A Simplified Implementation of DiffusionCLIP

Felipe Bartelt de Assis Pessoa [*] [1]

Abstract

This report presents a simplified implementation of DiffusionCLIP, a method for text-driven image manipulation based on diffusion models and CLIP guidance. Our approach reconstructs key components of the original framework—DDPM/ DDIM sampling, latent inversion, and directional CLIP-based fine-tuning—while intentionally relying on a lightweight diffusion model rather than a large pretrained backbone. Although our experimental results do not match the visual quality or precision of the original DiffusionCLIP work, the system is nonetheless able to generate images that reflect the semantic direction specified by a target prompt. The produced edits qualitatively resemble the intended transformations, demonstrating that even a minimal diffusion architecture combined with CLIP guidance can capture meaningful semantic shifts. We discuss the limitations of the simplified model, the gap relative to state-of-the-art performance, and possible improvements for future work.

1. Introduction

Diffusion models have emerged as one of the most powerful families of generative models in recent years, achieving state-of-the-art results in image, audio, and video synthesis (Ho et al., 2020; Song et al., 2022). These models generate data by reversing a gradual noising process: starting from pure noise, a neural network iteratively denoises the sample until it converges to a clean image. Their success is largely rooted in the stability of the training objective, the expressiveness of the noise prediction network, and the flexibility of the underlying probabilistic formulation.

[*]Equal contribution [1]Graduate Program in Electrical Engineering - Universidade Federal de Minas Gerais - Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil. Correspondence to: Felipe Bartelt de Assis Pessoa <fbartelt@ufmg.br>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

1.1. Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Models (DDPMs) (Ho et al., 2020) define a forward process in which Gaussian noise is added to an image over T steps. The forward Markov chain

$$q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}\, x_{t-1},\ (1-\alpha_t)I)\,,$$

progressively destroys structure until x_T becomes nearly isotropic noise. A key property of DDPMs is that the marginal distribution $q(x_t \mid x_0)$ admits the closed-form

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I),$$

where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. This allows sampling arbitrary noisy versions of an image in a single step.

The reverse process is learned by training a neural network $\epsilon_\theta(x_t, t)$ to predict the noise component ϵ from a noisy observation. The generative sampling procedure applies the learned reverse transition

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t), \Sigma_t\big),$$

iteratively denoising the sample until x_0 is reached.

Although DDPMs produce high-quality samples, the original sampling process requires hundreds or thousands of steps, which limits practical efficiency.

1.2. Deterministic Sampling via DDIM

Denoising Diffusion Implicit Models (DDIM) (Song et al., 2022) reinterpret the reverse process as a non-Markovian dynamical system. By removing the stochasticity in the reverse transitions, DDIM yields a deterministic update rule that preserves sample quality while dramatically reducing the number of required denoising steps. The resulting sampler behaves analogously to an implicit ODE solver, producing consistent trajectories and enabling techniques such as "DDIM inversion," in which a real image is mapped into a latent diffusion state at an arbitrary timestep. This inversion mechanism is central to text-guided manipulation methods such as DiffusionCLIP (Kim et al., 2022), which leverage intermediate noisy representations as starting points for editing or fine-tuning.

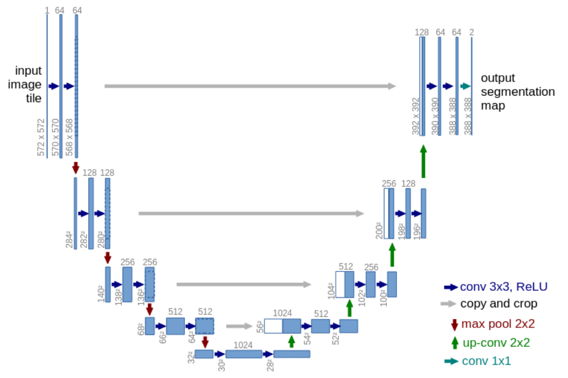


Figure 1: Original U-net architecture (Ronneberger et al., 2015)

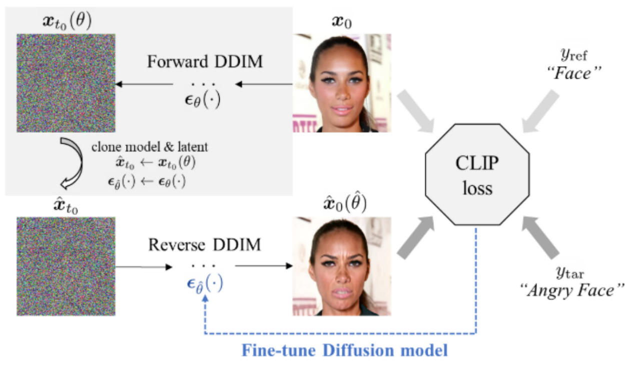


Figure 2: Overview of DiffusionCLIP. The input image is first converted to the latent via diffusion models. Then, guided by directional CLIP loss, the diffusion model is fine-tuned, and the updated sample is generated during reverse diffusion (Kim et al., 2022).

1.3. The UNet Backbone

At the heart of modern diffusion models lies the UNet architecture (Ronneberger et al., 2015), shown in Figure 1, a multiscale convolutional network that excels in dense prediction tasks. Its symmetric downsampling–upsampling structure allows information to flow across multiple spatial resolutions, which is crucial for capturing both fine details and global structure in images. Skip connections preserve high-frequency features, while time-conditioning layers inject timestep information into every resolution block. In our work, we adopt a compact UNet similar to those commonly used for lightweight or educational diffusion implementations.

1.4. CLIP as a Semantic Guide

Contrastive Language–Image Pretraining (CLIP) (Radford et al., 2021) is a dual-encoder model trained to align images and natural-language descriptions in a shared embedding space. Given an image x and a text prompt p, CLIP produces embeddings $e_{\text{img}}(x)$ and $e_{\text{text}}(p)$ whose cosine similarity reflects semantic alignment.

This cross-modal embedding space enables powerful text-driven control signals for generative models. By measuring semantic direction vectors—e.g., the difference between "smiling face" and "neutral face" embeddings—one can define gradient-based objectives that encourage a generated image to move toward a target concept while maintaining identity. DiffusionCLIP (Kim et al., 2022) introduced a framework for performing such fine-tuning efficiently by applying CLIP-based losses within the diffusion sampling trajectory, rather than optimizing in pixel space alone.

1.5. Our Goal

In this work we combine DDIM sampling, DDIM inversion, CLIP-guided directional losses, and a lightweight UNet backbone to implement a GPU-efficient adaptation of DiffusionCLIP. The method allows a pretrained diffusion model to be fine-tuned toward semantic targets while requiring only modest computational resources, making it accessible for experimentation and analysis.

2. DiffusionCLIP: Text-Guided Image Editing via Diffusion

2.1. Overview of DiffusionCLIP

Kim et al. (2022) propose *DiffusionCLIP*, a method to adapt diffusion models (originally trained purely for generation) to perform semantic edits guided by natural-language prompts. The core idea is to steer the diffusion reverse trajectory toward a target concept using gradients derived from a cross-modal semantic embedding provided by a pretrained image–text model (e.g., CLIP).

Concretely, given (1) a pretrained diffusion model trained on a generic image dataset, and (2) a pretrained contrastive image–text model (CLIP), DiffusionCLIP defines a *directional loss* in CLIP embedding space that encourages the edited image to move toward a textual target while preserving relevant characteristics of the original image (e.g. identity, structure). The method interleaves this semantic guidance with the denoising process — either by modifying the latent during reverse diffusion (at sampling time), or by fine-tuning the diffusion model's weights so future generations produce images aligned with the target concept.

2.2. Main Components

The key components of DiffusionCLIP are:

- **Pretrained diffusion model:** a standard DDPM/UNet architecture trained to denoise images.

- **DDIM sampling/inversion:** since DDIM provides a deterministic mapping between latents and images, it supports both classical sampling from noise and inversion of real images into latent space — enabling "seen $\rightarrow$ seen" editing.
- **Pretrained CLIP model:** used to embed images and text into a shared embedding space, enabling semantic comparison and directional guidance.
- **Directional CLIP loss:** by comparing embedding differences between edited and reference images, and between target and source text prompts, one can define a loss that aligns image changes with the desired semantic direction.
- **Identity / reconstruction regularization:** to prevent overfitting or excessive distortion, the method uses a loss (e.g. L1 or perceptual) that penalizes divergence from the reference image.
- **Fine-tuning or guided sampling:** either adjust the diffusion model's weights or steer sampling on-the-fly. In the fine-tuning variant, the model gradually adapts so that subsequent generations naturally follow the target semantics without repeated CLIP guidance.

A high-level diagram is provided in Figure 2: the reference image is inverted via DDIM to a latent, the latent is processed by the reverse diffusion trajectory while applying CLIP-based gradient updates or noise-injection guidance, yielding an edited image at step 0.

2.3. Loss Functions and Optimization

Let x_{ref} be the original image, and p_{src}, p_{tgt} be the source and target text prompts. Denote by $e_{\text{img}}(\cdot)$ and $e_{\text{text}}(\cdot)$ the CLIP image and text encoders. The directional text vector is:

$$d_{\text{text}} = \frac{e_{\text{text}}(p_{\text{tgt}}) - e_{\text{text}}(p_{\text{src}})}{\|e_{\text{text}}(p_{\text{tgt}}) - e_{\text{text}}(p_{\text{src}})\|}.$$

During reverse diffusion (or after decoding a latent), the current image estimate $\hat{x}$ is embedded, and the image-direction vector is computed:

$$d_{\text{img}} = \frac{e_{\text{img}}(\hat{x}) - e_{\text{img}}(x_{\text{ref}})}{\|e_{\text{img}}(\hat{x}) - e_{\text{img}}(x_{\text{ref}})\|}.$$

The directional CLIP loss is:

$$\mathcal{L}_{\text{dir}} = 1 - \cos(d_{\text{img}}, d_{\text{text}}),$$

which encourages the image modification to follow the semantic direction implied by the textual edit.

DiffusionCLIP additionally incorporates identity-preserving losses. First, a pixel-level reconstruction loss:

$$\mathcal{L}_1 = \|\hat{x} - x_{\text{ref}}\|_1,$$

which helps preserve low-level appearance when appropriate. Second, when editing human faces, they employ a *face identity loss* $\mathcal{L}_{\text{face}}$ based on ArcFace embeddings (Deng et al., 2022), enforcing consistency in identity-specific features:

$$\mathcal{L}_{\text{face}} = 1 - \cos\big(f(\hat{x}), f(x_{\text{ref}})\big),$$

where $f(\cdot)$ denotes the ArcFace and $\cos(\cdot,\cdot)$ denotes the cosine similarity.

The necessity of these identity losses depends on the editing task: for subtle attribute changes (e.g., expression, hair color), preserving human identity is crucial, while large semantic shifts (e.g., species change, stylization) may not require such constraints.

The full optimization objective is therefore:

$$\mathcal{L} = \lambda_{\text{dir}}\,\mathcal{L}_{\text{dir}} + \lambda_1\,\mathcal{L}_1 + \lambda_{\text{face}}\,\mathcal{L}_{\text{face}}, \quad \lambda_1, \lambda_{\text{face}} \geq 0.$$

Depending on the method variant, the optimization can be:

- **Guided sampling (no fine-tuning):** at each reverse step, the latent is perturbed via gradient updates to reduce $\mathcal{L}$, after which diffusion proceeds.
- **Fine-tuning the diffusion model:** the latent path is fixed, and the diffusion model parameters θ are updated to minimize $\mathcal{L}$ over a set of inverted latents. This yields a model that inherently produces images aligned with the target concept.

In our simplified implementation, we follow the fine-tuning approach with a lightweight UNet and periodic checkpointing throughout optimization.

2.4. Advantages and Tradeoffs

DiffusionCLIP enables complex, language-driven edits (e.g. style transfer, attribute modification, concept injection) without requiring labeled data or training from scratch. Because CLIP encodes high-level semantics, the method generalizes across domains (e.g. portraits, landscapes, objects). However, the approach involves trade-offs:

- **Computational cost:** decoding + encoding through CLIP at every denoising step introduces a significant overhead.
- **Optimization instability:** excessive guidance can degrade image quality or distort identity; balancing directional vs identity/perceptual losses is delicate.

- **Limited resolution/generalization:** with a small UNet and coarse latents, fine details may be lost; fine-tuning on a limited dataset may overfit.

2.5. Our Implementation Strategy

We implement DiffusionCLIP from scratch with a simplified architecture: a compact UNet backbone, standard DDIM sampling and inversion, and CLIP guidance for fine-tuning. Our goal is not to replicate all the features and refinements of the original work, but to provide a working minimal prototype that validates the methodology and allows experimentation with prompt-driven edits under constrained computational resources.

In what follows we describe the architecture, the training of the base diffusion model, the fine-tuning procedure with CLIP guidance, and the experiments we conducted.

3. Model Architecture

We adopt a lightweight U-Net backbone tailored for low-resolution diffusion modeling. The model predicts the noise term $\varepsilon_\theta(x_t, t)$ at arbitrary diffusion timestep (t), and is conditioned on time via sinusoidal embeddings injected into all residual blocks.

3.1. Sinusoidal Time Embedding

Each timestep $t \in 0, \ldots, T$ is mapped into a continuous embedding using a transformer-style sinusoidal encoding. Given embedding dimension d, we compute $\gamma(t) = \big[\sin(t \cdot \omega_1), \ldots, \sin(t \cdot \omega_{d/2}), \cos(t \cdot \omega_1), \ldots, \cos(t \cdot \omega_{d/2})\big]$,

where frequencies ω_k are logarithmically spaced. To improve expressiveness, the embedding passes through a two-layer MLP: $e_t = \mathrm{MLP}(\gamma(t))$.

This embedding $e_t \in \mathbb{R}^d$ conditions every block in the network.

3.2. Time-Conditioned Residual Blocks

The core building block is a residual unit that injects the time embedding into the feature activations. Given input feature maps x and time embedding e_t, the block applies two 3×3 convolutions with GroupNorm and ReLU activations. Time conditioning is introduced by projecting e_t to the channel dimension and broadcasting it spatially:

$$
\begin{aligned}
h &= \mathrm{ConvGNReLU}(x), \\
h &\leftarrow h + W_t e_t,
\end{aligned}
$$

followed by a second convolutional layer. A skip connection matches input and output channels either via identity or a 1×1 projection. These blocks make temporal information available at all spatial resolutions, analogous to those used in standard DDPM/Score UNets but in a significantly smaller configuration.

3.3. Tiny-U-Net Backbone

The overall architecture follows the encoder–decoder U-Net design with symmetric skip connections. Let C denote the base number of channels (set to 64).

3.3.1. Downsampling path

- Block 1: $3 \to C$ channels, producing feature map h_1.
- Block 2: Max pooling ($2\times$ downsampling) followed by $C \to 2C$, producing h_2.
- Block 3: Another Max pooling ($2\times$ downsampling) followed by $2C \to 4C$, producing h_3.

3.3.2. Bottleneck

A single residual block processes the lowest-resolution representation: $4C \to 4C$.

3.3.3. Upsampling path

Upsampling is performed using transposed convolutions, concatenated with skip features from the encoder:

1. $4C \to 2C$ followed by a residual block operating on concatenated $[u_1, h_2]$.
2. $2C \to C$ followed by a residual block on $[u_2, h_1]$.

3.3.4. Output layer

A final 1×1 convolution maps the resulting feature map back to RGB space: $C \to 3$.

The model outputs the predicted noise $\hat{\varepsilon}_\theta(x_t, t)$, as required by the denoising objective.

3.4. Design Rationale

This "TinyUNet" trades representational capacity for speed and stability, making it suitable for small-resolution datasets such as CelebA-HQ-32. Despite having fewer parameters than conventional DDPM UNets, it retains:

- multiscale feature hierarchies,
- timestep conditioning at all depths,
- symmetric skip connections essential for spatial fidelity.

This architecture is sufficient to support both standard DDPM training and CLIP-guided finetuning, as explored in later sections. We chose this structure due to computational constraints.

4. Diffusion Training

We train the model following the standard denoising diffusion probabilistic model (DDPM) objective. Let $x_0 \sim p_{\text{data}}$ be a real image, and let $\{\beta_t\}_{t=0}^{T-1}$ be a variance schedule with

$$\alpha_t = 1 - \beta_t, \qquad \bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s.$$

4.1. Forward (noising) process

The forward diffusion defines a Markov chain that gradually corrupts an image by adding Gaussian noise:

$$q(x_t \mid x_{t-1}) = \mathcal{N}(\sqrt{\alpha_t} x_{t-1},\, \beta_t \mathbf{I})\,.$$

A key DDPM property is that one can sample x_t directly from x_0 in closed form:

$$q(x_t \mid x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}\, x_0,\, (1 - \bar{\alpha}_t)\mathbf{I}\big)\,.$$

Thus, for a sampled noise $\varepsilon \sim \mathcal{N}(0, I)$,

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon.$$

This is exactly what the implementation computes inside `q_sample`, where the scalars $\sqrt{\bar{\alpha}_t}$ and $\sqrt{1 - \bar{\alpha}_t}$ are gathered per-sample.

4.2. Training objective

The model is trained to predict the noise added during the forward process. Given the UNet prediction $\varepsilon_\theta(x_t, t)$, the loss is the simple MSE term

$$\mathcal{L} = \mathbb{E}_{x_0,\, t,\, \varepsilon} \left[\|\varepsilon - \varepsilon_\theta(x_t, t)\|_2^2 \right],$$

where the timestep t is sampled uniformly from $\{0, \ldots, T - 1\}$.

Operationally, training proceeds as follows:

1. Sample a minibatch of real images x_0.
2. Sample timesteps $t \sim$ Uniform$\{0, \ldots, T - 1\}$.
3. Sample noise $\varepsilon \sim \mathcal{N}(0, I)$.
4. Construct x_t using the closed-form noising equation.
5. Predict noise $\hat{\varepsilon} = \varepsilon_\theta(x_t, t)$.
6. Minimize MSE$(\hat{\varepsilon},\, \varepsilon)$ via Adam.

4.3. Sampling during training

For monitoring, we generate samples using a deterministic DDIM-style update. Given a predicted noise term $\varepsilon_\theta(x_t, t)$, an estimate of the clean image is

$$\hat{x}_0 = \frac{x_t - \sqrt{1 - \bar{\alpha}_t}\, \varepsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}.$$

Then one can compute the next step x_{t-1} using the closed-form DDIM update,

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\, \hat{x}_0 \;+\; \sqrt{1 - \bar{\alpha}_{t-1}}\, \varepsilon_\theta(x_t, t).$$

This deterministic sampler provides fast qualitative feedback and is implemented at the end of each training epoch.

4.4. DDIM Sampling

Denoising Diffusion Implicit Models (DDIM) provide a deterministic variant of the reverse diffusion process, allowing efficient sampling without injecting additional noise. Let $\epsilon_\theta(x_t, t)$ denote the noise prediction network (the UNet used in our implementation) and let $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ denote the cumulative product of the forward-process noise schedule.

Given a latent variable x_t at timestep t, we compute the corresponding clean estimate $\hat{x}_0$ using the closed-form relationship of the forward diffusion process:

$$\hat{x}_0 = \frac{x_t - \sqrt{1 - \bar{\alpha}_t}\, \epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}. \tag{1}$$

Using this estimate, the next latent x_{t-1} is produced as

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\, \hat{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1}}\, \epsilon_\theta(x_t, t). \tag{2}$$

This is precisely the update rule used in our sampler, matching the original DDIM formulation when the stochasticity term is removed. The full sampling trajectory $x_T \to x_0$ is obtained by iterating (1)–(2) for $t = T, T-1, \ldots, 1$. Because no random noise is added at any step, the entire sampling path is deterministic and considerably faster than the ancestral DDPM sampler.

DDIM Inversion. We also use the same deterministic rule in the *forward* direction to map a clean RGB image x_0 into a noisy latent representation x_r at an intermediate timestep r. In our implementation this is done by starting at x_0 and applying the forward-form DDIM update

$$x_{t+1} = \sqrt{\bar{\alpha}_{t+1}}\, x_0 + \sqrt{1 - \bar{\alpha}_{t+1}}\, \epsilon_\theta(x_t, t), \tag{3}$$

for $t = 0, 1, \ldots, r - 1$. The resulting latent x_r serves as the input state for fine-tuning. Using a deterministic inversion ensures that each training iteration begins from a stable and reproducible latent code, which is essential for obtaining consistent gradients across optimization steps.

4.5. CLIP-Guided Fine-Tuning

To steer the pretrained diffusion model toward a desired semantic concept, we adopt the directional CLIP guidance strategy introduced by DiffusionCLIP. For each real image in the training set, we first compute its DDIM-inverted latent x_r at a fixed timestep r. Starting from this latent, the model performs a reverse DDIM pass. At each denoising step, the estimate $\hat{x}_0^{(t)}$ is computed using the same formula as in Eq. (1).

Let $e_{\text{img}}(x)$ denote the CLIP image encoder applied to an RGB image x, and let $e_{\text{text}}(p)$ denote the CLIP text encoder applied to a prompt p. As in our implementation, we precompute normalized text-direction vectors from a source prompt p_{src} to a target prompt p_{tgt}:

$$d_{\text{text}} = \frac{e_{\text{text}}(p_{\text{tgt}}) - e_{\text{text}}(p_{\text{src}})}{\|e_{\text{text}}(p_{\text{tgt}}) - e_{\text{text}}(p_{\text{src}})\|}. \tag{4}$$

During fine-tuning, the model's predicted clean estimate $\hat{x}_0^{(t)}$ is decoded (in our case through the diffusion decoder implicitly reconstructed by the UNet prediction) and passed through the CLIP image encoder to obtain $e_{\text{img}}(\hat{x}_0^{(t)})$. We then compute the corresponding image-direction vector

$$d_{\text{img}} = \frac{e_{\text{img}}(\hat{x}_0^{(t)}) - e_{\text{img}}(x_{\text{ref}})}{\left\|e_{\text{img}}(\hat{x}_0^{(t)}) - e_{\text{img}}(x_{\text{ref}})\right\|}, \tag{5}$$

where x_{ref} is the original clean input image. This vector captures how the current prediction is semantically deviating from the reference image.

The main optimization objective is the directional CLIP loss, which encourages the semantic displacement of the generated image to align with the intended text direction:

$$\mathcal{L}_{\text{dir}} = 1 - \cos(d_{\text{img}},\, d_{\text{text}})\,. \tag{6}$$

To prevent the model from drifting too far from the input identity or structure of the sample, we also employ an identity-preserving loss based on the pixel difference between $\hat{x}_0^{(t)}$ and the original image:

$$\mathcal{L}_{\text{id}} = \left\|\hat{x}_0^{(t)} - x_{\text{ref}}\right\|_1. \tag{7}$$

The final training objective is a weighted sum of these two losses:

$$\mathcal{L} = \lambda_{\text{dir}}\,\mathcal{L}_{\text{dir}} + \lambda_{\text{id}}\,\mathcal{L}_{\text{id}}. \tag{8}$$

Note that we did not use the face identity loss for simplicity. Gradients are applied directly at each reverse denoising step. After updating the UNet parameters, the latent state is updated using the deterministic DDIM rule in Eq. (2), and the process continues toward the next timestep. Because the method does not require backpropagation through the entire unrolled trajectory, it remains memory-efficient while allowing fine-grained, step-wise adjustment of the model parameters.

5. Experimental Setup

To evaluate the simplified DiffusionCLIP implementation, we conducted all experiments on a local workstation equipped with an Intel i7–14700 CPU and an NVIDIA GeForce RTX 4060 GPU running Arch Linux. All models were trained using PyTorch with mixed-precision disabled, and no additional pretrained diffusion weights were used—the diffusion UNet was trained from scratch on CelebA.

5.1. Dataset

We used the CelebA dataset (Liu et al., 2015), consisting of over 200,000 aligned celebrity face images. In all experiments:

- Images were center-cropped to remove background.
- All samples were resized to 128×128 resolution.
- Pixel intensities were scaled to $[-1, 1]$.
- Only the RGB channels were used.

The dataset was split into 90% training and 10% validation, with shuffling enabled at each epoch.

5.2. Model and Training Configuration

The UNet architecture follows a minimal diffusion model design adapted from standard DDPM baselines, reduced in depth to accommodate limited computational resources. CLIP ViT-B/32 (Radford et al., 2021) was used as the frozen text–image encoder for computing directional losses.

Table 1 summarizes the core hyperparameters used in all experiments. These values were chosen to balance training time with the stability required for DDPM training from scratch.

5.3. Procedure

Our experimental pipeline proceeded in three stages:

1. **Training the DDPM model.** The UNet was trained using the standard noise-prediction objective for 200 diffusion steps and 10 epochs with a linear variance schedule. Convergence was monitored using the validation loss and image samples generated by DDIM sampling every few epochs.
2. **Latent inversion.** For each reference image, we computed a deterministic DDIM inversion up to step

Table 1: Training and sampling parameters used in all experiments.

Parameter	**Value**
Dataset	CelebA
Image resolution	128×128
Batch size (DDPM training)	30
Batch size (CLIP fine-tuning)	30
Optimizer	Adam
Learning rate (DDPM pretraining)	2×10^{-4}
Learning rate (fine-tuning)	2×10^{-3}
Diffusion steps T	200
Sampling steps (DDIM)	50
DDIM inversion step r	50
CLIP model	ViT-B/32 (frozen)
Directional loss weight λ_{dir}	1.0
Identity loss weight λ_{id}	0.3
Face identity loss	0
Number of fine-tuning epochs	5
Hardware	RTX 4060 (8GB)

$r = 50$, producing a consistent noisy latent x_r used for all fine-tuning experiments.

3. **CLIP-guided fine-tuning.** We set the target prompt “sketch” and computed directional losses at each reverse step of the DDIM trajectory. Only the UNet parameters were updated; CLIP remained frozen.

This setup allowed the entire pipeline to run within the memory limits of the RTX 4060 while producing edits that qualitatively resemble the target prompts, despite the reduced model capacity.

6. Results

During fine-tuning, we observed that the CLIP-guided objective did not decrease as expected. Throughout training, the CLIP loss plateaued around an average value of approximately 0.5, whereas the identity-preserving loss reliably reached values near 0.1. This suggests that the model maintained reasonable similarity to the original input image, but struggled to move sufficiently in the semantic direction specified by the text prompt. In practice, this indicates that the current hyperparameters—as well as the architectural capacity of the simplified U-Net—are insufficient for effective text-driven manipulation.

Representative qualitative results are presented in Figure 3, where each row contains the original input image, its corresponding latent representation at the selected DDIM timestep, and the final edited image. As expected, the latent representation resembles structured noise but still retains coarse features of the input. This behavior is consistent with DDIM inversion, but the reconstructions suggest that the U-Net used in our implementation lacks the expressive power required to accurately decode these representations. Increasing the number of parameters, depth, or attention capacity of the network would likely improve reconstruction fidelity.

The final edited outputs exhibit partial alignment with the prompt “sketch”. Although the generated images show a tendency toward simplified contours and reduced texture, the transformation is incomplete and often visually inconsistent. This indicates that the CLIP loss weight was too small relative to the identity loss, and that the number of training samples was insufficient to meaningfully guide the model toward the desired editing direction. Incorporating a pretrained U-Net—as done in the original DiffusionCLIP implementation by Kim et al. (2022)—would likely produce stronger semantic shifts while preserving image quality.

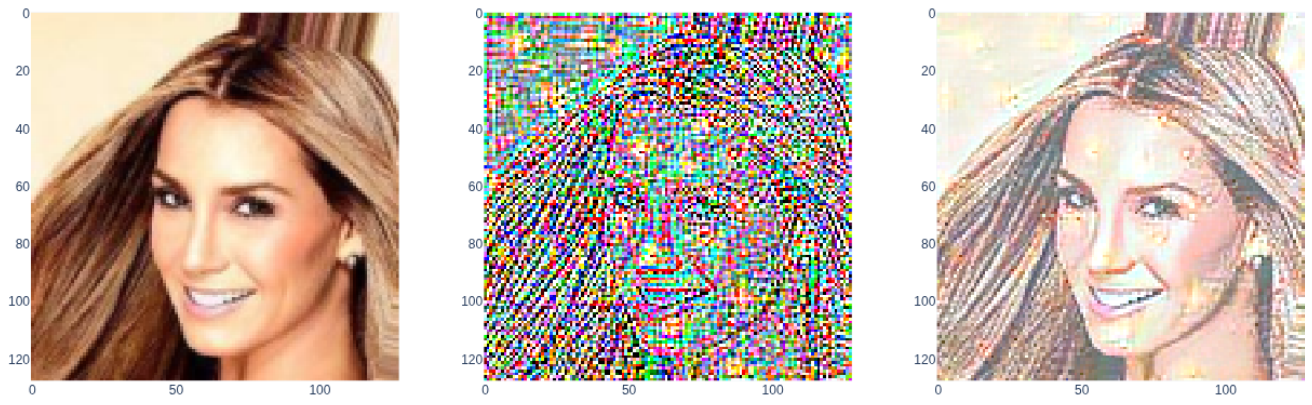
Overall, while none of the results fully achieved the intended “sketch” appearance, our findings suggest that the general pipeline remains viable even when using significantly simplified diffusion models. With additional training data, refined loss balancing, and a larger or pretrained U-Net, we expect that performance would approach the results reported in the original work.

7. Conclusion and Future Work

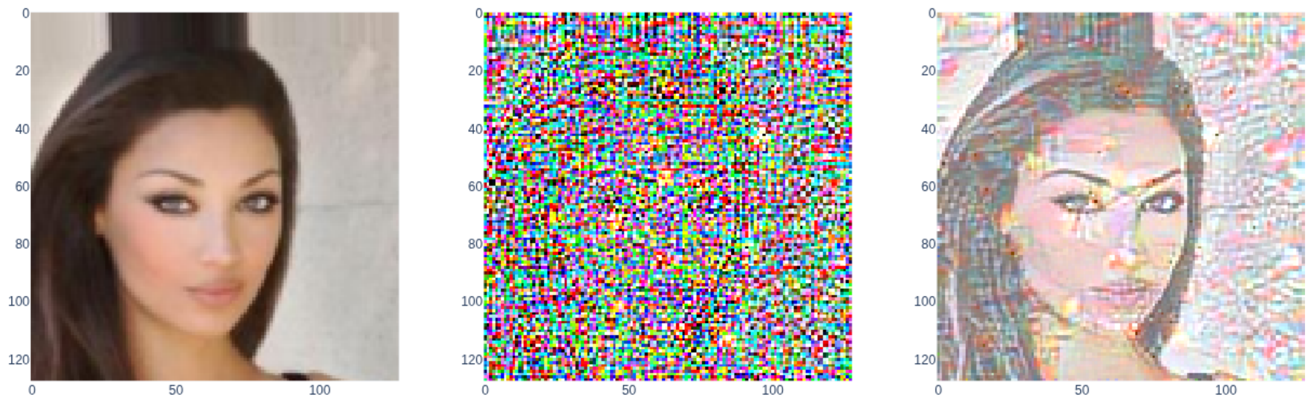
In this work, we implemented and trained a simplified version of DiffusionCLIP. Our approach replaced the large pretrained diffusion backbone used in the original method with a lightweight U-Net trained from scratch, and fine-tuned it using a pretrained CLIP model (ViT-B/32). We considered only a single editing objective in this study, focusing on guiding images toward the “sketch” concept through directional CLIP loss. Despite the significant simplifications to the diffusion model and training pipeline, the system produced edited images that partially resembled the target prompt, demonstrating that even reduced-scale architectures can capture some degree of CLIP-guided semantic manipulation.

However, the quality and consistency of the results remain significantly below those reported in the original DiffusionCLIP paper. Our experiments indicate that model capacity, dataset size, and loss weighting all play crucial roles in successful text-driven editing. In particular, the simplified U-Net struggled with accurate DDIM inversion and reconstruction, limiting the effectiveness of fine-tuning. The limited number of samples and relatively weak CLIP-loss weight further constrained the model’s ability to move toward the desired semantic direction.

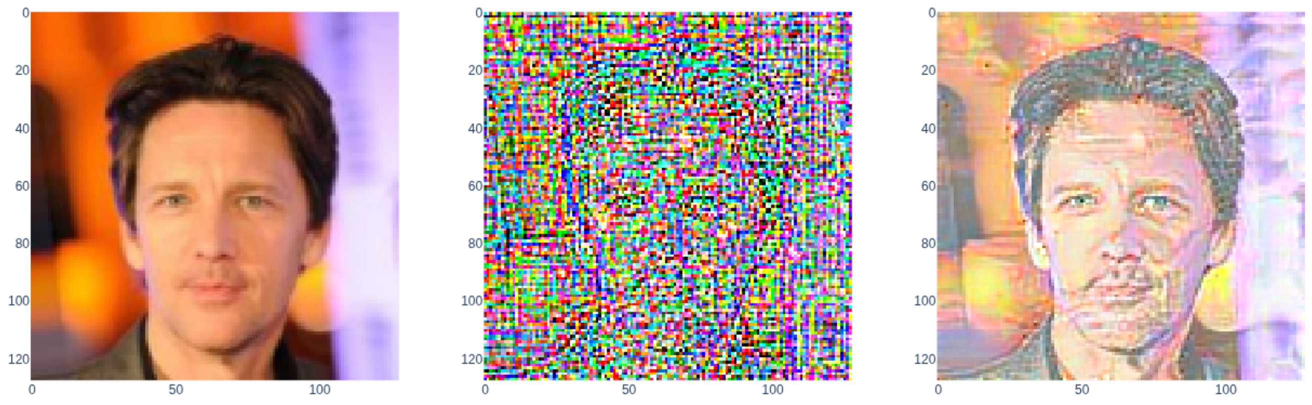
Overall, the results suggest that CLIP-guided diffusion editing is still feasible in simplified or resource-constrained



(a) First Image



(b) Second Image



(c) Third Image

Figure 3: Original, latent and edited images (stacked horizontally) for three different cases.

settings, but achieving high-quality transformations requires more expressive architectures and more robust training strategies. Incorporating a pretrained diffusion backbone would likely provide a stronger starting point and yield performance closer to that of full DiffusionCLIP models. Another promising direction is expanding the training dataset and experimenting with selectively training only the U-Net and relying on the original fine-tuning structure.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Deng, J., Guo, J., Yang, J., Xue, N., Kotsia, I., and Zafeiriou, S. Arcface: Additive angular margin loss for deep face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):5962–5979, October 2022. ISSN 1939-3539. doi: 10.1109/tpami.2021.3087709. URL http://dx.doi.org/10.1109/TPAMI.2021.3087709.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Kim, G., Kwon, T., and Ye, J. C. Diffusionclip: Text-guided diffusion models for robust image manipulation, 2022. URL https://arxiv.org/abs/2110.02711.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.