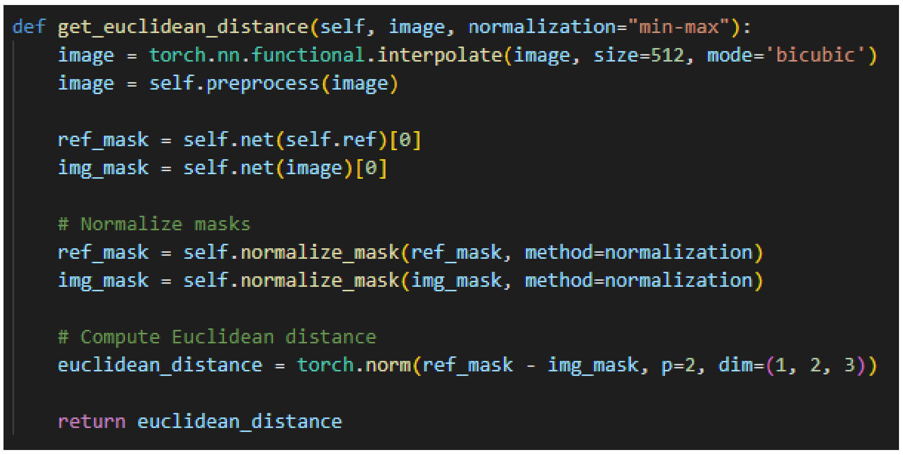


```
c1 = at_next.sqrt() * (1 - at / at_next) / (1 - at)
c2 = (at / at_next).sqrt() * (1 - at_next) / (1 - at)
c3 = (1 - at_next) * (1 - at / at_next) / (1 - at)
c3 = (c3.log() * 0.5).exp()
xt_next = c1 * x0_t + c2 * xt + c3 * torch.randn_like(x0_t)

l1 = ((et * et).mean().sqrt() * (1 - at).sqrt() / at.sqrt() * c1).item()
l2 = l1 * 0.02
rho = l2 / (norm_grad * norm_grad).mean().sqrt().item()
xt_next -= rho * norm_grad
```

Figure 15. Final denoising formula

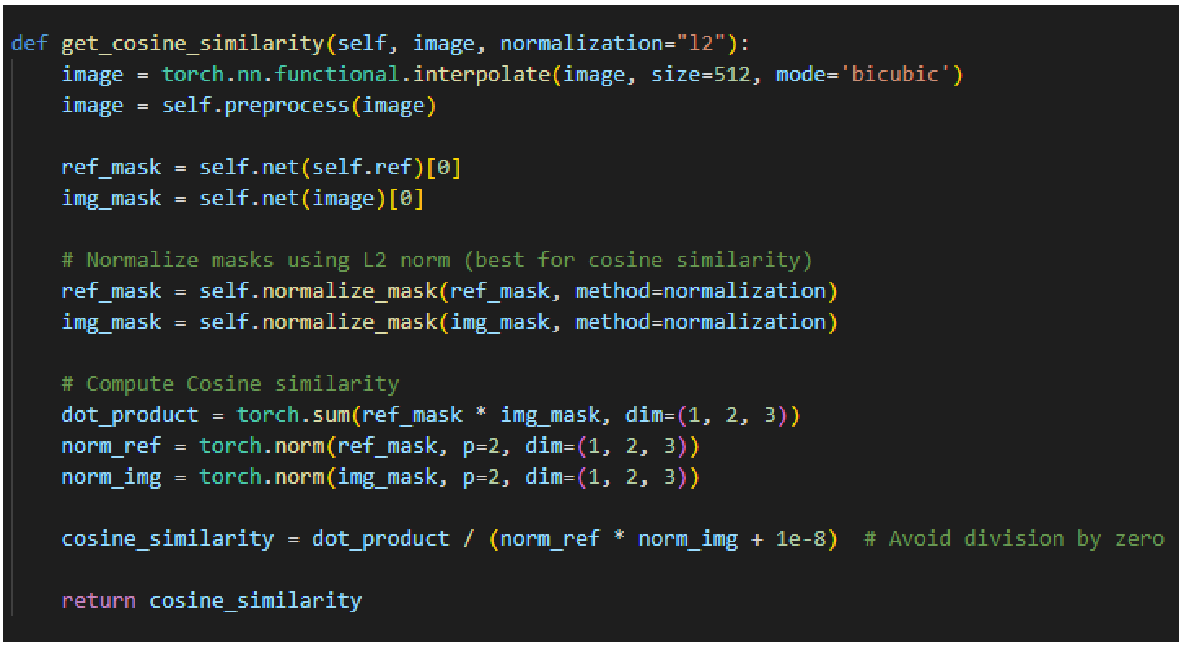
8.1.5 Interaction Models

The following code snippets (Figures 16–21) illustrate the implementation of various interaction models and similarity functions: Euclidean distance, cosine similarity, Pearson correlation coefficient, polynomial kernel, and Gaussian kernel. These figures specifically depict the interaction model class functions within the facial segmentation map class. While similar functions exist for other conditional classes—such as text, landmarks, face ID, and sketches—the calculations vary slightly due to differences in the output characteristics of each pre-trained model. However, the underlying logic remains consistent across all classes.



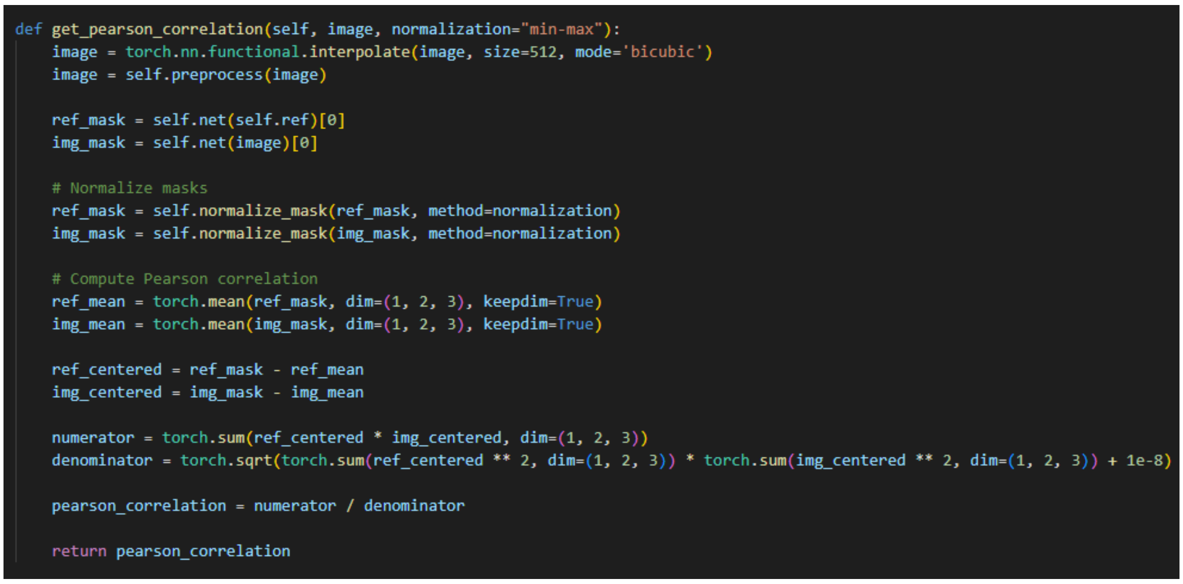
```
def get_euclidean_distance(self, image, normalization="min-max"):
    image = torch.nn.functional.interpolate(image, size=512, mode='bicubic')
    image = self.preprocess(image)

    ref_mask = self.net(self.ref)[0]
    img_mask = self.net(image)[0]

    # Normalize masks
    ref_mask = self.normalize_mask(ref_mask, method=normalization)
    img_mask = self.normalize_mask(img_mask, method=normalization)

    # Compute Euclidean distance
    euclidean_distance = torch.norm(ref_mask - img_mask, p=2, dim=(1, 2, 3))

    return euclidean_distance
```

Figure 16. Euclidean distance interaction model



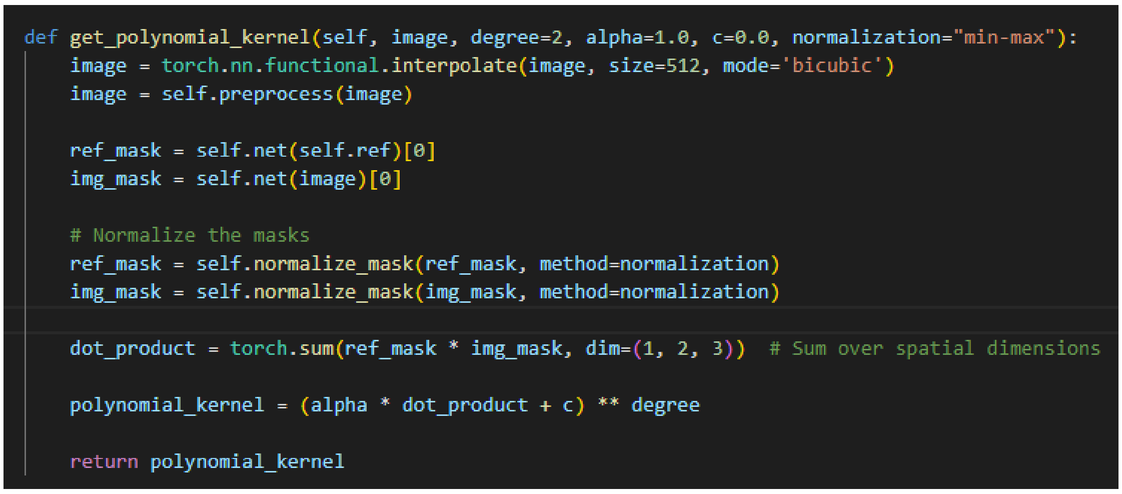
```python
def get_cosine_similarity(self, image, normalization="l2"):
    image = torch.nn.functional.interpolate(image, size=512, mode='bicubic')
    image = self.preprocess(image)

    ref_mask = self.net(self.ref)[0]
    img_mask = self.net(image)[0]

    # Normalize masks using L2 norm (best for cosine similarity)
    ref_mask = self.normalize_mask(ref_mask, method=normalization)
    img_mask = self.normalize_mask(img_mask, method=normalization)

    # Compute Cosine similarity
    dot_product = torch.sum(ref_mask * img_mask, dim=(1, 2, 3))
    norm_ref = torch.norm(ref_mask, p=2, dim=(1, 2, 3))
    norm_img = torch.norm(img_mask, p=2, dim=(1, 2, 3))

    cosine_similarity = dot_product / (norm_ref * norm_img + 1e-8)  # Avoid division by zero

    return cosine_similarity
```

Figure 17. Cosine similarity interaction model



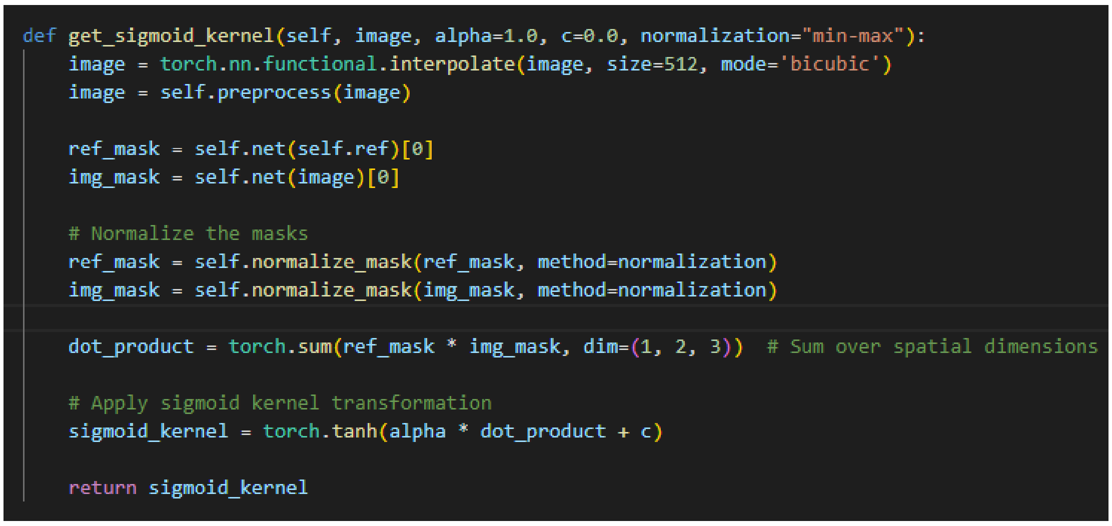
```python
def get_pearson_correlation(self, image, normalization="min-max"):
    image = torch.nn.functional.interpolate(image, size=512, mode='bicubic')
    image = self.preprocess(image)

    ref_mask = self.net(self.ref)[0]
    img_mask = self.net(image)[0]

    # Normalize masks
    ref_mask = self.normalize_mask(ref_mask, method=normalization)
    img_mask = self.normalize_mask(img_mask, method=normalization)

    # Compute Pearson correlation
    ref_mean = torch.mean(ref_mask, dim=(1, 2, 3), keepdim=True)
    img_mean = torch.mean(img_mask, dim=(1, 2, 3), keepdim=True)

    ref_centered = ref_mask - ref_mean
    img_centered = img_mask - img_mean

    numerator = torch.sum(ref_centered * img_centered, dim=(1, 2, 3))
    denominator = torch.sqrt(torch.sum(ref_centered ** 2, dim=(1, 2, 3)) * torch.sum(img_centered ** 2, dim=(1, 2, 3)) + 1e-8)

    pearson_correlation = numerator / denominator

    return pearson_correlation
```

Figure 18. Pearson correlation interaction model



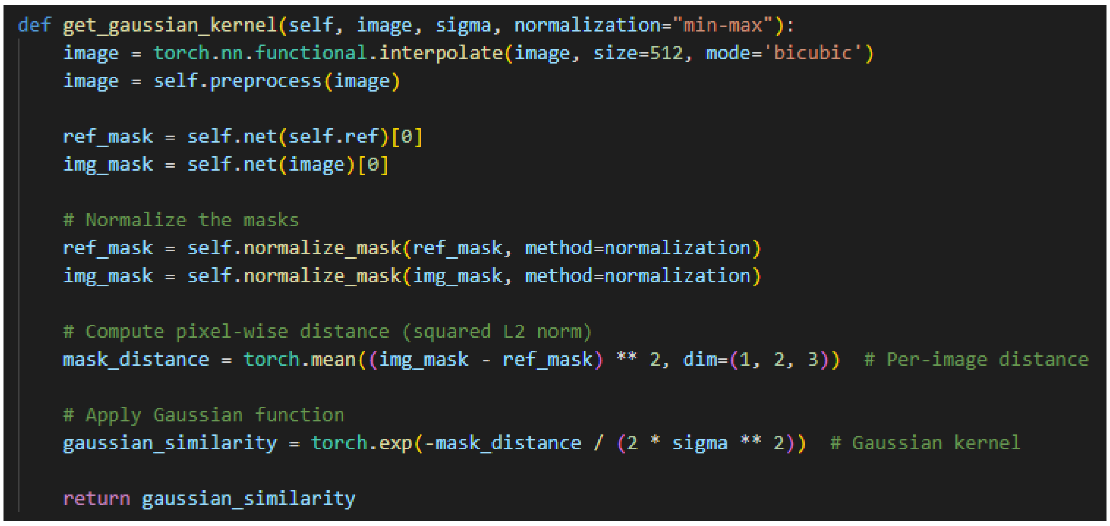
```python
def get_polynomial_kernel(self, image, degree=2, alpha=1.0, c=0.0, normalization="min-max"):
    image = torch.nn.functional.interpolate(image, size=512, mode='bicubic')
    image = self.preprocess(image)

    ref_mask = self.net(self.ref)[0]
    img_mask = self.net(image)[0]

    # Normalize the masks
    ref_mask = self.normalize_mask(ref_mask, method=normalization)
    img_mask = self.normalize_mask(img_mask, method=normalization)

    dot_product = torch.sum(ref_mask * img_mask, dim=(1, 2, 3))  # Sum over spatial dimensions

    polynomial_kernel = (alpha * dot_product + c) ** degree

    return polynomial_kernel
```

Figure 19. Polynomial function interaction model



```python
def get_sigmoid_kernel(self, image, alpha=1.0, c=0.0, normalization="min-max"):
    image = torch.nn.functional.interpolate(image, size=512, mode='bicubic')
    image = self.preprocess(image)

    ref_mask = self.net(self.ref)[0]
    img_mask = self.net(image)[0]

    # Normalize the masks
    ref_mask = self.normalize_mask(ref_mask, method=normalization)
    img_mask = self.normalize_mask(img_mask, method=normalization)

    dot_product = torch.sum(ref_mask * img_mask, dim=(1, 2, 3))  # Sum over spatial dimensions

    # Apply sigmoid kernel transformation
    sigmoid_kernel = torch.tanh(alpha * dot_product + c)

    return sigmoid_kernel
```

Figure 20. Sigmoid function interaction model



```python
def get_gaussian_kernel(self, image, sigma, normalization="min-max"):
    image = torch.nn.functional.interpolate(image, size=512, mode='bicubic')
    image = self.preprocess(image)

    ref_mask = self.net(self.ref)[0]
    img_mask = self.net(image)[0]

    # Normalize the masks
    ref_mask = self.normalize_mask(ref_mask, method=normalization)
    img_mask = self.normalize_mask(img_mask, method=normalization)

    # Compute pixel-wise distance (squared L2 norm)
    mask_distance = torch.mean((img_mask - ref_mask) ** 2, dim=(1, 2, 3))  # Per-image distance

    # Apply Gaussian function
    gaussian_similarity = torch.exp(-mask_distance / (2 * sigma ** 2))  # Gaussian kernel

    return gaussian_similarity
```

Figure 21. Gaussian kernel interaction model

8.2 Appendix B – Old Project Schedule

Time Period	Objectives

August 2024	**<u>Background Phase</u>**
	1. Literature Review
	  a. Deep Learning Models (GANs, VAEs, Diffusion, ...etc.)
	  b. Performance Metrics (FID, IS, KID, PPL, ...etc.)
	2. Learning Deep Learning Frameworks
	  a. TensorFlow
	  b. PyTorch
	3. Understanding Image Processing Libraries
	  a. OpenCV
	  b. Pillow

September 2024	**Background Phase** 1. Literature Review a. Deep Learning Models (GANs, VAEs, Diffusion, ...etc.) b. Performance Metrics (FID, IS, KID, PPL, ...etc.) 2. Learning Deep Learning Frameworks a. TensorFlow b. PyTorch 3. Understanding Image Processing Libraries a. OpenCV b. Pillow
October 2024	**Reimplementation Phase** - Build automated image generation models for the learned deep learning models **Benchmarking Phase** - Assess every model built with a set of appropriate assessment metrics **Analysis Phase** - Performance and limitation analysis of models - Comparative analysis
November 2024	**Reimplementation Phase** - Build automated image generation models for the learned deep learning models **Benchmarking Phase** - Assess every model built with a set of appropriate assessment metrics **Analysis Phase** - Performance and limitation analysis of models - Comparative analysis
December 2024	**Reimplementation Phase** - Build automated image generation models for the learned deep learning models **Benchmarking Phase** - Assess every model built with a set of appropriate assessment metrics **Analysis Phase** - Performance and limitation analysis of models

	- Comparative analysis
January 2025	**Innovation Phase** - Literature Review of new scarcely-researched for (potential) implementation and analysis - (Potential) modifications to existing methods to address identified limitations **Final Development Phase** - Finalize best performing automated image generation model. - Create a functioning interface for the best performing automated image generation model - Deployment Submission of Interim Report – 27th January 2025
February 2025	**Project Finalization Phase** - Code refactoring (Submission preparation) - Final Deployment - Report Finalization - Work on Presentation
March 2025	**Presentation Preparation and rehearsal** Submission of Final Report – 24th March 2025
April 2025	**Presentation Preparation and rehearsal** Submission of Amended Final Report – 18th April 2025
May 2025	**Oral Presentation** Oral Presentation – 9, 12-14th May (To be Confirmed)

Table 11. Old Project Schedule, 31 August 2024

9. References

[1] Berry, T., and Sauer, T., 'Local kernels and the geometric structure of data', *arXiv*, https://doi.org/10.48550/arXiv.1407.1426, 2014.

[2] Chen, C., 'PyTorch Face Landmark: A fast and accurate facial landmark detector', *github*, https://github.com/cunjian/pytorch_face_landmark, 2021.

[3] Chung, H., Sim, B., and Ye, J. C., 'Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction', *arXiv*, https://doi.org/10.48550/arXiv.2112.05146, 2022.

[4] Deng, J., Guo, J., Xue, N., and Zafeiriou, S., 'Arcface: Additive angular margin loss for deep face recognition', *arXiv*, https://doi.org/10.48550/arXiv.1801.07698, 2019.

[5] Dhariwal, P., and Nichol, A., 'Diffusion models beat gans on image synthesis', *arXiv*, https://doi.org/10.48550/arXiv.2105.05233, 2021.

[6] Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., Cohen-Or, D., 'Prompt-to-Prompt Image Editing with Cross Attention Control', *arXiv*, https://doi.org/10.48550/arXiv.2208.01626, 2022

[7] Hofmann, T., Schölkopf, B., and Smola, A. J., 'Kernel methods in machine learning', *arXiv*, https://doi.org/10.48550/arXiv.math/0701907, 2008.

[8] Kim, G., Kwon, T., and Ye, J., 'DiffusionCLIP: Text-Guided Diffusion Models for Robust Image Manipulation', *arXiv*, https://doi.org/10.48550/arXiv.2110.02711, 2021

[9] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F., 'A tutorial on energy-based learning', *MIT Press*, http://yann.lecun.com, 2006.

[10] Lin, H.-T., and Lin, C.-J., 'A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods', 2003

[11] Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S., 'SDEdit: Guided image synthesis and editing with stochastic differential equations', *arXiv*, https://doi.org/10.48550/arXiv.2108.01073, 2022.

[12] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al., 'Learning transferable visual models from natural language supervision', *arXiv*, https://doi.org/10.48550/arXiv.2103.00020, 2021.

[13] Rasmussen, C. E., and Williams, C. K. I., 'Gaussian processes for machine learning', *MIT Press*, https://doi.org/10.7551/mitpress/3206.001.0001, 2006.

[14] Schölkopf, B., & Smola, A. J., ‘Learning with kernels: Support Vector Machines, regularization, optimization, and beyond.’, *MIT Press*,

https://doi.org/10.7551/mitpress/4175.001.0001, 2001

[15] Shawe-Taylor, J., and Cristianini, N., 'Kernel methods for pattern analysis', *Cambridge University Press*, https://doi.org/10.1017/CBO9780511809682, 2004.

[16] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B., 'Score-based generative modelling through stochastic differential equations', *arXiv*, https://doi.org/10.48550/arXiv.2011.13456, 2021.

[17] Xia, W., Yang, Y., Xue, J.-H., and Wu, B., 'TediGAN: Text-guided diverse face image generation and manipulation', *arXiv*, https://doi.org/10.48550/arXiv.2012.03308, 2021.

[18] Xiang, X., Liu, D., Yang, X., Zhu, Y., Shen, X., and Allebach, J. P, 'Adversarial open domain adaptation for sketch-to-photo synthesis', *arXiv*, https://doi.org/10.48550/arXiv.2104.05703, 2022.

[19] Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., and Sang, N., 'Bisenet: Bilateral segmentation network for real-time semantic segmentation', *arXiv*, https://doi.org/10.48550/arXiv.1808.00897, 2018.

[20] Yu, J., Wang, Y., Zhao, C., Ghanem, B., and Zhang, J., 'FreeDoM: Training-free energy-guided conditional diffusion model', *arXiv*, https://doi.org/10.48550/arXiv.2303.09833, 2023.

[21] Zhang, L., Rao, A., Agrawala, M., 'Adding Conditional Control to Text-to-Image Diffusion Models', *arXiv*, https://doi.org/10.48550/arXiv.2302.05543, 2023.