Imperial College London

MEng Individual Project Final Report

Imperial College London

Department of Electrical and Electronic Engineering

Paint from Noise: Mask-free Image inpainting with diffusion models

Author:
Adam Rehman

CID:
01720256

Supervisor:
Dr. Chen Qin

Second Marker:
Professor Jeremy Pitt

November 21, 2025

Plagiarism statement

I affirm that I have submitted, or will submit, an electronic copy of my final year project report to the provided EEE link.

I affirm that I have submitted, or will submit, an identical electronic copy of my final year project to the provided Blackboard module for Plagiarism checking.

I affirm that I have provided explicit references for all the material in my Final Report that is not authored by me, but is represented as my own work.

Contents

Abstract

Image inpainting is a crucial task in computer vision that aims to restore missing or damaged regions within an image. One of the key challenges in this domain is accurately identifying the contours encapsulating said regions, commonly referred to as mask generation. This report presents an advanced approach for mask generation specifically designed for image inpainting through diffusion models, with a focus on the application of this technique for seamless image restoration.

Acknowledgments

In this moment of accomplishment, I wholeheartedly extend my gratitude towards several individuals who have played instrumental roles throughout my journey. Firstly, my sincere appreciation goes to my teachers, family, and friends, whose steadfast support has been a constant source of strength for me.

A special note of thanks is due to my supervisor, Dr. Qin Chen, and the PhD student Yuyang Xue. Their time, expert insights, and unwavering guidance have been crucial to the progress and success of this project. Their commitment has shaped not just this work, but also my approach towards academic research.

Lastly, I would like to acknowledge the participants who took the time to evaluate the generated content. Your involvement was key to the integrity and quality of this work. Thank you for your valuable contributions.

Abbreviations

- SOTA - State of the art
- VAE - Variational Auto-Encoder
- GAN - Generative Adversarial Networks
- CNN - Convolutional Neural Network
- ELBO - Evidence Lower Bound
- KL Divergence - Kullback–Leibler Divergence
- DDPM - Denoising diffusion probabilistic model
- CLIP - Contrastive Language-Image Pretraining (refers to a zero-shot Image-Text classification model)
- GPU - Graphics Processing Unit

Chapter 1

Introduction

1.1 Problem Statement

Image inpainting refers to the process of identifying a specific region in an image - it could be damaged or simply contain unwanted content - and replacing it with something else. The restored region should ideally blend with the rest of the image, yielding a plausible output. This necessitates the deployment of precise and efficient mask generation techniques.

Deep learning models such as Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and more recently, diffusion models have become standard tools in digital inpainting. Specifically, diffusion models have recently outperformed GANs in the image inpainting task (**dhariwal2021diffusion**).

The term "mask-free inpainting" is the subject of this paper and refers to inpainting tasks where a user is not required to provide a mask along with the image. Specifically, it pertains to text-prompt-based mask generation methods where the user merely has to supply a string specifying what needs to be replaced, and optionally, what it should be replaced with.

1.2 Proposed Method

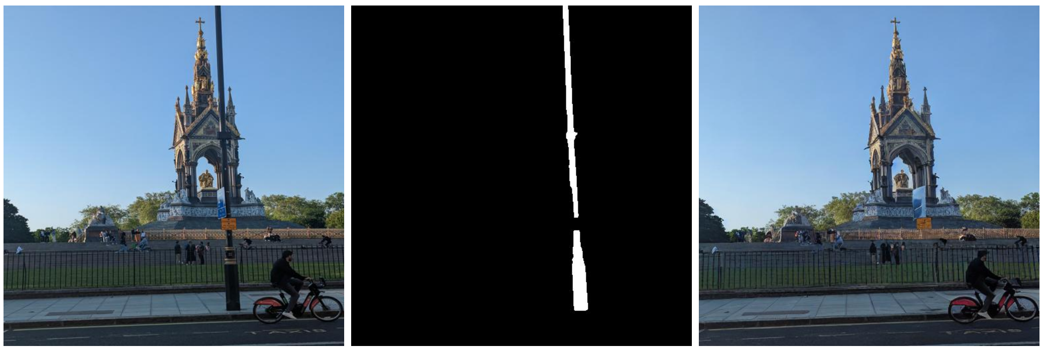


Figure 1.1: From left to right: Original image, segCLIP generated mask with the prompt "pole", inpainted image using stable diffusion

The proposed method, dubbed "segCLIP", harnesses recent advancements in deep learning, particularly in image segmentation models, to create high-quality masks from a text prompt. This extends the functionality of Meta's Segment Anything model 2.5.3 to allow for segmentation from a text prompt - a feature not currently implemented. In addition, we evaluate and compare other state-of-the-art (SOTA) text-based segmentation methods in terms of image quality, prompt adherence, and efficiency.

Considering time and cost constraints, several pre-trained diffusion models 2.3 are employed to assess the mask generation methods. While these guided diffusion models are used to inpaint the region, they also hold potential for generating high-quality masks for segmentation (**couairon2022diffedit**).

Our experiments illustrate segCLIP's effectiveness using a dataset of photos from South Kensington and Imperial College London 4.1. The inpainting results exhibit impeccable restoration, with the reconstructed areas blending perfectly with the surrounding context to maintain the visual consistency of the images.

In summary, this report proposes an innovative mask segmentation method tailored for image inpainting, leveraging SOTA image segmentation and diffusion models. The proposed method exhibits remarkable accuracy and efficiency in recognizing desired objects from text prompts, offering potential for enhanced image inpainting models that don't require manual mask drawing.

1.3 Wider Applications

The outcomes of this research can potentially aid numerous applications, including photo editing and object removal, where precise and seamless inpainting is crucial. An excellent example would be image editing on smartphones, many of which are now equipped with tensor cores to run such models. Users can simply type what they wish to find and replace.

1.4 Report Structure

This report will proceed in the following order: Literature Review, Requirements Capture, Design and Implementation, Results and Evaluation, and Conclusion.

Chapter 2

Literature Review

The literature review is structured as follows:

- A short introduction of convolutional neural networks
- An overview of generative models used to perform the inpainting task
- A deeper insight into diffusion models
- A comprehensive review of the current SOTA text-based segmentation methods required for the inpainting tasks

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep, feed-forward artificial neural networks widely employed in visual imagery analysis. Inspired by the structure of the visual nervous system, CNNs autonomously and adaptively learn patterns directly from data (**fukushima1980neocognitron**, **lecun2015deep**).

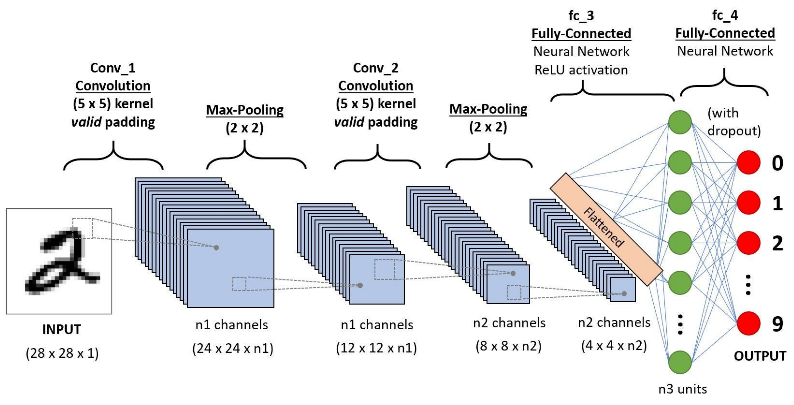


Figure 2.1: Example of a convolutional neural network

CNNs consist of an input layer, an output layer, and multiple hidden layers. These hidden layers typically include convolutional layers, nonlinear layers such as ReLU, pooling layers, fully connected layers, and normalization layers (**lecun2015deep**).

Convolutional layers perform convolutions on the input data, passing the results to the subsequent layer. This convolution process simulates the response of individual neurons to visual stimuli. ReLU layers introduce non-linearity to the network by applying the non-saturating activation function $f(x) = \max(0, x)$. Pooling layers reduce data dimensions by combining outputs of neuron clusters from one layer into a single neuron in the next layer. Fully connected layers establish connections between every neuron in one layer and every neuron in another layer (**lecun2015deep**).

By employing multiple layers of these convolutional operations, CNNs can effectively recognize intricate patterns in data. Consequently, they have demonstrated their utility in various applications, including image and video recognition, recommendation systems, and natural language processing.

2.1.1 UNet

The U-Net (**ronneberger2015u**) is a type of CNN that was developed for biomedical image segmentation at the Computer Science Department of the University of Freiburg, Germany. The network is named U-Net because of its U-shaped architecture.

U-Net builds upon the classic CNN architecture and is designed to work with very few training images, making it a suitable model for medical imaging tasks (**ronneberger2015u**). It was originally intended to segment biomedical images, but it has been found to be effective for a variety of other image segmentation problems, such as denoising an image.

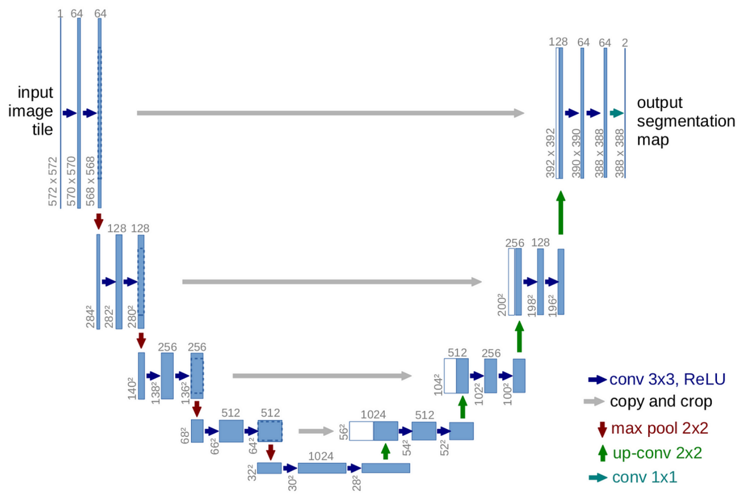
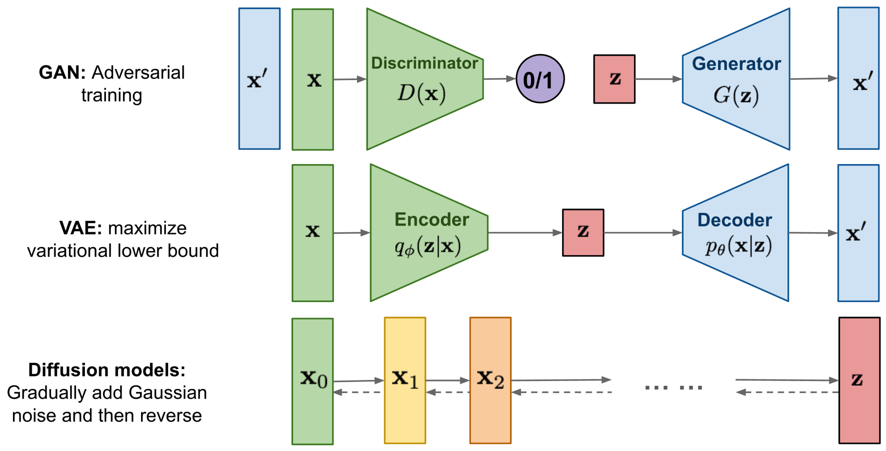


Figure 2.2: Unet architecture

U-Net's architecture is distinguished by its symmetry. It begins with repeated 3x3 convolutions (unpadded) followed by ReLU and 2x2 max pooling for downsampling. The network then upsamples using up-convolution layers and concatenation with cropped feature maps from the downsampling path, followed by 3x3 convolutions with ReLU. This expansive path enables the model to capture ample contextual information, benefiting tasks like semantic segmentation that require a holistic understanding of the image for accurate pixel labeling (**ronneberger2015u**). The U-Net architecture essentially involves a contracting (downsampling) path to capture context and a symmetric expanding (upsampling) path that allows precise localization, making it a very effective architecture for the task of image segmentation.

2.2 Generative models

In this section we introduce two classes of generative models which have had historical success in image generation, and consequently image inpainting.



GAN: Adversarial training
$\mathbf{x}'$
$\mathbf{x}$
Discriminator $D(\mathbf{x})$
0/1
$\mathbf{z}$
Generator $G(\mathbf{z})$
$\mathbf{x}'$
VAE: maximize variational lower bound
$\mathbf{x}$
Encoder $q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
Decoder $p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
Diffusion models: Gradually add Gaussian noise and then reverse
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_2$
... ...
$\mathbf{z}$

Figure 2.3: Overview of different types of generative models [**weng2021diffusion**]

2.2.1 GAN

Generative adversarial networks are a class of deep-learning based generative models which have seen recent inpainting success in models such as AttnGAN (**Xu_2018_CVPR**), DMGAN (**zhu2019dm**), and DF-GAN (**tao2020df**). The GAN model trains two separate sub-models, a discriminator and a generator. As such, the discriminator is trained on the dataset to decide whether a sample is "real" (in the data distribution) or "fake". Concurrently the generator is trained to generate samples that the discriminator believes is real.

In the paper **goodfellow2020generative**, the GAN objective is modeled using a value function, a two player minimax game:

$$x \sim p_r(x)\text{: real } \boldsymbol{x}$$
$$x \sim p_g(x)\text{: generated } \boldsymbol{x}$$
$$z \sim p_z(z)\text{: latent variable } \boldsymbol{z}$$

$$\begin{aligned}\min_G \max_D L(D,G) &= E_{x\sim p_r(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \\ &= E_{x\sim p_r(x)}[\log D(x)] + E_{x\sim p_g(x)}[\log(1 - D(x)]\end{aligned}$$

goodfellow2020generative

The first term $E_{x\sim p_r(x)}[\log D(x)]$ denotes the expectation that a point from the real data $p_r(x)$ is accurately identified as such by the discriminator D. Similarly, the second term $E_{z\sim p_z(z)}[\log(1 - D(G(z)))]$ denotes the expectation that a sample $p_g(x)$ created from the generator G is rejected by the discriminator.

Compared to DDPM models, sampling from a GAN only requires one forward pass and so takes considerably less time to generate a sample. For example, it takes around 20 hours to sample 50k images of size 32 Œ 32 from a DDPM, but less than a minute to do so from a GAN on an Nvidia 2080 Ti GPU (**song2020denoising**).

DDIM models (2.3.2) and sampling techniques such as a strided sampling schedule every $[\frac{T}{S}]$ steps, help to alleviate this gap in sampling the outputs of the respective models (**weng2021diffusion**).

A main problem with GANs is that training can be unstable. Concurrently updating both models means that convergence cannot be guaranteed. Also, the generator can be prone to producing very similar outputs that can pass the discriminator but do not accurately represent the variety of the complex real data distribution (**weng2017gan**).

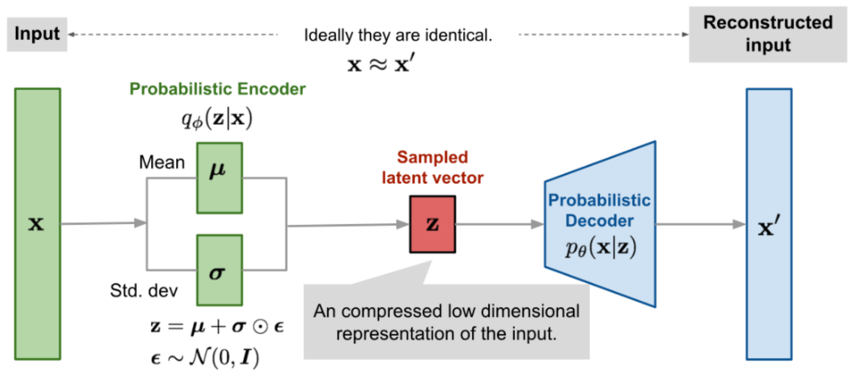

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
Mean
$\boldsymbol{\mu}$
Std. dev
$\boldsymbol{\sigma}$
$\mathbf{x}$
Sampled latent vector
$\mathbf{z}$
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
An compressed low dimensional representation of the input.

Figure 2.4: Variational autoencoder architecture (**weng2018VAE**)

2.2.2 VAE

The variational autoencoder is a generative model that has had widespread success in image generation, and is also used in various generative models to transform data into and out of latent spaces. It consists of two sequential parts: an encoder $q_\phi(\mathbf{z}|\mathbf{x})$ and a decoder $p_\theta(\mathbf{x}|\mathbf{z})$. Unlike the autoencoder, the VAE maps data into a latent gaussian probability distribution $q_\phi(\mathbf{z}|\mathbf{x})$ where the latent variable $\mathbf{z}$ can be sampled from, rather than straight to a vector $\mathbf{z}$ (as an autoencoder would). By sampling from the learned probability distribution and decoding, an image from the learned data space can be sampled. **doersch2016tutorial**.

The optimal parameters θ maximise the probability of the model generating real samples, i.e:

$$\theta^* = \arg\max_\theta \prod_{i=1}^{n} p_\theta(\mathbf{x}^{(i)})$$

The probability of generating a real sample is shown considering the forward pass through the decoder from $\mathbf{z}$:

$$p_\theta(\mathbf{x}^{(i)}) = \int p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$$

Evaluating $p_\theta(\mathbf{x}^{(i)})$ is very computationally expensive as we must integrate with respect to the entire distribution of latent variables in z. Considering that the posterior of the decoder $p_\theta(\mathbf{z}|\mathbf{x})$ should be close to the encoder distribution $q_\theta(\mathbf{z}|\mathbf{x})$, a loss function can be found by considering the KL divergence of the respective distributions.

$$KL(P||Q) = \sum_x P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

The KL-divergence between two distributions measures how "similar" they are to each other, considering the case for the VAE we have:

$$D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) = \log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) - E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})$$

Rearranging terms we get

$$\log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) = E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$$

where LHS is the objective of the VAE, which minimises the KL-divergence, whilst also maximizing $\log p_\theta(\mathbf{x})$, the probability of the model generating a real sample. This is called the Variational Lower Bound, also referred to as Evidence Lower Bound (ELBO).

So for our final loss function we have:

$$\begin{aligned} L_{\text{VAE}}(\theta,\phi) &= -\log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) \\ &= -E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \\ \implies \qquad \theta^*,\phi^* &= \arg\min_{\theta,\phi} L_{\text{VAE}} \end{aligned}$$

Derivation from **weng2018VAE**.

2.3 Diffusion Models

The application of nonequilibirium thermodynamic equations to the field of generative models was proposed in **sohl2015deep**, introducing the diffusion model, which claimed to be both "tractable and flexible". These diffusion models work by using a Markov chain to sequentially add noise to an image until it is approximately pure noise. Then the model learns to reverse each noise step in order to generate the image from noise.

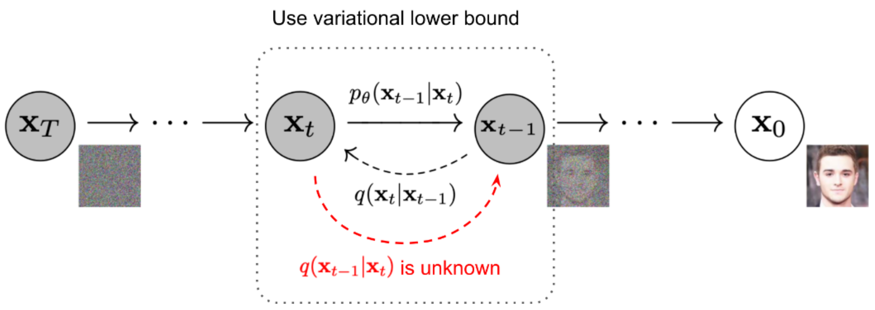

Use variational lower bound
$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is unknown
$\mathbf{x}_0$

Figure 2.5: The Markov chain of the diffusion process of generating a sample by slowly adding/removing noise [**weng2021diffusion**]

2.3.1 Denoising Diffusion Probabalistic Models

The forward diffusion process procedurally adds noise to the image:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

Where $\beta_{[0:T]}$ is a predefined variance schedule, **sohl2015deep** defined it as a linearly increasing variance schedule, however a cosine variance schedule (**song2020denoising**) has been shown to be an improvement.

Also using the substitutions $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, the noised image at step t can be directly calculated instead of iteratively adding noise to the image.

This leaves us with the forward process

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

to calculate the noised image at each step, $\mathbf{x}_t$.

To generate a sample, the posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ must simply be calculated T times from pure noise ($\approx \mathbf{x}_T$) to get back to the data distribution:

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

The posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is well approximated by a diagonal gaussian for small $1-\alpha_t$, so the idea is to use a model $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to approximate the posterior of the noising sequence $q(\mathbf{x}_{t-1}|\mathbf{x}_t) \approx p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$. The variance $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$ is usually replaced with the cumulative beta schedule $\tilde{\beta}$ discussed earlier due to instability in training. Therefore the inference process is as shown:

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

Variational Lower Bound loss

Similarly to the VAE (2.2.2), the loss function of the model is optimised by considering the ELBO (Evidence Lower Bound) which minimizes the distance between the posterior of the noising step $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and the learned model $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, whilst also increasing the probability of a real sample being generated.

The reverse diffusion process: $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})$ is tractable when conditioned on $\mathbf{x}_0$.

Therefore the ELBO can be proven to be:

$$\begin{aligned} L_{\text{VLB}} &= E_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \\ &= E_q[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}] \end{aligned}$$

Score matching with Langevin Dynamics

Stochastic gradient Langevin Dynamics can produce samples from a probability density $p(\mathbf{x})$ using only the score function $\nabla_\mathbf{x} p(\mathbf{x})$. With a fixed step size $\delta > 0$, and an initial value $\mathbf{x}_0$ from the dataset, when $\delta \hookrightarrow 0$ and $T \hookrightarrow \infty$, $\hat{\mathbf{x}}_T$ becomes an exact sample from $p(\mathbf{x})$ (**song2019generative**).

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \frac{\delta}{2}\nabla_\mathbf{x} \log p(\mathbf{x}_{t-1}) + \sqrt{\delta}\boldsymbol{\epsilon}_t, \quad \text{where } \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Using score matching, we can directly train a score network $s_\theta(x)$ to estimate ∇x log $p_{data}(x)$, without training a model to estimate $p_{data}(x)$ first. Thus we can approximately obtain samples using Langevin Dynamics.

Simple objective

ho2020denoising showed that a score-based approach is proportional to calculating the mean-squared error between the noise prediction and the actual noise added throughout each step in the Markov chain (**nichol2021glide**). The simple loss is shown below:

$$\begin{aligned} L_t^{\text{simple}} &= E_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\Big[||\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)||^2\Big] \\ &= E_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\Big[||\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)||^2\Big] \end{aligned}$$

ho2020denoising

Noise predictor architecture

To complete the process, the next step is to select a model that excels in denoising or score matching. A widely preferred option is the UNet architecture with attention. This choice is made because it effectively incorporates temporal and caption/conditional awareness into the model. It has been widely recognized as one of the top-performing architectures for image segmentation, as mentioned in 2.1.1.

2.3.2 Denoising Diffusion Implicit Models

DDIMs question the reasoning behind using Markov chains to perform the generative phase, and rather introduces non-Markovian diffusions steps that can lead to much faster sampling (10x to 50x faster).

> In particular, we are able to use non-Markovian diffusion processes which lead to short generative Markov chains (Section 4.2) that can be simulated in a small number of steps. This can massively increase sample efficiency only at a minor cost in sample quality.
>
> **song2020denoising**

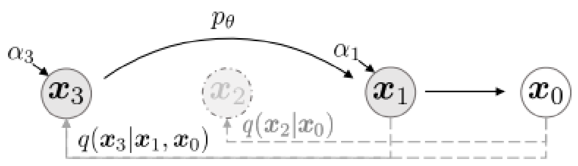


Figure 2: Graphical model for accelerated generation, where $\tau = [1, 3]$.

Instead of predicting $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$, the inference stage is reparameterised to estimate $p(\mathbf{x}_{\tau_{i-1}}|\mathbf{x}_{\tau_i})$, where τ is a subsequence of the steps $\mathbf{t}$ in the Markov chain.

Compared to DDPM, DDIM can:

- Generate higher-quality samples using a much fewer number of steps.
- Have the consistency property since the generative process is deterministic, meaning that multiple samples conditioned on the same latent variable should have similar high-level features.
- Because of the consistency, DDIM can do semantically meaningful interpolation in the latent variable.

(**weng2021diffusion**)

Sidenote Recently, **song2023consistency** introduced the consistency model, directly mapping noise to data, allowing for single step inference whilst maintaining good sample quality. This is still relatively new and so will not be explored further in this paper.

2.3.3 Latent Diffusion Models

Latent diffusion models propose a significant alteration to traditional diffusion models by suggesting that the diffusion process can be executed in the latent space, rather than directly in the pixel space (**vahdat2021score**). This shift from pixel space to latent space operation introduces multiple advantages that enhance the model's efficiency and performance.

At the outset, the data must be transformed, or "compressed," into a latent space representation. This is commonly achieved by utilizing an autoencoder or variational autoencoder model 2.2.2. These models encode the high-dimensional, pixel-level input data into a lower-dimensional latent space representation. The reduction in dimensionality brings about two key benefits (**rombach2022high**).

Firstly, the reduction in the data's complexity streamlines the subsequent steps of the diffusion model, notably the sampling and training phases. These processes generally become less computationally demanding in the latent space, given its lower dimensionality compared to the pixel space. This translates into enhanced computational efficiency, allowing the model to operate faster and more effectively.

Secondly, the latent space is generally more semantically meaningful compared to the pixel-level representations. The process of encoding into the latent space seeks to capture the significant, high-level features of the image that carry semantic information. Consequently, the latent diffusion model, operating in this space, is more aligned with these semantic features rather than pixel-level perceptual attributes. This alignment can potentially improve the model's performance in tasks that require semantic understanding, such as object segmentation and inpainting, thus broadening its applicability in image editing and related tasks.

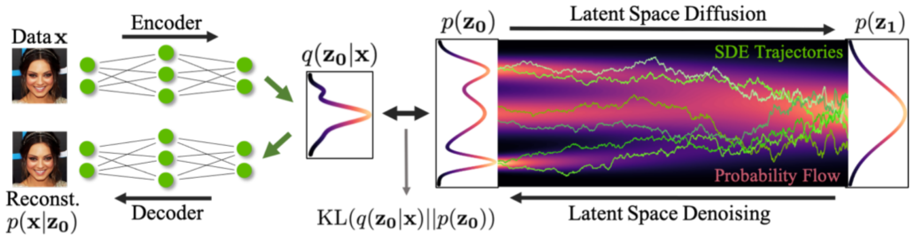


Figure 2.6: Nvidia latent Score-based Generative Model

2.3.4 CLIP

A brief overview of CLIP is provided to bring context to the next section of guided diffusion models.

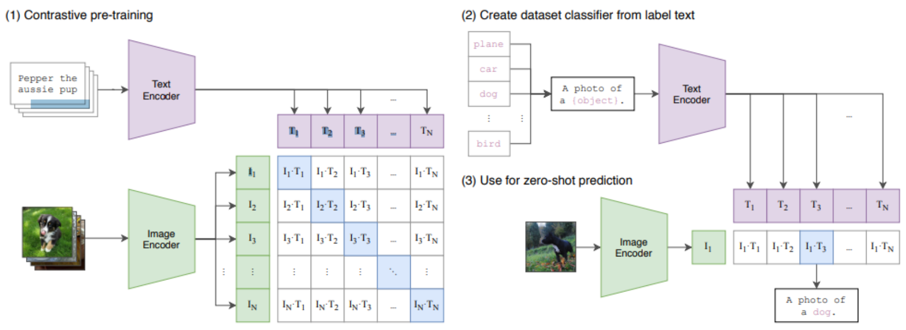

(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 … T_N
I_1 I_2 I_3 … I_N
$I_1 \cdot T_1$ … $I_N \cdot T_N$
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
(3) Use for zero-shot prediction
Image Encoder
I_1
A photo of a dog.

Figure 2.7: Overview of CLIP architecture (**radford2021learning**)

CLIP (Contrastive Language-Image Pretraining) is a state-of-the-art zero-shot classifier, i.e a classifier that has no prior knowledge of the classes that the images belong to. It was developed by OpenAI and introduced in their research paper "Learning Transferable Visual Models From Natural Language Supervision" in 2021. CLIP learns to map image and textual data into a common space, where relationships between the two modalities can be represented and understood. The model is trained on a massive amount of text and image data to extract meaningful semantic and visual features. It has been used in some guided diffusion models to provide contextual embeddings and to ensure the output image is consistent with a given prompt (**radford2021learning**).

2.3.5 Guided diffusion models

Guided diffusion models aim to add contextual information (conditioning) to the diffusion model to guide it towards a given prompt. This allows the user of the model to guide the output rather than getting a random image close to the data distribution from random noise.

There are two main approaches to guided diffusion: classifier, and classifier-free guided diffusion. **dhariwal2021diffusion** proposes to guide the image generation process using the gradients of a classifier, with the additional cost of having to train the classifier on noisy images (**hu2022self**). Note that this model proved that diffusion models outperform other state-of-the-art generative models for guided image generation. It uses CLIP as a classifier 2.3.4, rather than a traditional CNN (convolutional neural network) classifier, trained on images and their respective sequentially noised counterparts to steer the model toward the conditioning.

On the other hand, classifier-free models do not require training of this external classifier model, but instead train the model both with conditioning, $(\mathbf{y})$ and without $(\mathbf{0})$. At the inference stage, both $\epsilon_\theta(\mathbf{x}_t|\mathbf{y})$ and $\epsilon_\theta(\mathbf{x}_t|\mathbf{0})$ are calculated, and the difference is used to guide the model toward the captioned state with some multiplicative scale factor s (**nichol2021glide**).

The reasoning for this is that simply predicting the noise $\epsilon_\theta(\mathbf{x}_t|\mathbf{y})$ does not guide the sample towards the prompt enough, so by scaling the difference and adding it to the predicted noise we one can 'push' the model to sample towards the conditioning.

In the GLIDE paper, **nichol2021glide**, the score function for guided diffusion and its expected noise values are shown:

$$\begin{aligned}
\nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) &= \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|y) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) \\
&= -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\Big(\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\Big) \\
\bar{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, t, y) &= \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \sqrt{1-\bar{\alpha}_t}\; w \nabla_{\mathbf{x}_t} \log p(y|\mathbf{x}_t) \\
&= \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) + w\big(\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\big) \\
&= (w+1)\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, y) - w\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)
\end{aligned}$$

DALL-E 2, is an example of a guided diffusion model, released to the public for free in late 2022, and is built upon both GLIDE and CLIP.

2.4 Inpainting using Diffusion Models

This section familiarizes the reader with a conceptual understanding of image inpainting with diffusion models through RePaint and Stable Diffusion.

2.4.1 RePaint

The goal of inpainting is to predict missing pixels of an image using a mask region as a condition (**lugmayr2022repaint**). In RePaint, the diffusion model is applied within the context of image inpainting by using a standard DDPM model to fill in the mask.

$$\mathbf{x}_{t-1}^{\text{known}} \sim \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

The $\mathbf{x}_{t-1}^{\text{known}}$ diffusion step is equivalent to the standard forward diffusion process, in that the known input image is diffused with parameterised noise.

$$\mathbf{x}_{t-1}^{\text{unknown}} \sim \mathcal{N}(\mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

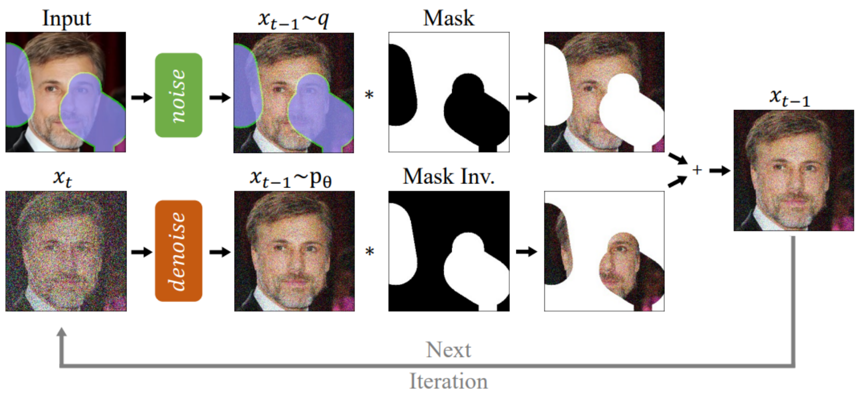


Figure 2.8: RePaint process (**lugmayr2022repaint**)

The unknown region (masked input) is sampled from the model (with the assumption that $\mathbf{x}_T^{\text{unknown}}$ is pure gaussian noise. Combining the regions together gives us the sample at $\mathbf{x}_{t-1}$.

$$\mathbf{x}_{t-1} = m \odot \mathbf{x}_{t-1}^{\text{known}} + (1 - m) \odot \mathbf{x}_{t-1}^{\text{unknown}}$$

The $\odot$ operator denotes the pixel-wise and operation with the mask. The sampling process reconstructs $\mathbf{x}_0$ by sequentially applying the above algorithm from $\mathbf{x}_T \approx$ (pure noise), and replacing the unmasked region of $\mathbf{x}_{t-1}$ with the known diffused pixels.

In the RePaint paper, they present quantitative results showing that it had beaten the previous state of the art approaches.

2.4.2 Stable Diffusion

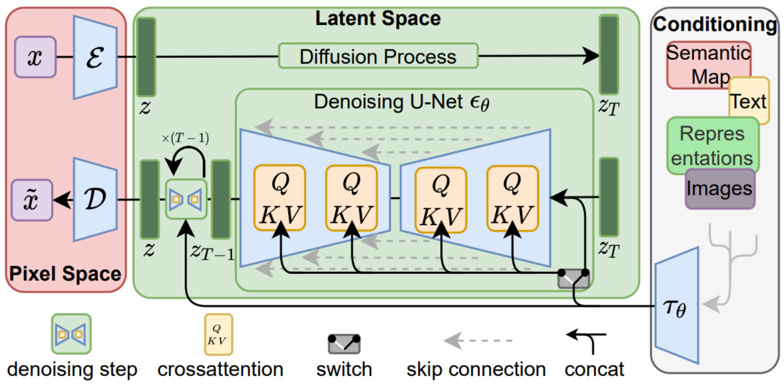


Figure 2.9: Stable Diffusion architecture

Stable Diffusion (**rombach2022high**) expands upon the principles established in RePaint by conducting the diffusion process within the latent space, encouraging the model to learn a semantic representation of the input dataset. Furthermore, to improve computational efficiency during inference, it employs a Denoising Diffusion Implicit Model (DDIM) scheduler. This enhancement significantly reduces computational costs at the expense of a minor decrease in image quality.

Stable Diffusion is also a guided diffusion model. This implies that during the inpainting task, the user can influence the model's inpainting behaviour using a text-based prompt.

This paper utilizes Stable Diffusion in its evaluation of mask-generation methods due to its expedited inference time and increased flexibility provided by its guidance capabilities.

2.5 Image segmentation methods for image inpainting

This section delves into the nuances of three prominent mask-generation methods, each of which brings unique advantages to the field of mask-free image inpainting. Both CLIPSeg and DiffEdit are capable of accepting input text as a query, whereas Segment Anything offers highly precise masks but does not support text queries, making it an excellent point of reference.

2.5.1 CLIPSeg

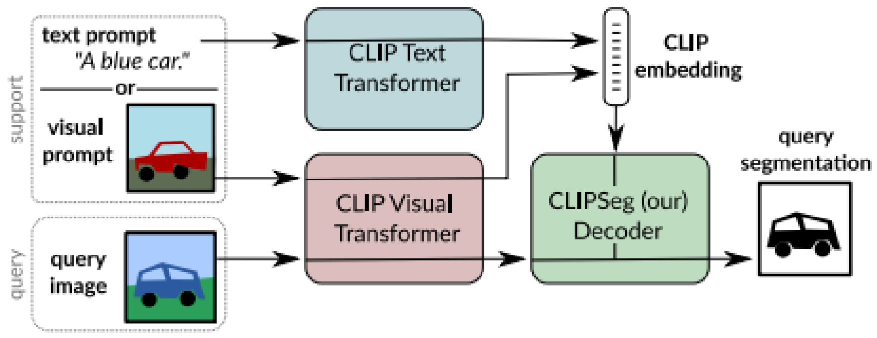


Figure 2.10: CLIPSeg model architecture

CLIPSeg, as presented by **luddecke2021prompt**, is a sophisticated segmentation model that leverages the capabilities of the CLIP (2.3.4) text and visual models. The model capitalizes on CLIP's ability to comprehend and connect a text prompt (or an alternative image) to the input image to generate a mask of the specified region. Owing to its transformer-based architecture, CLIPSeg excels in terms of its inference speed and mask quality. The model's rapid inference time is particularly noteworthy, and this, combined with the high-quality masks it generates, makes it a formidable tool in the realm of image segmentation for inpainting tasks.

2.5.2 DiffEdit

DiffEdit (**couairon2022diffedit**) is an innovative image segmentor that utilizes the capabilities of conditional diffusion models to generate masks. The process involves several steps. Initially, the input image is transformed into a noised version using the forward process of the diffusion model. Subsequently, the reverse process is applied to reconstruct the original image using two distinct queries that have a contrasting region, which represents the area we intend to mask. This process is repeated more than ten times to create a depth map. Finally, the depth map undergoes further processing through thresholding to generate the desired mask.

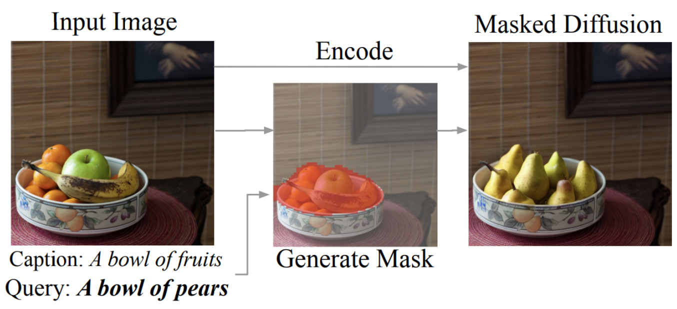


Figure 2.11: Diffedit process

The process is computationally expensive, although allows for the same model to be used for both inpainting and mask generation, possibly optimal where GPU memory is a bottleneck.

2.5.3 Segment Anything

Meta AI's Segment Anything (**kirillov2023segment**) is a high-speed segmentation model, trained on the largest mask-image dataset to date, containing over 1 billion masks and 11 million images. The model supports input in the form of points and bounding boxes, which are used as prompts to select the appropriate mask. Notably, it does not currently support text-based prompts. Like CLIPSeg, Segment Anything employs a transformer-based architecture, leveraging its capabilities for parallel processing and sequence learning to enhance the speed and accuracy of its segmentation tasks. This architectural design choice contributes to the model's impressive performance and its quick turnaround time.

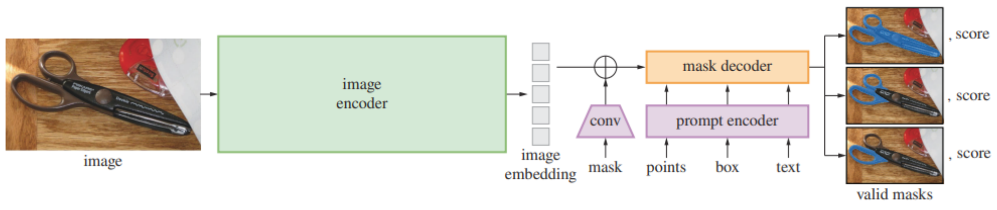


Figure 2.12: Segment Anything model architecture

The paper suggests that text based prompts will be supported in the future, however since it can segment everything in an input image, we can reduce this to a classification problem using an image to text decoder such as CLIP as later discussed.

Chapter 3

Requirements Capture

This phase aims to construct a detailed brief, capturing our objectives, and the corresponding deliverables.

3.1 Key Requirements

Our project's key requirements encompass the exploration, improvement, and evaluation of methods to generate high-quality masks from text prompts, particularly for use in diffusion models. Specifically, the requirements include:

High-Quality Mask Generation We require the development of an effective and efficient method for generating high-quality masks from textual prompts. The performance of this approach will be measured by its ability to produce masks with high resolution and accuracy.

Text-based Segmentation Methods To achieve the requirement above, we aim to evaluate and compare the current State-Of-The-Art (SOTA) text-based segmentation methods such as DiffEdit, CLIPSeg, and Segment Anything. The goal is to understand their capabilities, limitations, and potential areas of improvement.

Extension of Existing Models Furthermore, an essential requirement is to extend existing models like Meta's Segment Anything to include segmentation based on textual prompts. This would add a new dimension of usability to these models, potentially improving their performance and utility.

Pre-trained Diffusion Models The use of pre-trained diffusion models for inpainting and mask generation is another key requirement. These models can offer a strong starting point for our research, enabling us to accelerate the development process and leverage prior knowledge and techniques.

Seamless Restoration and Visual Consistency The final deliverable should ensure seamless restoration and visual consistency in inpainted images. Our success will be gauged on the model's ability to generate realistic and visually coherent images post-inpainting.

3.2 Project's Wider Aims and Objectives

Beyond these specific requirements, this project seeks to address broader objectives. The overarching goal is to enhance image inpainting models by incorporating an efficient method of mask generation using text prompts. This approach aims to simplify image editing and object removal processes, making these operations more accessible and user-friendly, especially for non-technical users.

In summary, we set the foundation for a project that combines cutting-edge research with practical application. This phase not only outlines the requirements necessary for the successful completion of this project, but it also aims to align all involved towards the wider objectives. By building upon existing models and developing new approaches for mask generation from text prompts, we aim to advance the field of image inpainting and make it accessible to a wider audience.

Chapter 4

Design and Implementation

4.1 Dataset

While models such as Stable Diffusion, Segment Anything, and CLIP are trained on extensive and varied datasets, implementing the mask-generation task requires only a limited number of images. These images are primarily used for testing and providing a comparative analysis between various methods.

In the context of this study, a dataset was composed of 18 photographs taken in South Kensington and on the Imperial College London campus. These images were specifically chosen to demonstrate the model's capability to remove occluding objects as well as random ones, thereby highlighting the model's ability to generate accurate segmentation for these objects.

To ensure a balance between efficient inference time and maintaining perceptual quality, all images were resized to a resolution of 512x512 pixels.

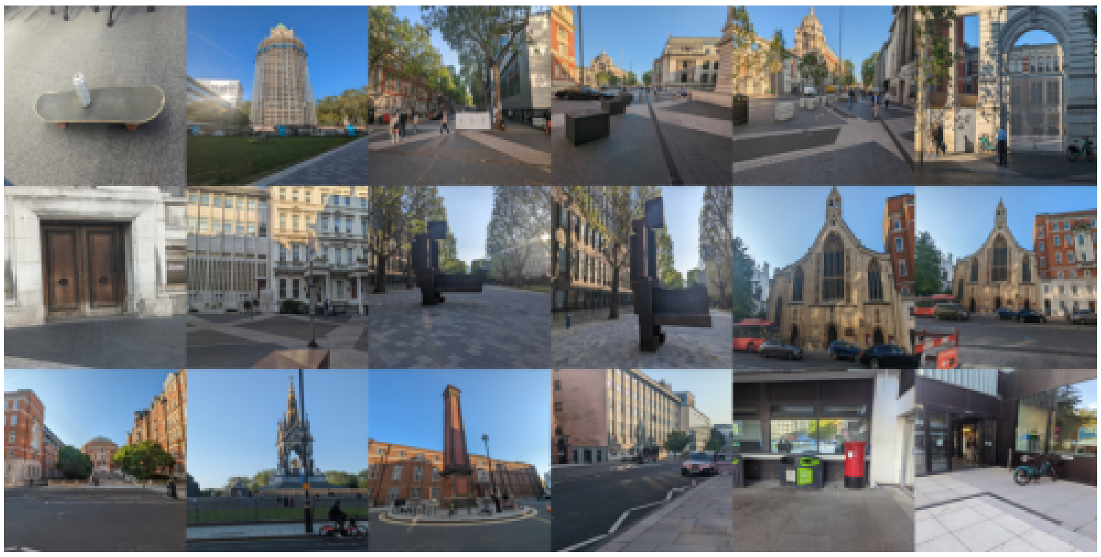


Figure 4.1: Collected Dataset of photos around Imperial College Campus

4.2 Hardware requirements

All experiments were run on Python 3 with a NVIDIA A100 Tensor Core GPU.

4.3 Design Process

This research project primarily aims to examine diffusion models and their potential application for image inpainting, which presents a variety of subfields for investigation.

4.3.1 Initial Steps

Our initial strategy involved training a Denoising Diffusion Probabilistic Model (DDPM) using the Celeb-A faces dataset. Following this process, it became evident that considerable computational resources, power, and time were necessary to train such a network for generating credible images.

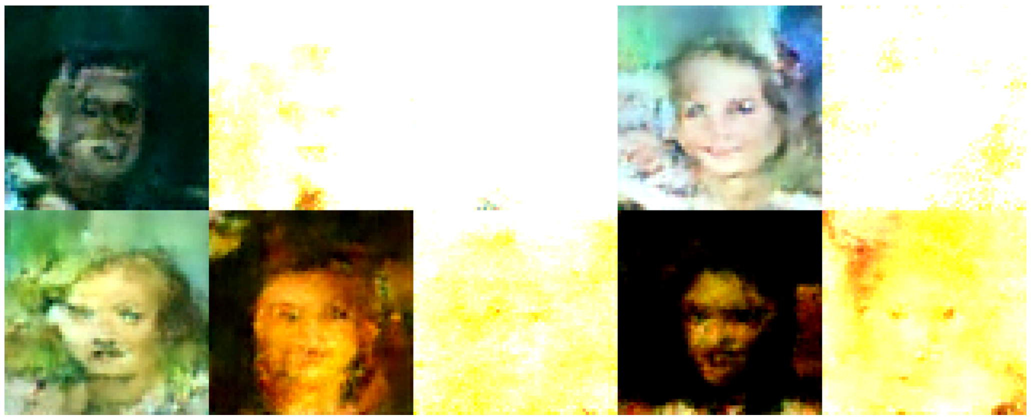


Figure 4.2: Trained DDPM Samples

4.3.2 Mask Generation

Switching our focus to inpainting rather than image generation, it became apparent that mask generation techniques served as an optimal initial step. Leveraging text prompts for mask generation appeared the most effective way to allow end-users, typically operating on smartphones or computers without dedicated GPUs, to inpaint with diffusion models seamlessly.

Consequently, we employed models such as DiffEdit and CLIPSeg to execute the mask-generation process, utilizing pre-trained models like RePaint and Stable Diffusion to inpaint.

4.3.3 Enhancing DiffEdit

Observing that DiffEdit generally consumes more time than most other segmentation methods, we sought to minimize computational complexity and inference time. Given that DiffEdit requires more than 20 diffusion process iterations for a single mask, our approach was to limit this to between 5 to 10 iterations with a higher threshold. We planned to use these keypoints with a conventional segmentation model to reduce inference time.

As depicted in figure 4.3, the detected keypoints capture the intended object effectively. However, the presence of anomalous points presents a significant challenge as it prevents segmentation models from accurately masking only the desired region. This issue is less pronounced in DiffEdit when used as a mask, as the inpainting models seamlessly fill the noisy mask areas.

4.3.4 SegCLIP

This paper's primary focus, SegCLIP, serves as a method to incorporate textual prompts into Meta AI's powerful Segment Anything model while preserving the model's computational speed. We achieved rapid inference of the masked region by employing the CLIP model to select the most

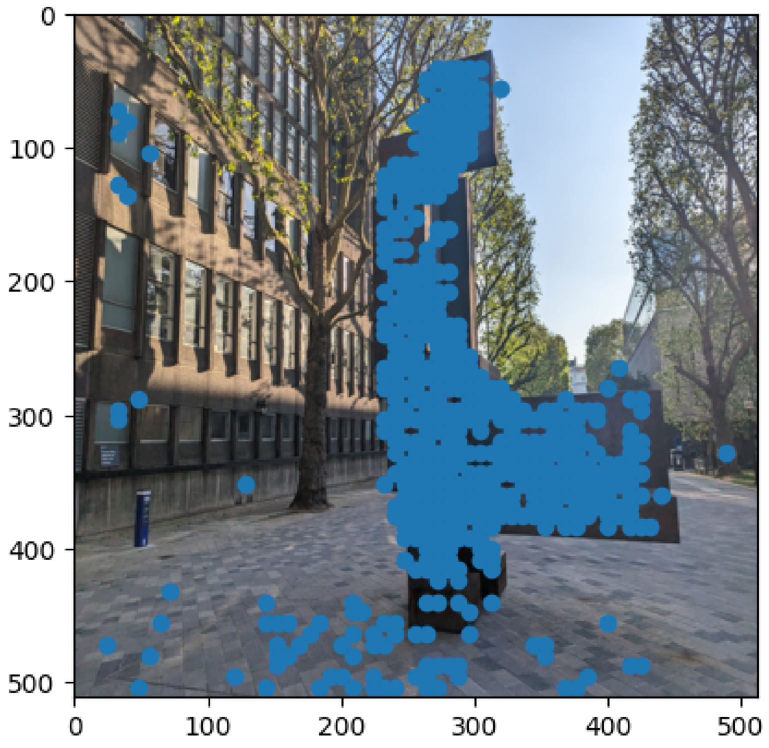


Figure 4.3: Keypoint generation using DiffEdit with prompts 'A sculpture in the street' and 'A street'

probable matches. This method also leverages Segment Anything's high precision in mask generation. A more comprehensive explanation of the model is provided in 4.4.

4.4 Implementation

4.4.1 High-level overview

SegCLIP begins with utilizing Segment Anything to segment everything in the input photo. This returns all the masks, and masked images. The CLIP model then computes a similarity matrix between the input text prompt and the masked images, to return the index of the optimal mask.

The optimal mask is post-processed and then inpainting with Stable Diffusion (or any inpainting algorithm of your choice) is performed.

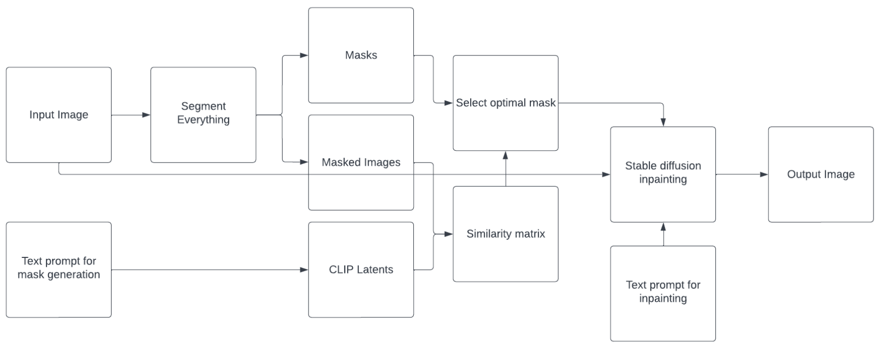


Figure 4.4: SegCLIP architecture

Segment Everything

In SegCLIP, we utilise the ability of Segment Anything to segment everything in the photo.

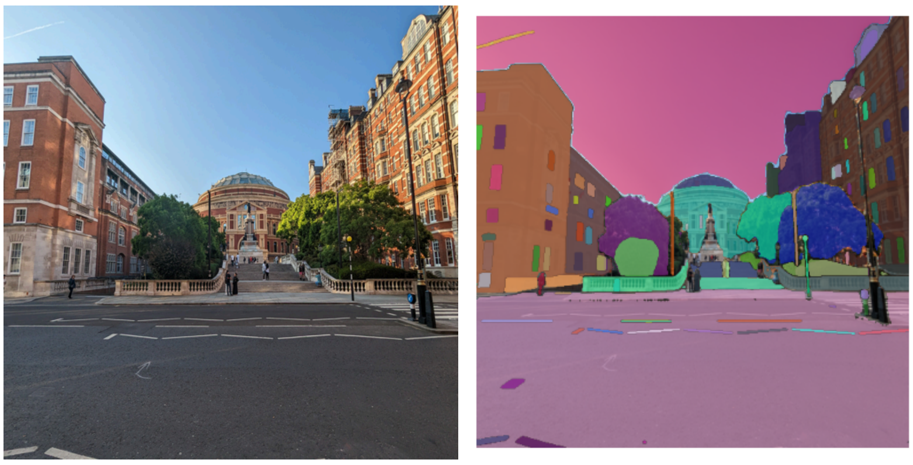


Figure 4.5: Generated masks using Segment Anything

All generated masks are applied to the input image, and the masked images are cropped so that an array of segmented objects are returned.

CLIP selection

Subsequently, the most optimal mask is chosen by computing a similarity matrix between the segmented objects and the input text prompt latents. Currently, only one object can be selected using this algorithm. An option to remedy this is to "OR" multiple masks which have a CLIP score above a certain threshold, however determining this threshold is tricky and introduces the possibility of irrelevant objects being included in the masked region.

Mask post-processing

Experimentally, more precise masks seem to produce bad outputs, as shadows and missed pixels around it have a strong effect on the inpainting algorithm. To improve inpainting quality, SegCLIP blurs the edges of the masks to make them slightly larger and smoother. This effect is clear when you consider the difference between SegCLIP samples and Segment Everything samples, which otherwise would look exactly the same.

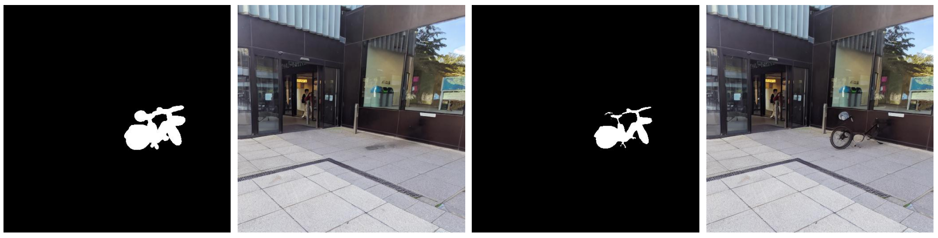


Figure 4.6: SegCLIP (left) Segment Anything (right)

We see that from a few unmasked edge pixels, the inpainting algorithm hallucinates the bicycle we are trying to remove.

4.4.2 Code repository

The implementation for SegCLIP and other mask-generation algorithms are available in this notebook.

4.4.3 Sampling

An example of mask generation and inpainting using SegCLIP and various other models is shown below.

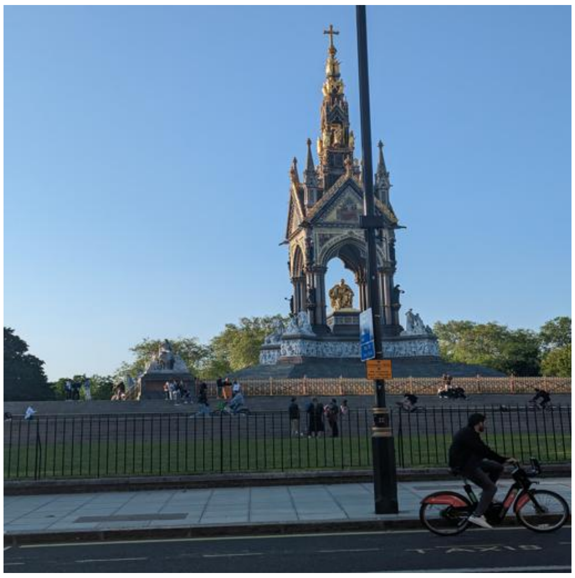


Figure 4.7: Original image

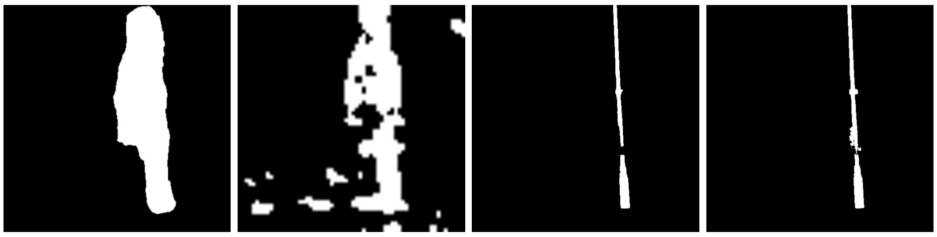


Figure 4.8: Mask generated with input prompt "pole"

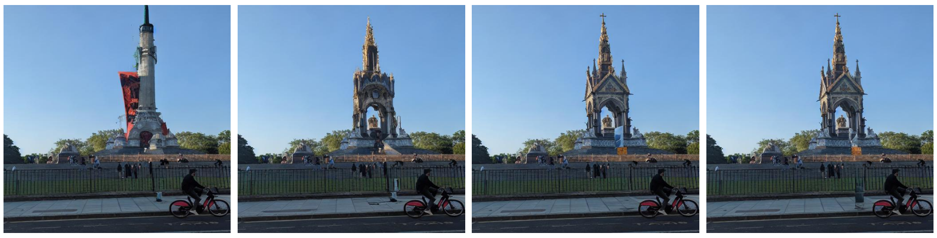


Figure 4.9: From left to right: CLIPSeg, DiffEdit, SegCLIP, Segment Anything with input points

Chapter 5

Results and Evaluation

5.1 Qualitative Analysis

We undertook a qualitative assessment of SegCLIP and other mask-generation models to evaluate their performance. This involved generating a variety of masks pertaining to specific objects, followed by the subsequent inpainting of the masked image using Stable Diffusion. A carefully structured questionnaire (7.1) was employed to measure image quality and consistency with respect to the input image and the provided prompt.

5.1.1 Questionnaire Structure

Q1: Object Removal Efficacy "Has the object specified in the prompt been removed effectively?" [Yes (1) / No (0)]

Q1 is formulated to gauge the capability of the mask-generation model to accurately identify and eliminate the object of interest as defined by the prompt. A higher score implies superior performance.

Q2: Unintentional Alterations "Has any other part of the image been inadvertently altered or removed?" [Yes (1) / No (0)]

Q2 serves as a reliable metric to ascertain the potential destructiveness of the mask generation algorithm. The ideal scenario is zero unintentional alterations to the original image.

Q3: Inpainting Harmony "On a scale of 0-10, how well does the inpainted region blend with the rest of the image?" [0 (Poor Harmony) - 10 (Excellent Harmony)]

Q3 aims to quantify the quality of the mask by assessing the harmonious integration of the inpainted region with the overall image.

Sample Size

For a robust evaluation, the questionnaire was distributed to a sample size of 20 individuals. Each participant was asked to evaluate 16 images that were each inpainted after using four distinct mask-generation models. The responses provide a comprehensive and comparative understanding of the performances of these models.

5.1.2 Questionnaire Analysis and Ranking of Methods

This section presents a structured analysis of the results from the questionnaire, as reflected in the table (Table 5.1). The performance of each mask-generation model is evaluated according to five

Method	Q1 (σ)	Q2 (σ)	Q3 (σ)	Q3|Q1 (σ)	Q1 & $\sim$ Q2 (σ)
CLIPSeg	0.95 (0.06)	0.43 (0.21)	7.04 (1.01)	7.12 (0.96)	0.53 (0.19)
DiffEdit	0.48 (0.10)	0.89 (0.06)	4.85 (1.23)	4.38 (1.42)	0.025 (0.04)
SegCLIP (ours)	0.78 (0.08)	0.30 (0.21)	7.19 (0.89)	7.42 (0.88)	0.54 (0.16)
Segment Anything	0.72 (0.12)	0.50 (0.24)	6.06 (0.99)	6.25 (0.92)	0.34 (0.15)

Table 5.1: Experiment results

key measures.

Q1: Object Removal Efficacy

When we assess the models' efficacy in removing the targeted object, CLIPSeg leads with a mean score of 0.95, also with the lowest standard deviation, suggesting it is the most reliable model at removing the given object. SegCLIP demonstrates a lower mean score of 0.78, which we attribute to the model's inability to capture multiple objects in the mask. The models DiffEdit and Segment Anything obtain lower mean scores of 0.48 and 0.72 respectively: we theorise that DiffEdit lacks due to poor recognition of small objects; Segment Anything due to the sharp mask boundaries.

Q2: Unintentional Alterations

SegCLIP outperforms the other models in minimizing unintended alterations to the image, achieving a mean score of 0.30. This implies a strong consistency in the preservation of non-targeted image content. In contrast, the other models, namely CLIPSeg, DiffEdit, and Segment Anything, exhibit larger mean scores and standard deviations, suggesting higher likelihoods of unintentional alterations during the mask generation process.

Q3: Inpainting Harmony

The capability of the models to generate visually coherent, inpainted images is assessed next. SegCLIP, with a mean score of 7.19 and standard deviation of 0.89, shows a high degree of harmony in its inpainted output. CLIPSeg closely follows with a mean score of 7.04, albeit with a slightly larger standard deviation of 1.01. In contrast, DiffEdit and Segment Anything yield lower mean scores, indicating less consistency in the visual coherence of their outputs.

Q3 Conditioned on Q1: Harmonious Inpainting Given Correct Object Detection

SegCLIP demonstrates an exceptional performance when considering the quality of inpainting given successful object detection (Q3|Q1). It achieves a mean score of 7.42 with a standard deviation of 0.88, which suggests a high degree of harmony in the inpainted region when the targeted object has been correctly identified and removed. The other models, namely CLIPSeg, DiffEdit, and Segment Anything, score lower mean values and show larger standard deviations under the same condition.

Q1 & $\sim$ Q2: Effective Object Removal with Minimal Unintentional Alterations

In evaluating the balance between effective object removal and minimal unintentional alterations, SegCLIP attains the highest mean score of 0.54. This suggests that SegCLIP is not only successful in accurately removing the specified object, but also excellent in maintaining the integrity of the non-targeted image content. CLIPSeg closely follows with a mean score of 0.53, while DiffEdit and Segment Anything lag behind with lower mean scores.

SegCLIP vs. Segment Anything

SegCLIP noticeably outperforms Segment Anything, as evidenced by the evaluation of the inpainted samples. Segment Anything does not process mask boundaries adequately for inpainting. Specifically, it falls short in extending and smoothing the borders of the segmented regions into a mask. SegCLIP, on the other hand, demonstrates better handling of mask boundaries, thereby yielding superior inpainting results.

DiffEdit's Performance Limitations

A notable aspect of our findings is the relative underperformance of DiffEdit across most metrics compared to the other models. This is largely attributed to the nature of its design, which utilizes diffusion prompts to guide the entirety of the image, rather than targeting specific objects within it.

This diffusion-based approach has inherent limitations when handling scenarios where the object of interest constitutes a small portion of the overall image. DiffEdit operates under the premise that the whole image should align with the provided text prompt, which may not always be appropriate. Specifically, in situations where the text prompt is supposed to direct the model to a small object or region within the image, DiffEdit struggles as it assumes that the prompt should dictate the nature of the entire image.

As a result, DiffEdit falls short in scenarios that involve minor elements or localized changes, which are often the target of mask-generation tasks. Its design assumption prevents it from adequately responding to prompts focused on such small-scale objects or alterations.

Consequently, despite its novel approach to using diffusion-based prompts, DiffEdit underperforms in our assessment, particularly in the accuracy of object removal and preservation of non-targeted image content. This observation highlights the importance of mask-generation models being flexible and adaptable to a wide range of text prompts, including those that pertain to small or localized image elements.

Summary

In conclusion, CLIPSeg excels in object removal and inpainting harmony, while SegCLIP showcases strength in minimizing unintended image alterations and generating visually harmonious inpainting, particularly when the correct object has been detected. Furthermore, due to the mask generation process being almost identical, SegCLIP's superiority over Segment Anything emphasises the need for adequate mask post-processing when linking segmentation models to image inpainting with diffusion models. With further improvements, especially in its capability to mask multiple objects, SegCLIP could set new benchmarks in the field of text guided mask generation for image inpainting.

5.2 Runtime analysis

Method	Avg. Time (s)
CLIPSeg	0.411
DiffEdit	42.00
SegCLIP (ours)	5.512
Segment Anything	0.518

Table 5.2: Average runtime for mask generation

Table 5.2 presents the average runtimes for each model. It can be observed that there is a significant range in the time taken by these models to generate masks. DiffEdit takes the longest time with an average of 42.00 seconds, indicating a much higher computational demand compared to the other models.

On the other hand, CLIPSeg and Segment Anything demonstrate significantly lower runtimes, with averages of 0.411 and 0.518 seconds respectively. This suggests greater computational efficiency, which could be a critical factor for real-time or large-scale applications.

SegCLIP, our proposed model, takes an average of 5.512 seconds to generate a mask. While this is higher than CLIPSeg and Segment Anything, it is still considerably lower than DiffEdit. Given SegCLIP's superior performance across several key qualitative metrics, this time complexity is acceptable, although there is potential for optimization. Further work focusing on computational efficiency could render SegCLIP more viable for real-time applications and accessible to a broader range of users.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this study, we introduce SegCLIP, an extension of Segment Anything, which incorporates the ability to generate masks via text prompts. We conduct an extensive comparative evaluation of SegCLIP within the realm of image inpainting, employing diffusion models. The evaluation of our performance metrics (as shown in Table 5.1) endorses the robustness of SegCLIP and its potential as a formidable asset in image editing tasks.

While CLIPSeg demonstrates a marginally superior performance in the area of object removal, SegCLIP distinguishes itself by its remarkable balance and proficiency in executing effective object removal with minimal unintended alterations to the image. In this composite metric, SegCLIP's score exceeds all other models, underlining its aptitude in preserving the integrity of the non-targeted image content—a critical factor influencing the quality of inpainting.

When we evaluate the congruity of the inpainted region with the rest of the image, SegCLIP consistently demonstrates high performance. Notably, when SegCLIP accurately identifies the correct object for removal, it generates an inpainted image that blends extraordinarily well with the rest of the image, outperforming all other methods in this regard.

Additionally, SegCLIP showcases a significant improvement over Segment Anything in terms of mask boundary processing—an essential element for smooth inpainting.

In conclusion, SegCLIP demonstrates impressive performance across a wide array of key metrics, reinforcing its potential as an efficient, robust tool for mask generation for image inpainting. Despite some areas where it is marginally outperformed, its overall consistency and balance across all areas make it an encouraging model for practical applications in the field of image editing.

6.2 Future Work

While the current study highlights the promise of SegCLIP, several areas for further development are identified:

- **Multi-Object Mask Generation:** One of the main limitations of SegCLIP is its inability to concurrently mask multiple objects that satisfy the text prompt. Implementing multi-object mask generation functionality would improve its performance, potentially bringing it on par with or surpassing the object removal efficacy of CLIPSeg.
- **Runtime Optimization:** The efficiency of SegCLIP, in terms of computation time, could be further optimized. Faster performance would make it more viable for real-time applications and user-friendly for non-technical users.
- **High Resolution Testing:** Although our current study has demonstrated the effectiveness of these models on images of 512x512 pixels, real-world applications often involve dealing

with images of significantly higher resolution.

- **Enabling Advanced Image Editing on Smartphones:** The evolution of mask-generation models with the ability to generate masks from text prompts could lead to the implementation of on-the-fly "find and replace" of objects in photos taken or stored on a smartphone.

Chapter 7

Appendix

7.1 Questionnaire Example

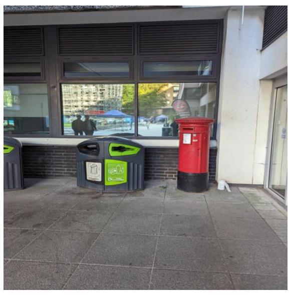


Figure 7.1: Original image with prompt “green bin”

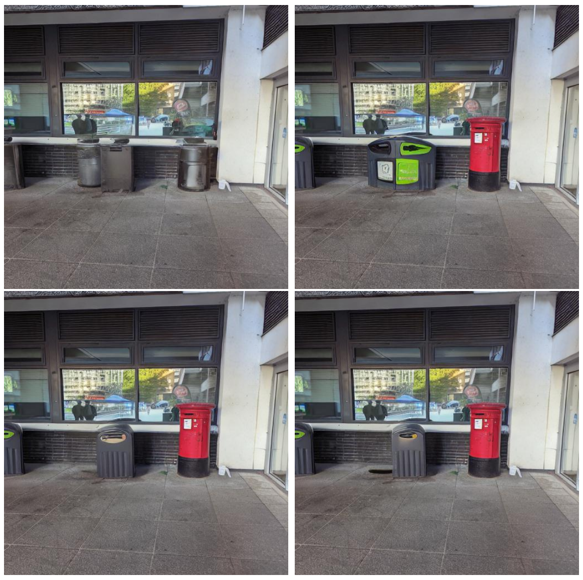


Figure 7.2: Image $\frac{1,2}{3,4}$

Q1 Has the object specified in the prompt been removed effectively? [Yes (1) / No (0)]

Q2 Has any other part of the image been inadvertently altered or removed? [Yes (1) / No (0)]

Q3 On a scale of 0-10, how well does the inpainted region blend with the rest of the image? [0 (Poor Harmony) - 10 (Excellent Harmony)]

	Q1	Q2	Q3
Image 1			
Image 2			
Image 3			
Image 4			

7.2 Samples

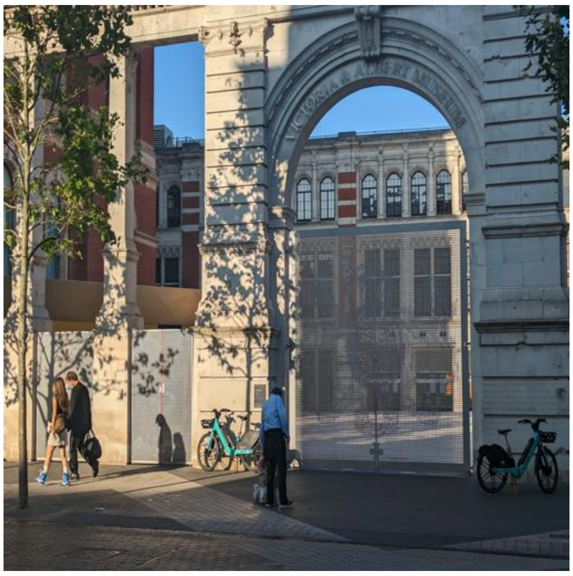


Figure 1: Original image with prompt "bicycle"

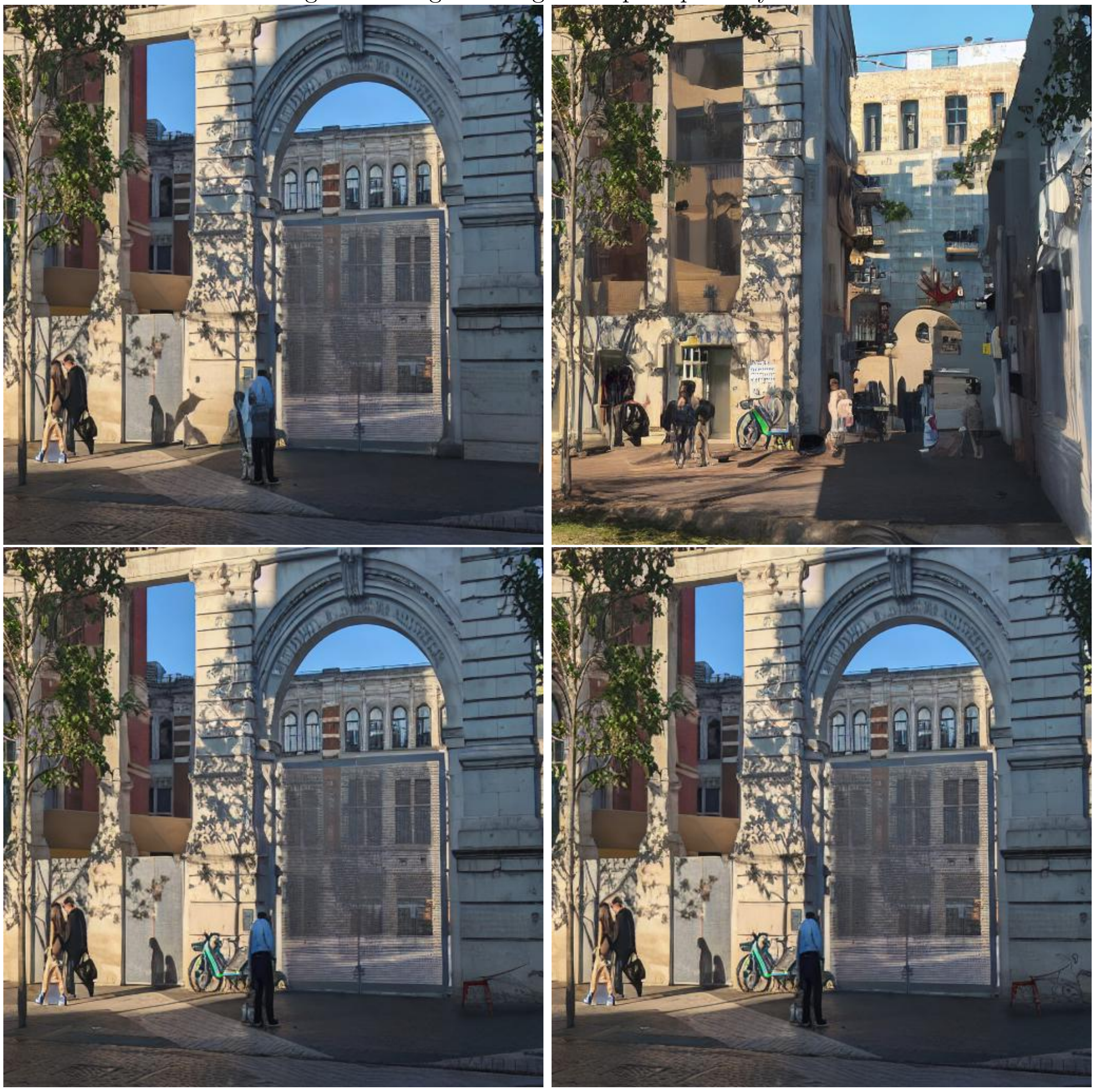


Figure 2: $\begin{array}{c|c} \textbf{CLIPSeg} & \textbf{DiffEdit} \\ \hline \textbf{SegCLIP} & \textbf{Segment Anything} \end{array}$

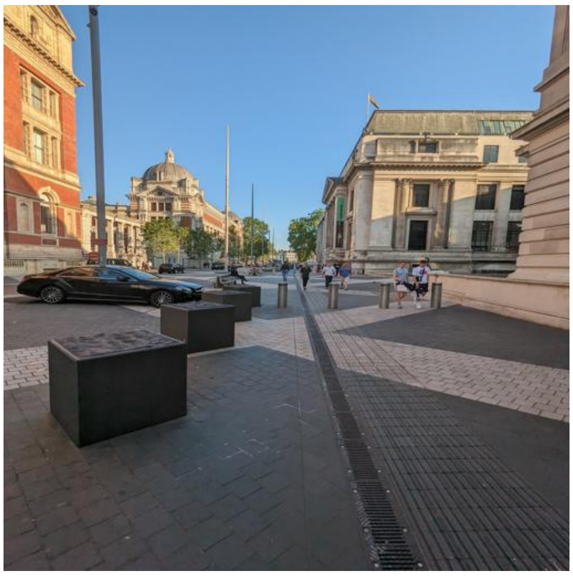


Figure 3: Original image with prompt ”black car”

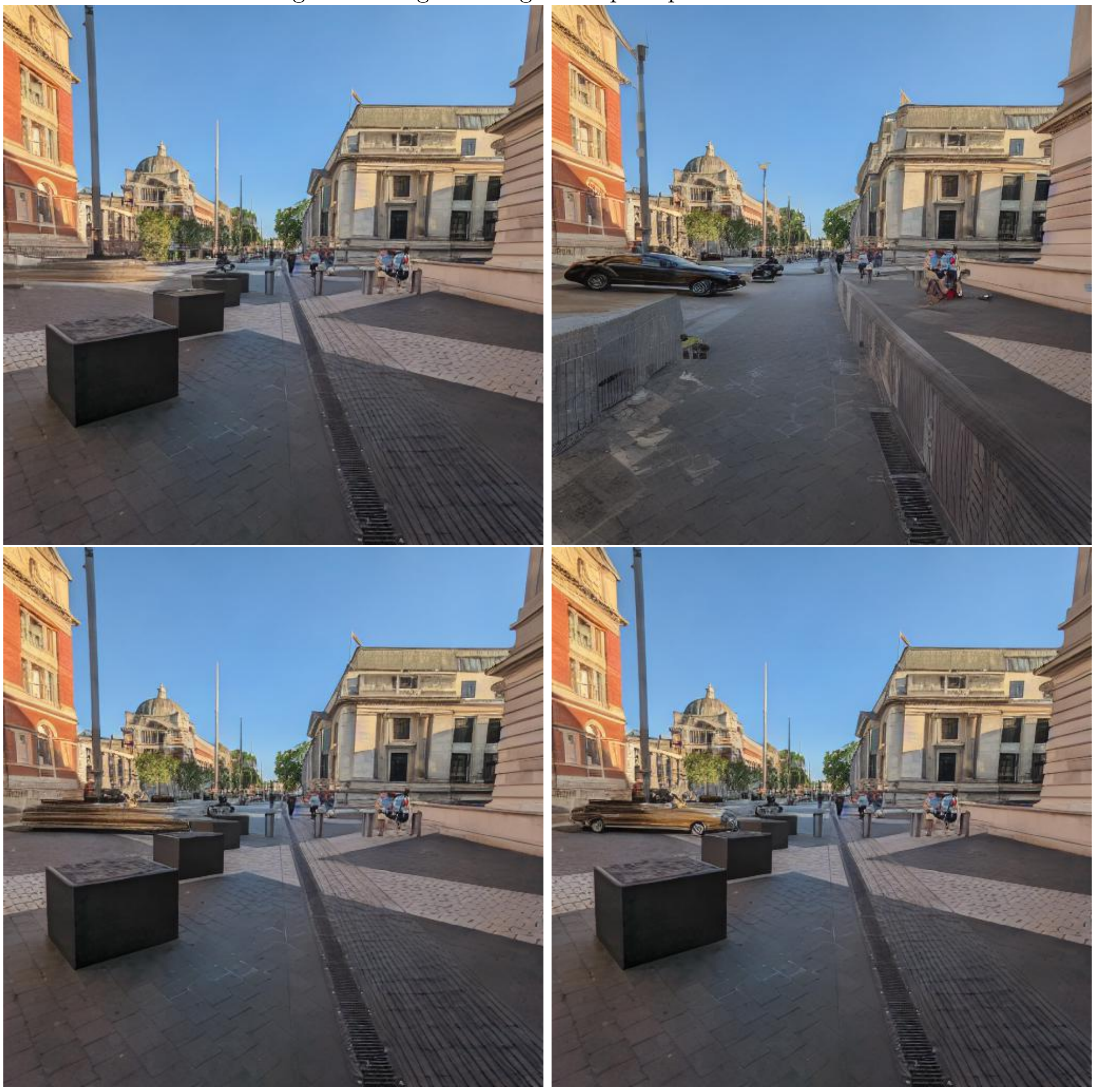


Figure 4: $\frac{\textbf{CLIPSeg}}{\textbf{SegCLIP}}\Big|\frac{\textbf{DiffEdit}}{\textbf{Segment Anything}}$

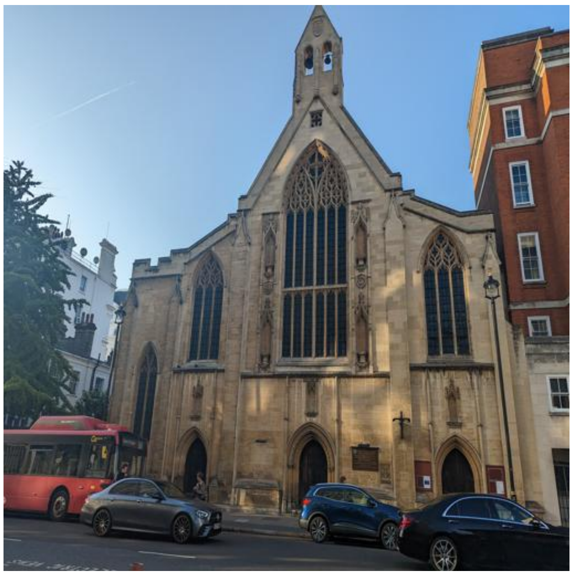


Figure 5: Original image with prompt "bus"

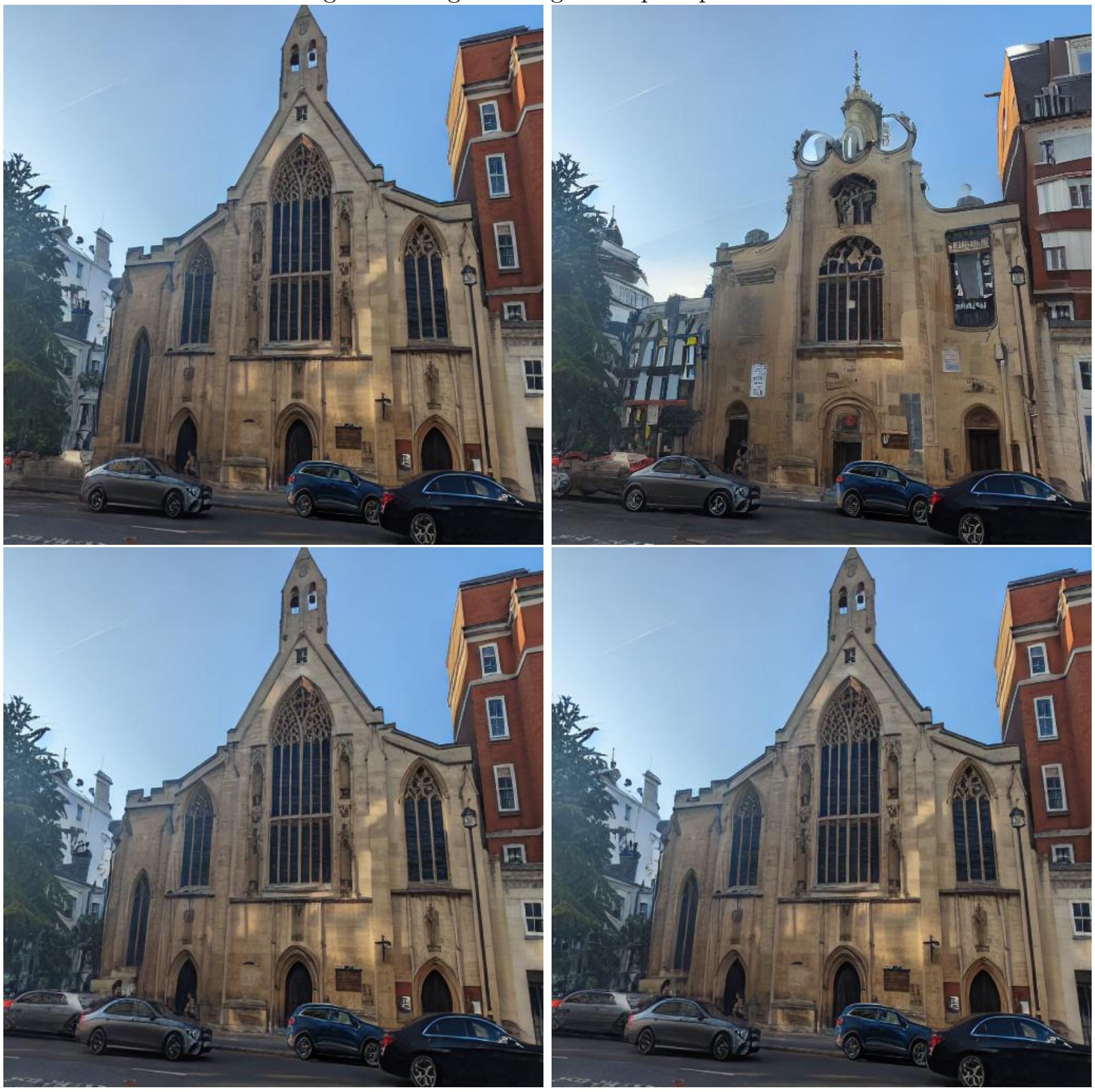


Figure 6: $\left.\frac{\textbf{CLIPSeg}}{\textbf{SegCLIP}}\right|\frac{\textbf{DiffEdit}}{\textbf{Segment Anything}}$

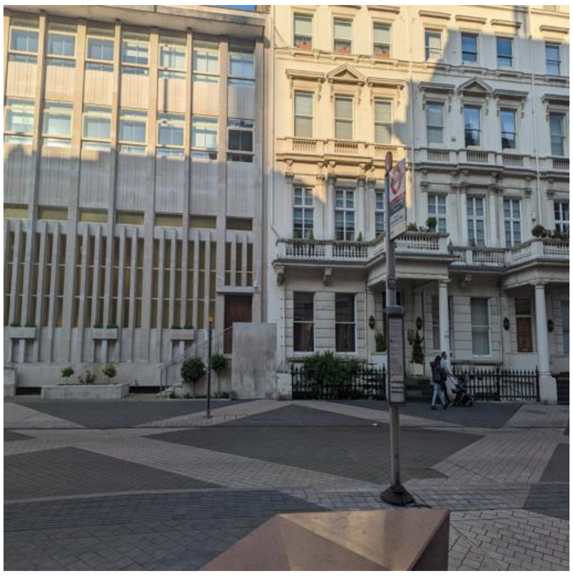


Figure 7: Original image with prompt "bus stop sign"

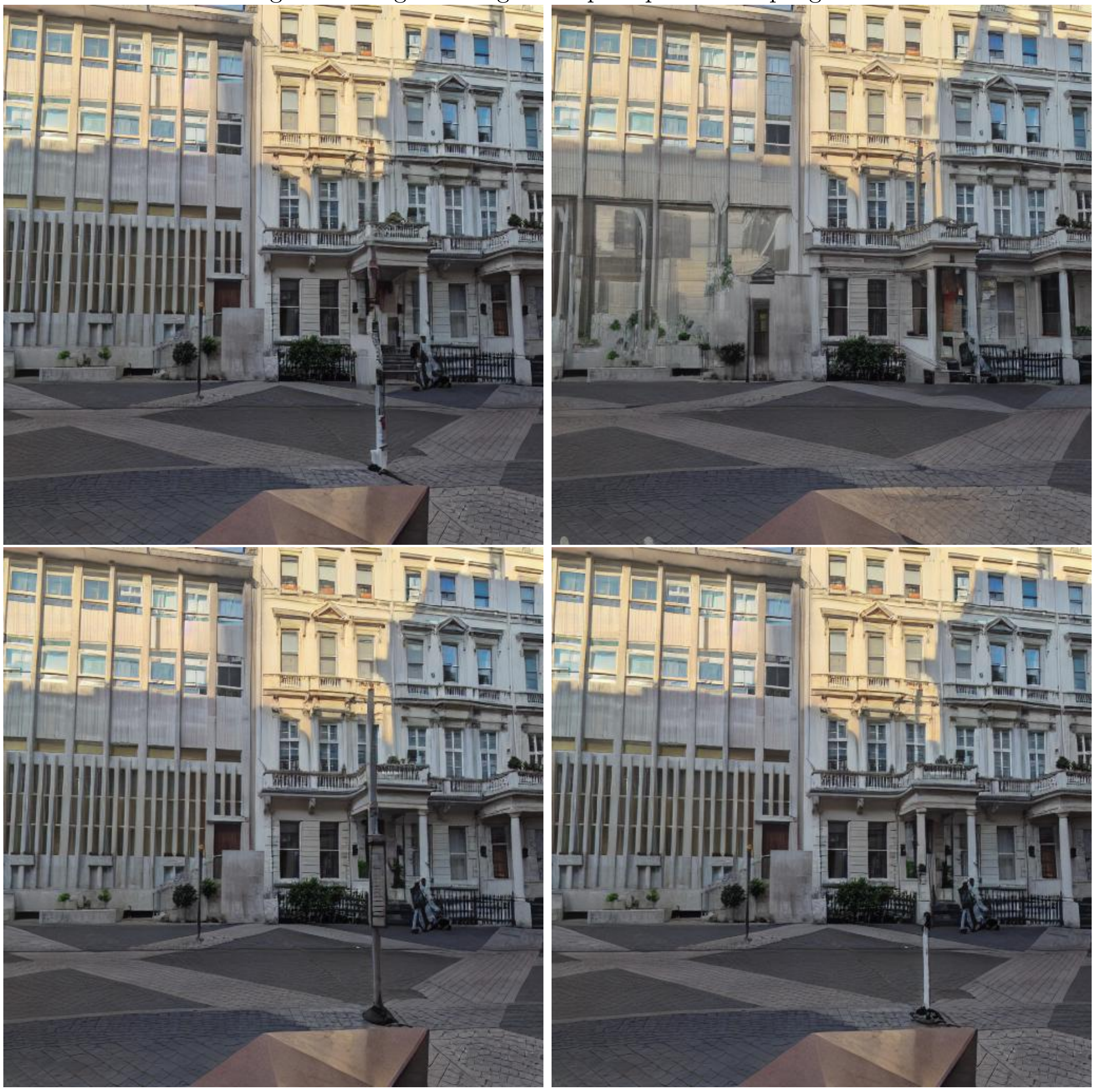


Figure 8: **CLIPSeg** | **DiffEdit** / **SegCLIP** | **Segment Anything**

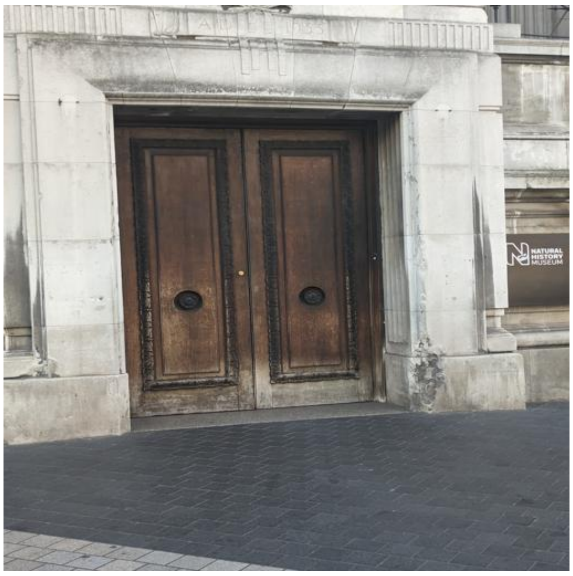


Figure 9: Original image with prompt "double doors"

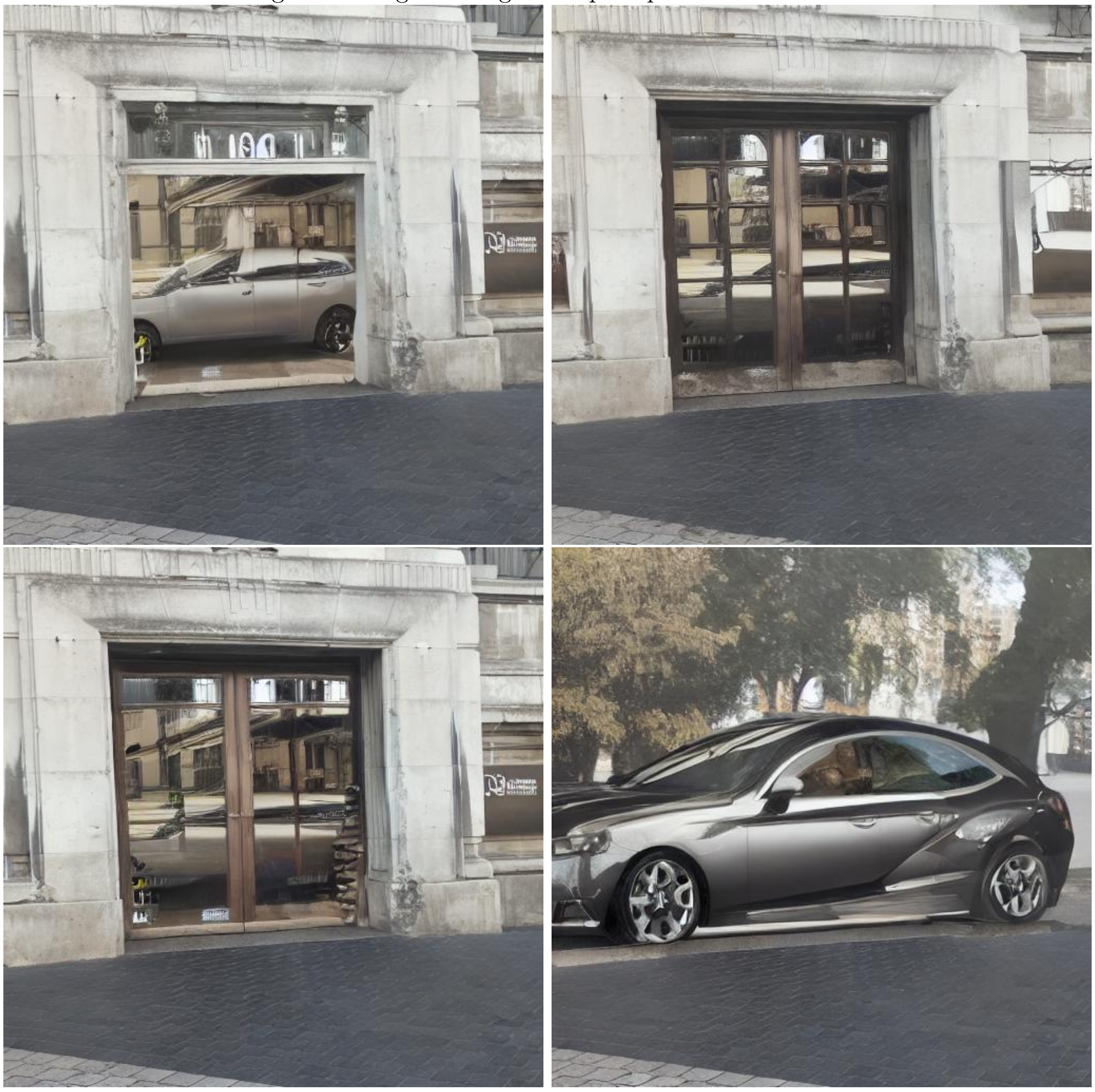


Figure 10: $\begin{array}{c|c} \textbf{CLIPSeg} & \textbf{DiffEdit} \\ \hline \textbf{SegCLIP} & \textbf{Segment Anything} \end{array}$

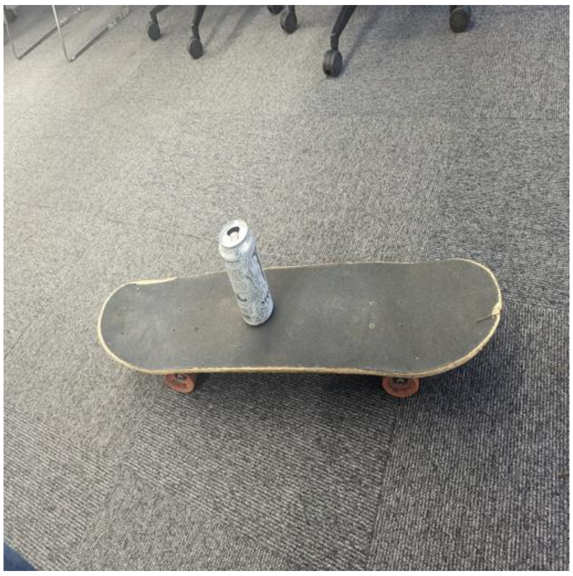


Figure 11: Original image with prompt ”white can”

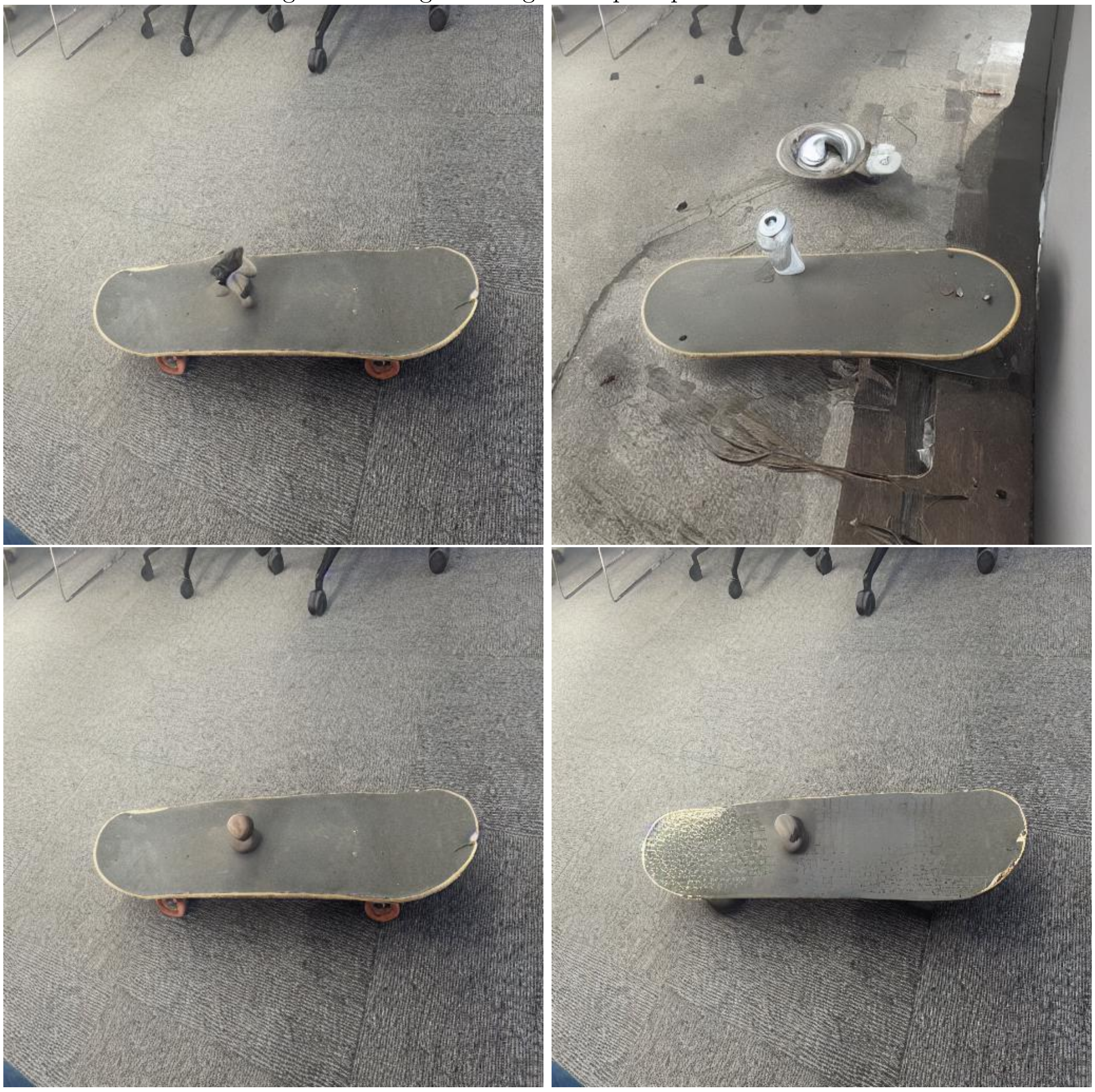


Figure 12: $\frac{\textbf{CLIPSeg}}{\textbf{SegCLIP}} \Big| \frac{\textbf{DiffEdit}}{\textbf{Segment Anything}}$

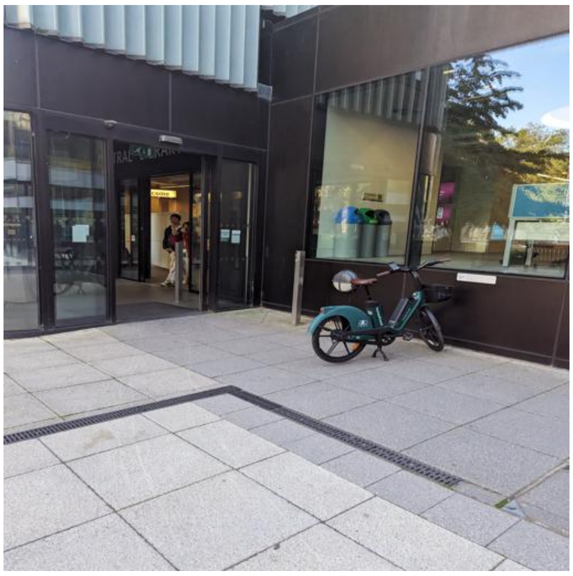


Figure 13: Original image with prompt "bicycle"

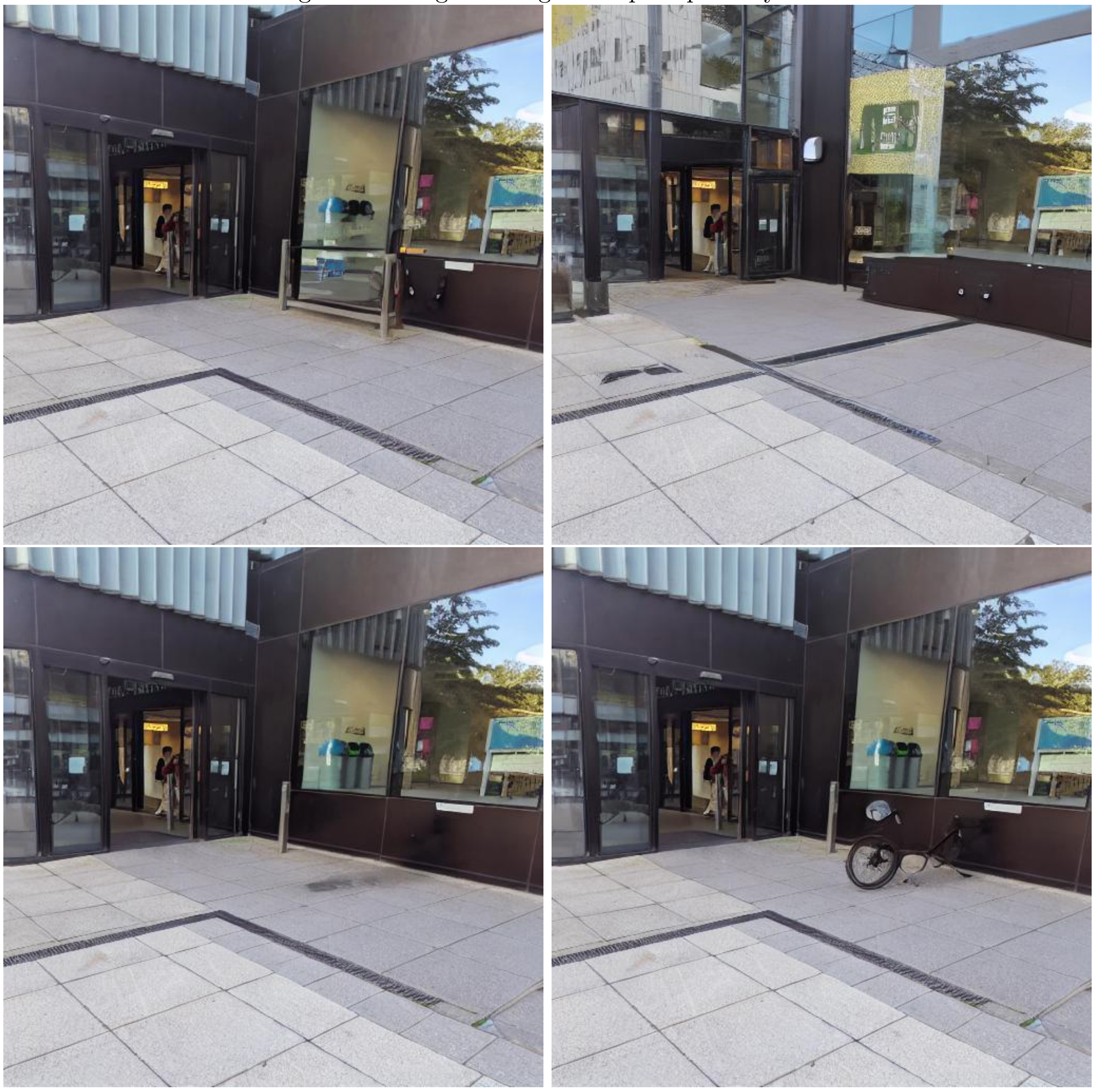


Figure 14: $\frac{\textbf{CLIPSeg}}{\textbf{SegCLIP}}\Big|\frac{\textbf{DiffEdit}}{\textbf{Segment Anything}}$

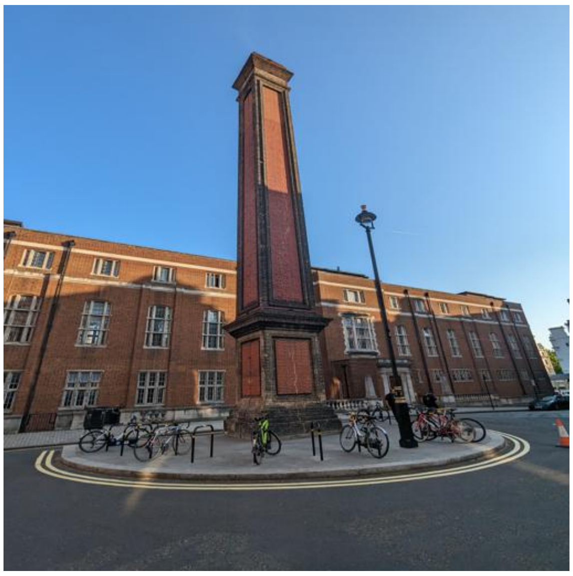


Figure 15: Original image with prompt ”brick pillar”

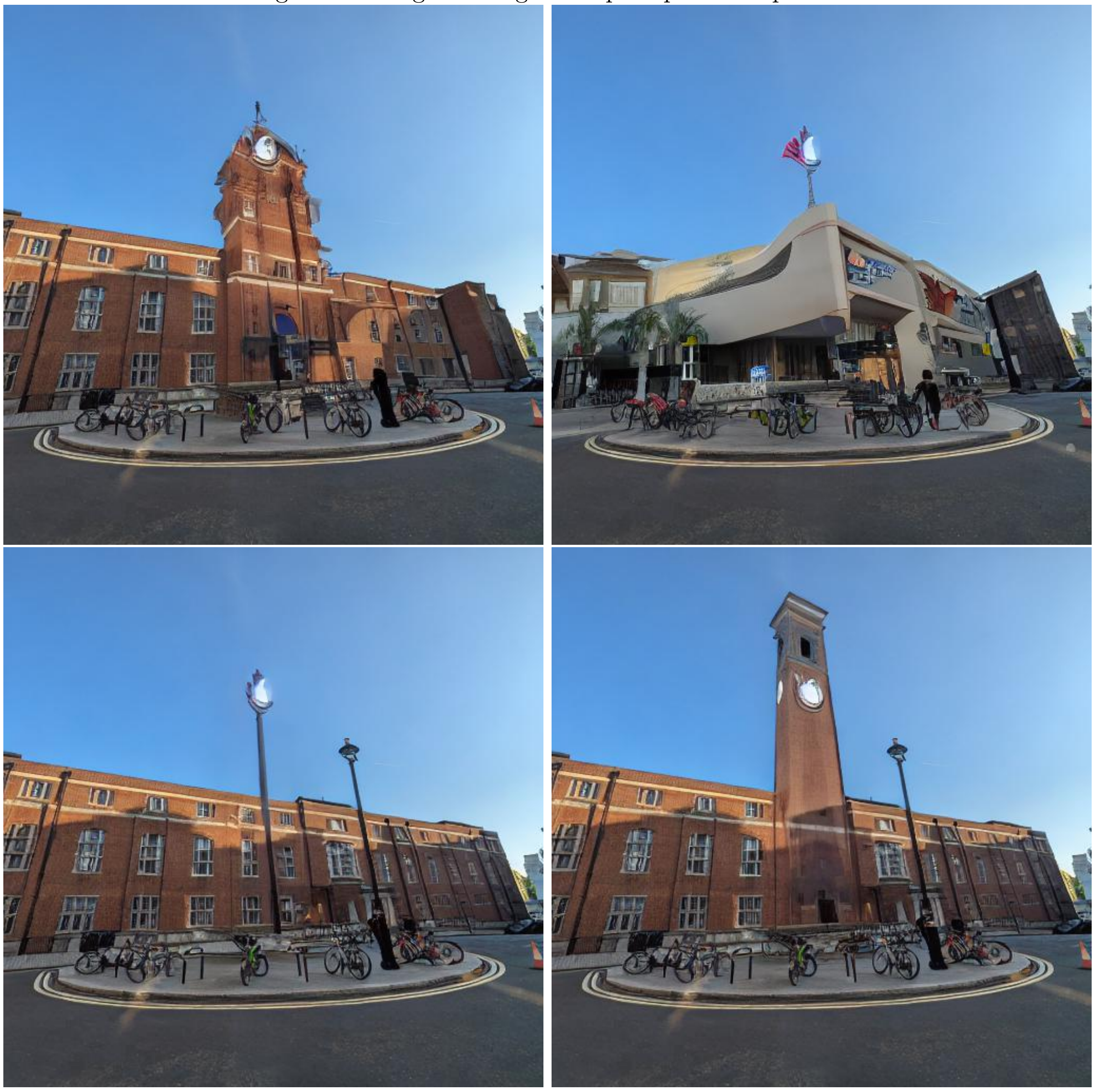


Figure 16: **CLIPSeg** | **DiffEdit**
SegCLIP | **Segment Anything**

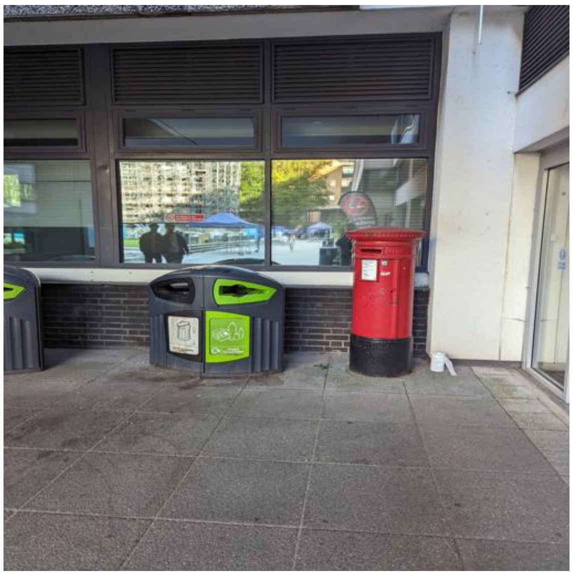


Figure 17: Original image with prompt ”red postbox”

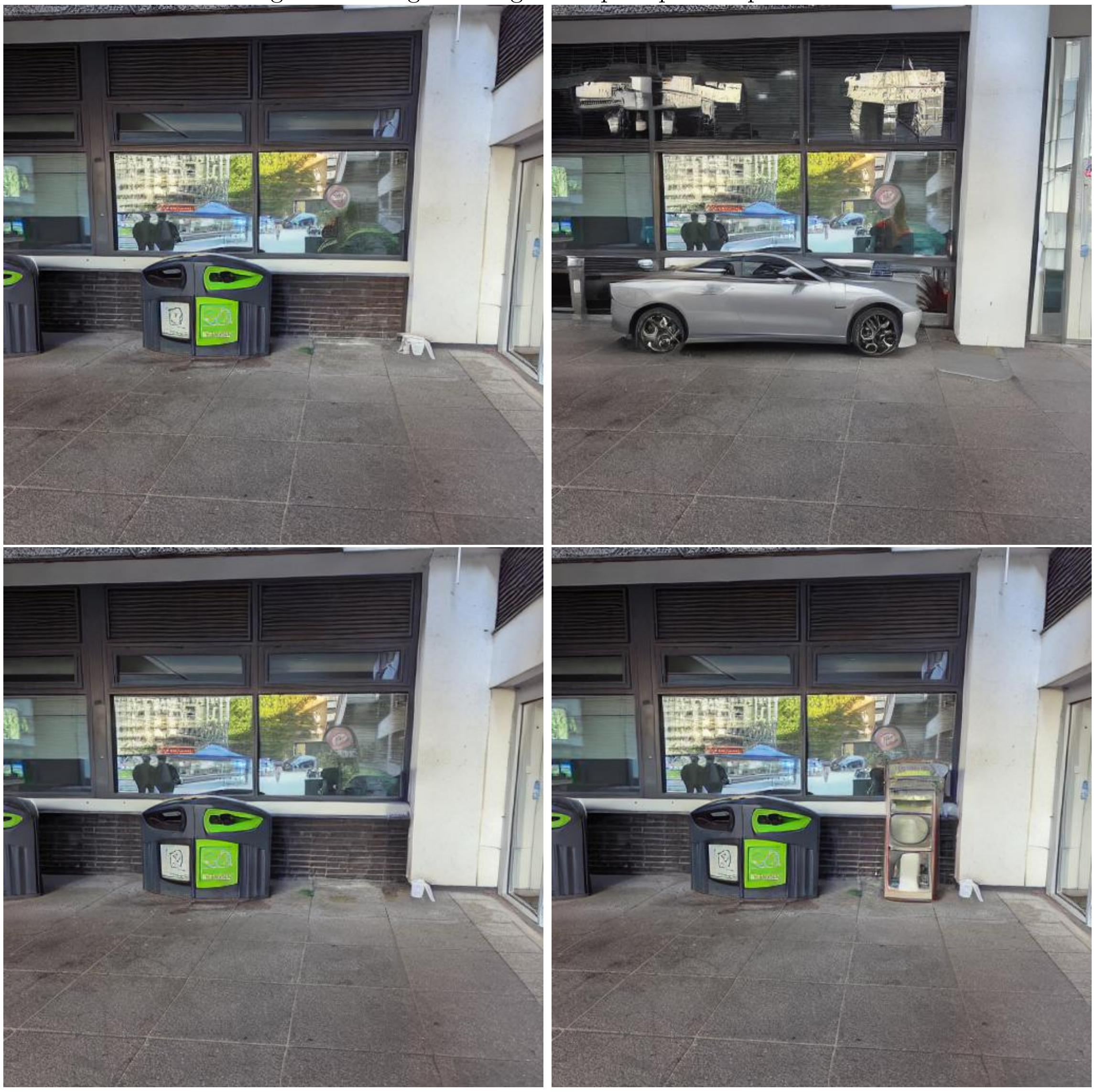


Figure 18: **CLIPSeg**|**DiffEdit**
SegCLIP|**Segment Anything**

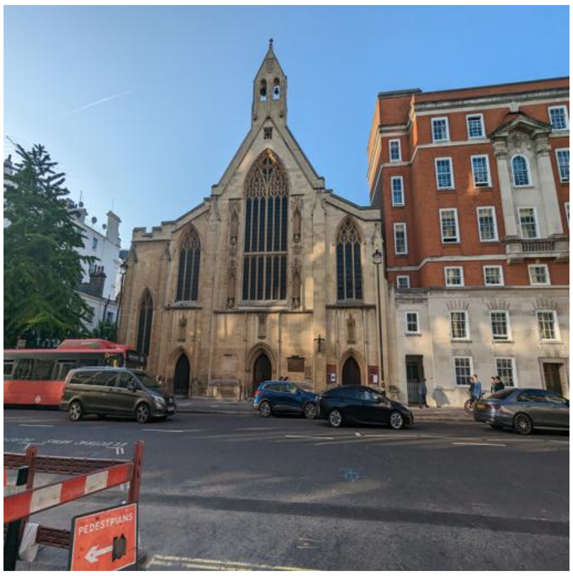


Figure 19: Original image with prompt ”trees”

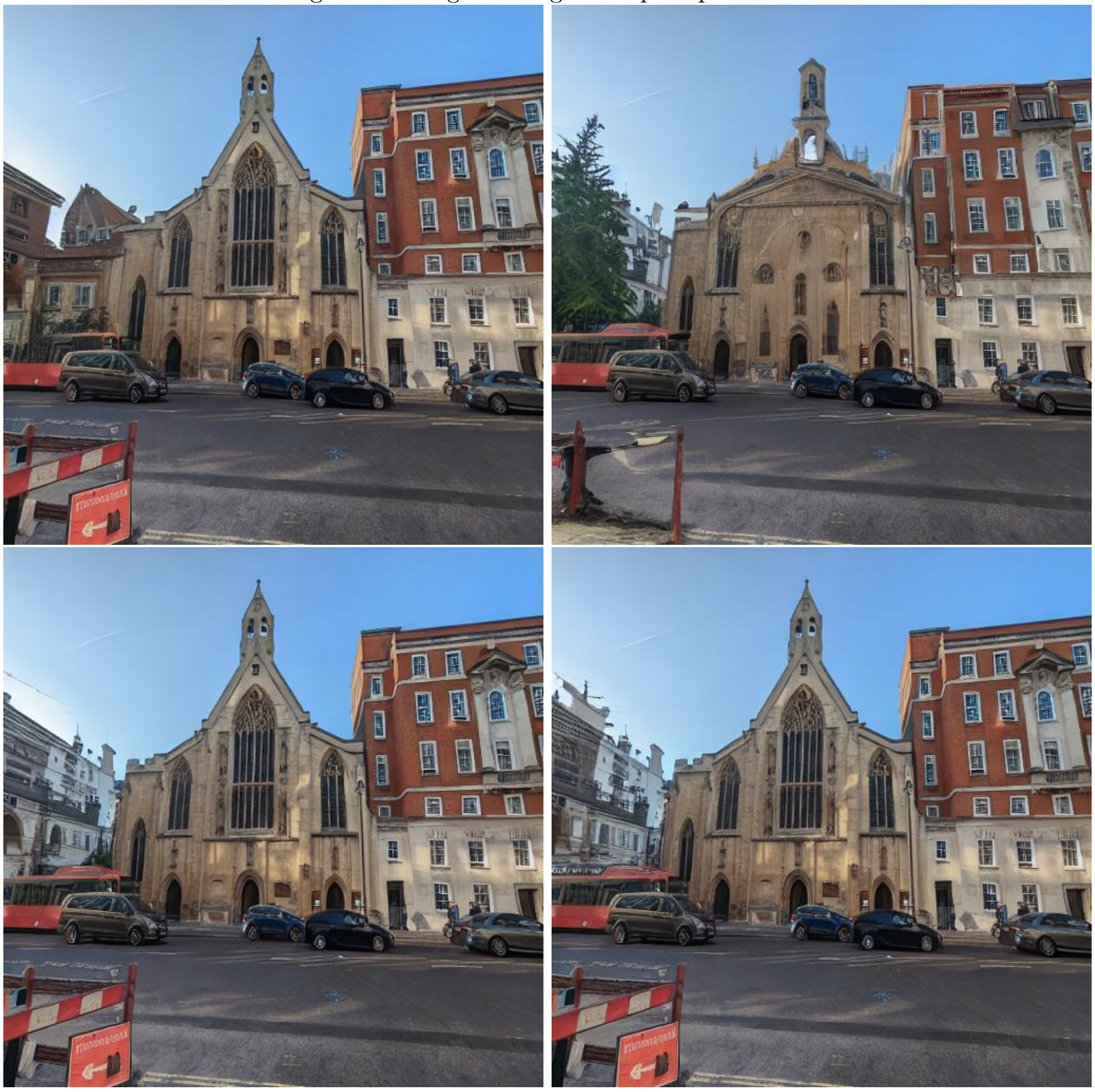


Figure 20: $\frac{\textbf{CLIPSeg}}{\textbf{SegCLIP}}\Big|\frac{\textbf{DiffEdit}}{\textbf{Segment Anything}}$

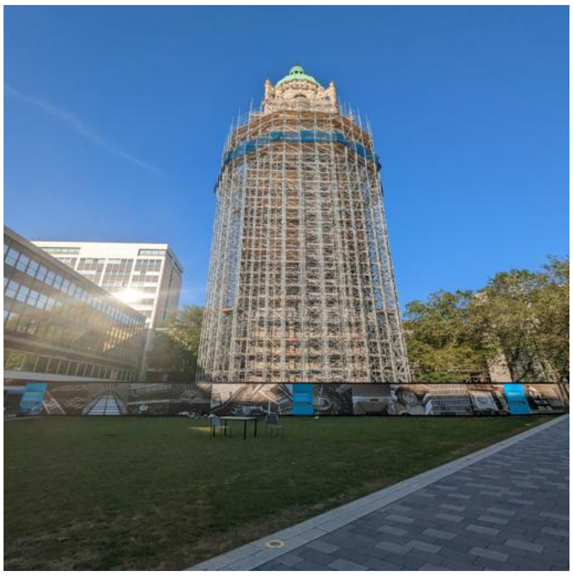


Figure 21: Original image with prompt ”scaffolding”

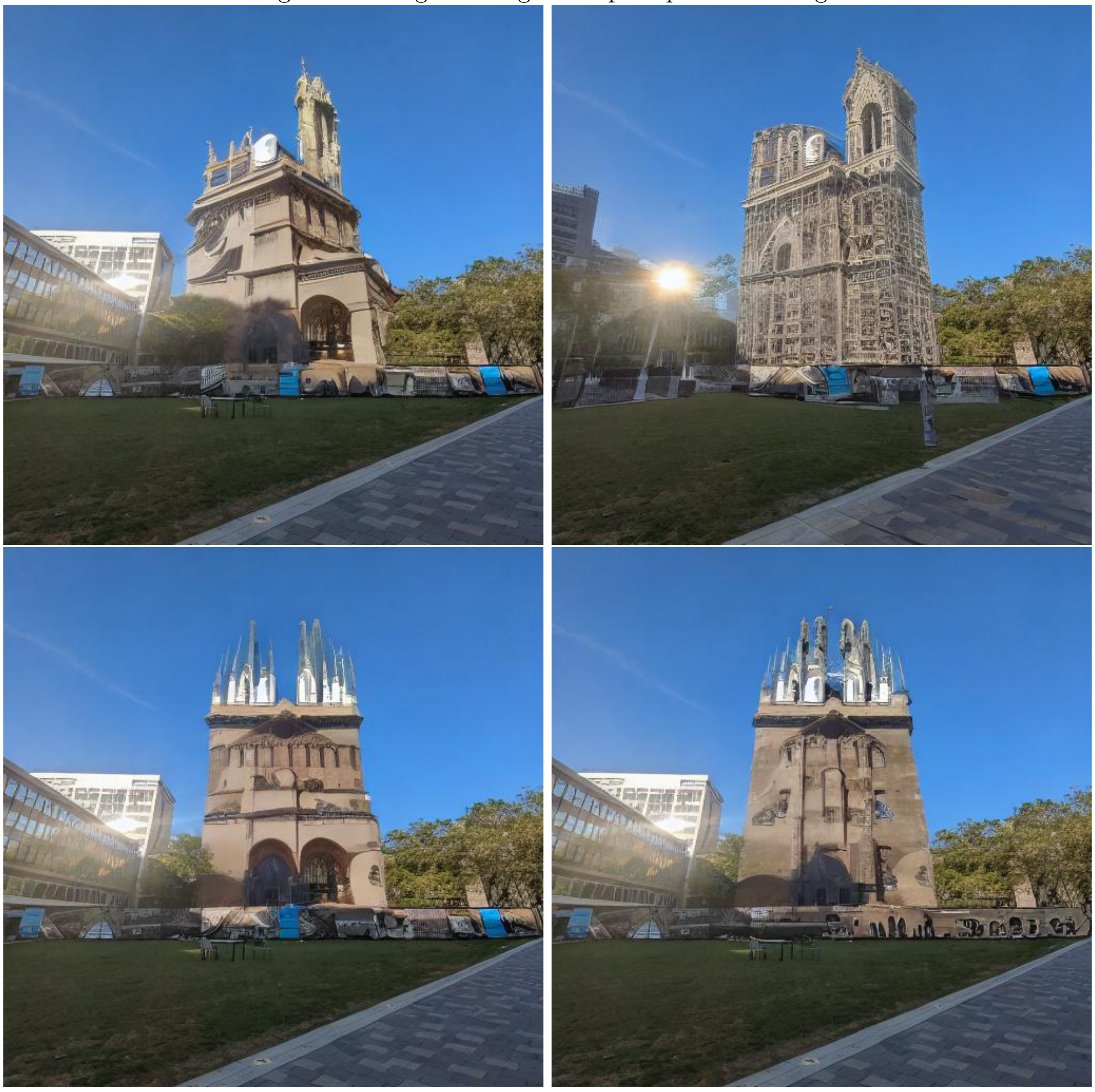


Figure 22: $\frac{\textbf{CLIPSeg}}{\textbf{SegCLIP}}\Big|\frac{\textbf{DiffEdit}}{\textbf{Segment Anything}}$

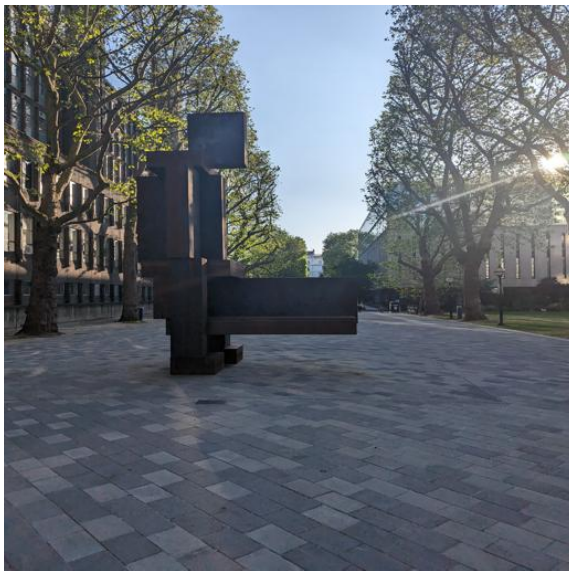


Figure 23: Original image with prompt ”sculpture”

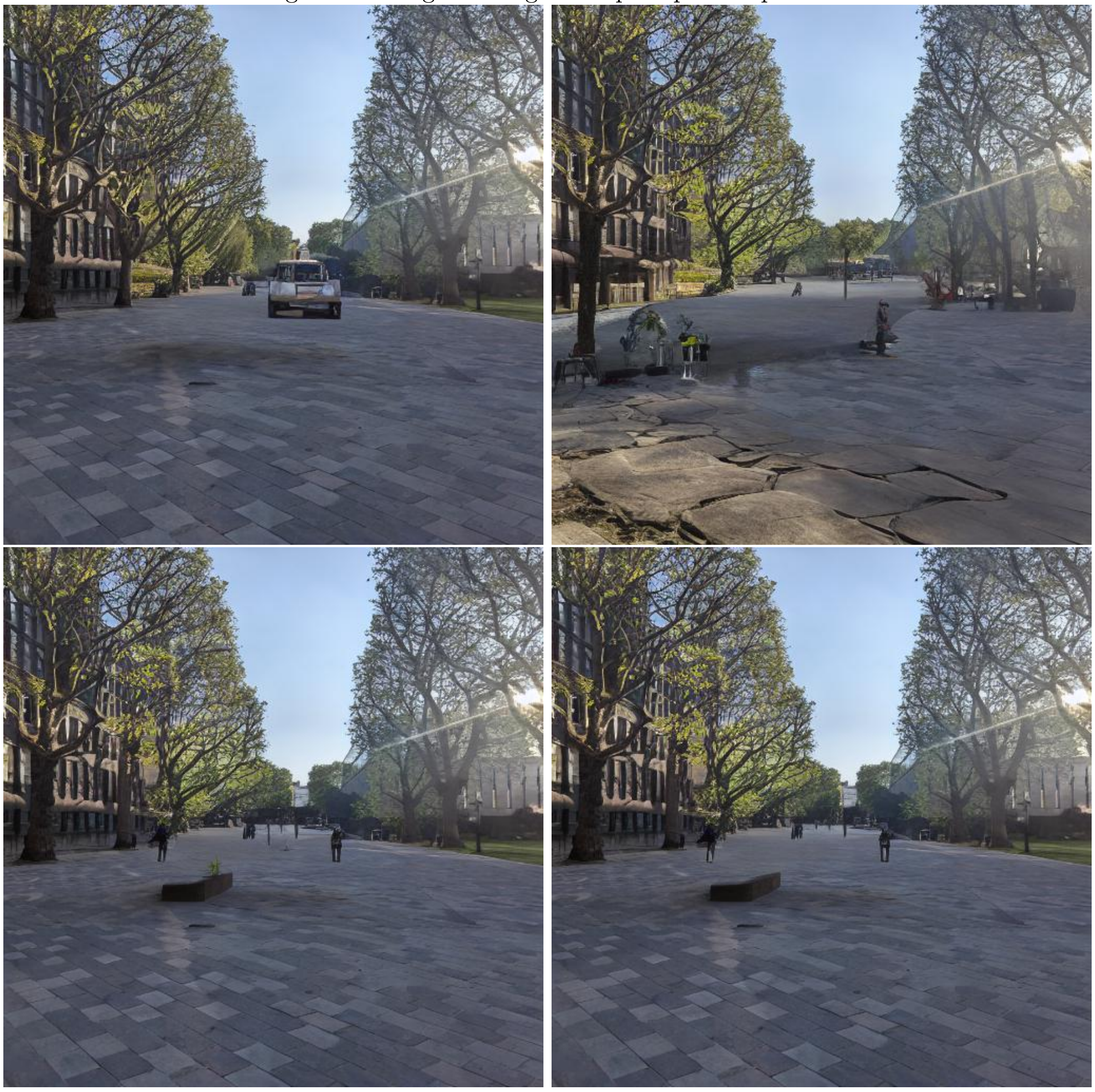


Figure 24: **CLIPSeg**|**DiffEdit**
SegCLIP|**Segment Anything**

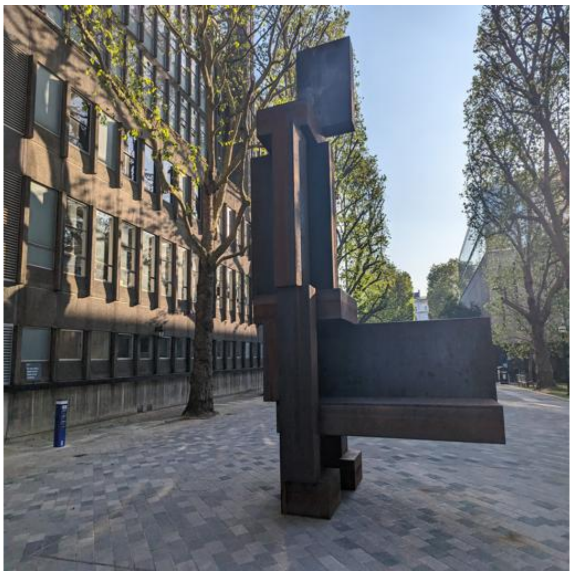


Figure 25: Original image with prompt "sculpture"

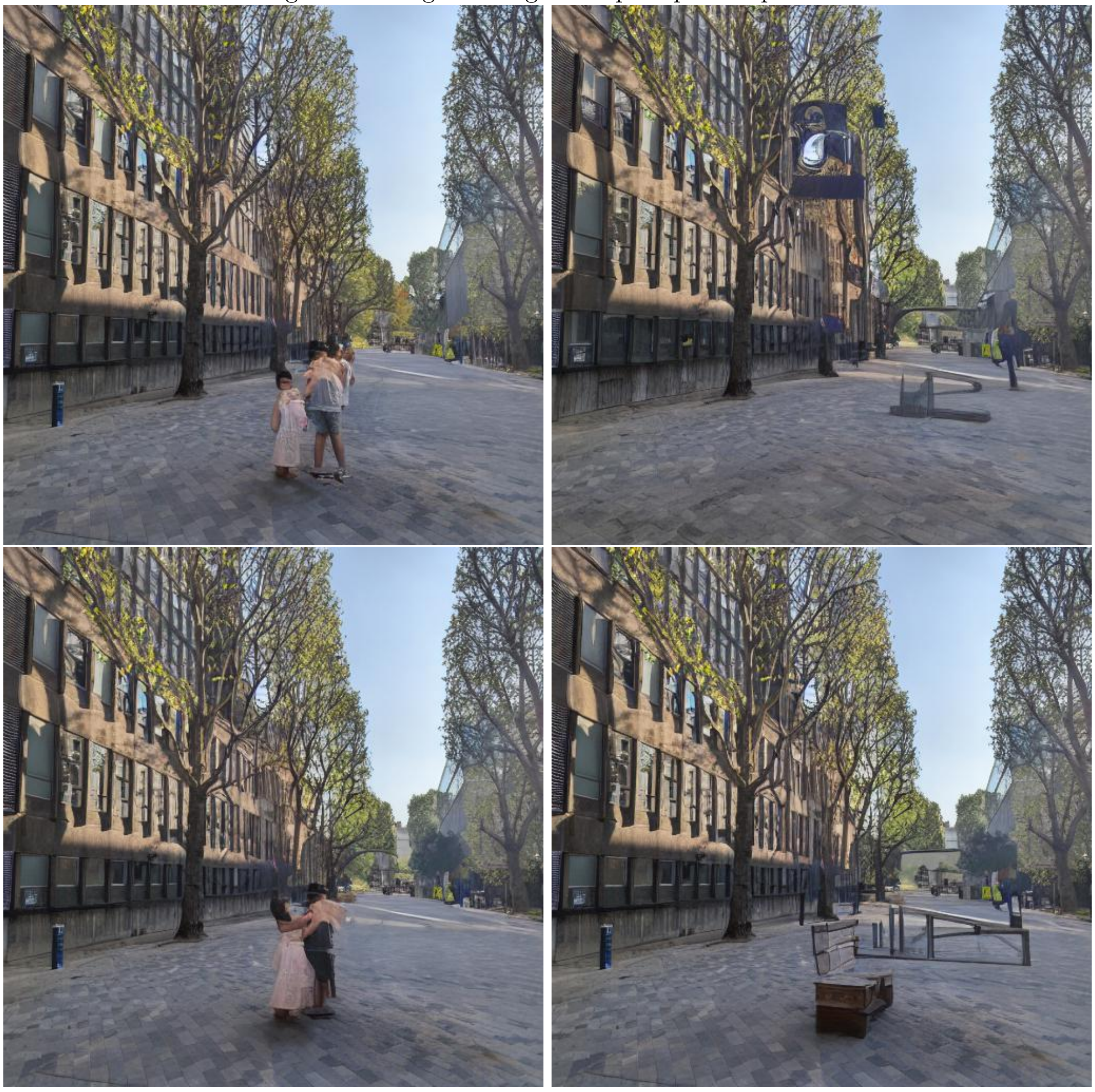


Figure 26: **CLIPSeg** | **DiffEdit**
SegCLIP | **Segment Anything**

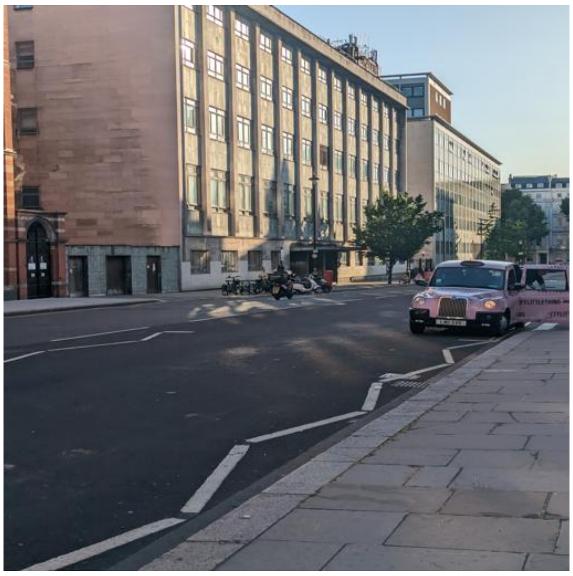


Figure 27: Original image with prompt ”pink taxi”

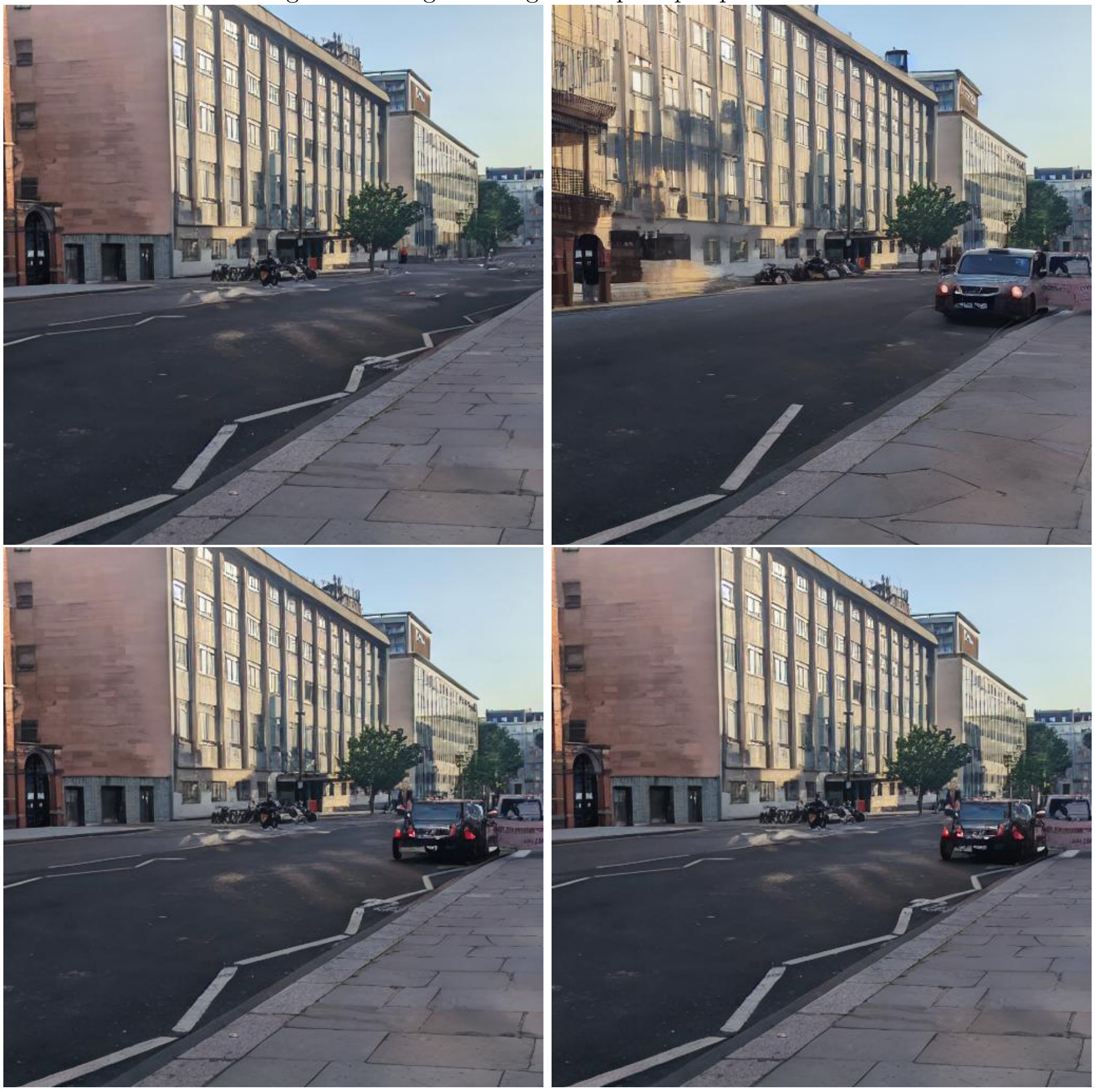


Figure 28: **CLIPSeg** | **DiffEdit** / **SegCLIP** | **Segment Anything**

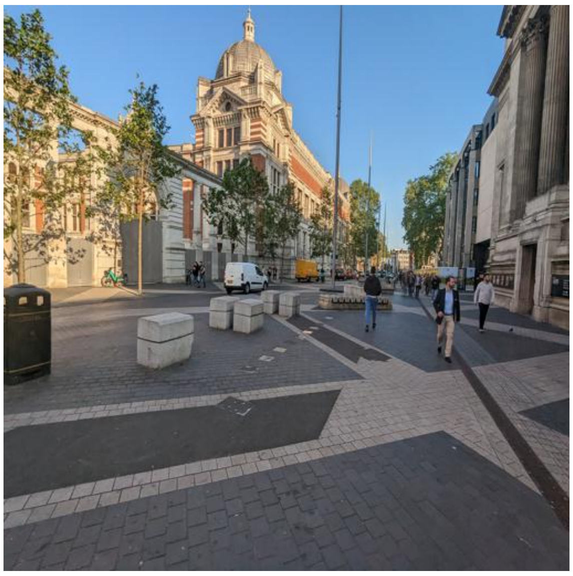


Figure 29: Original image with prompt "white van"

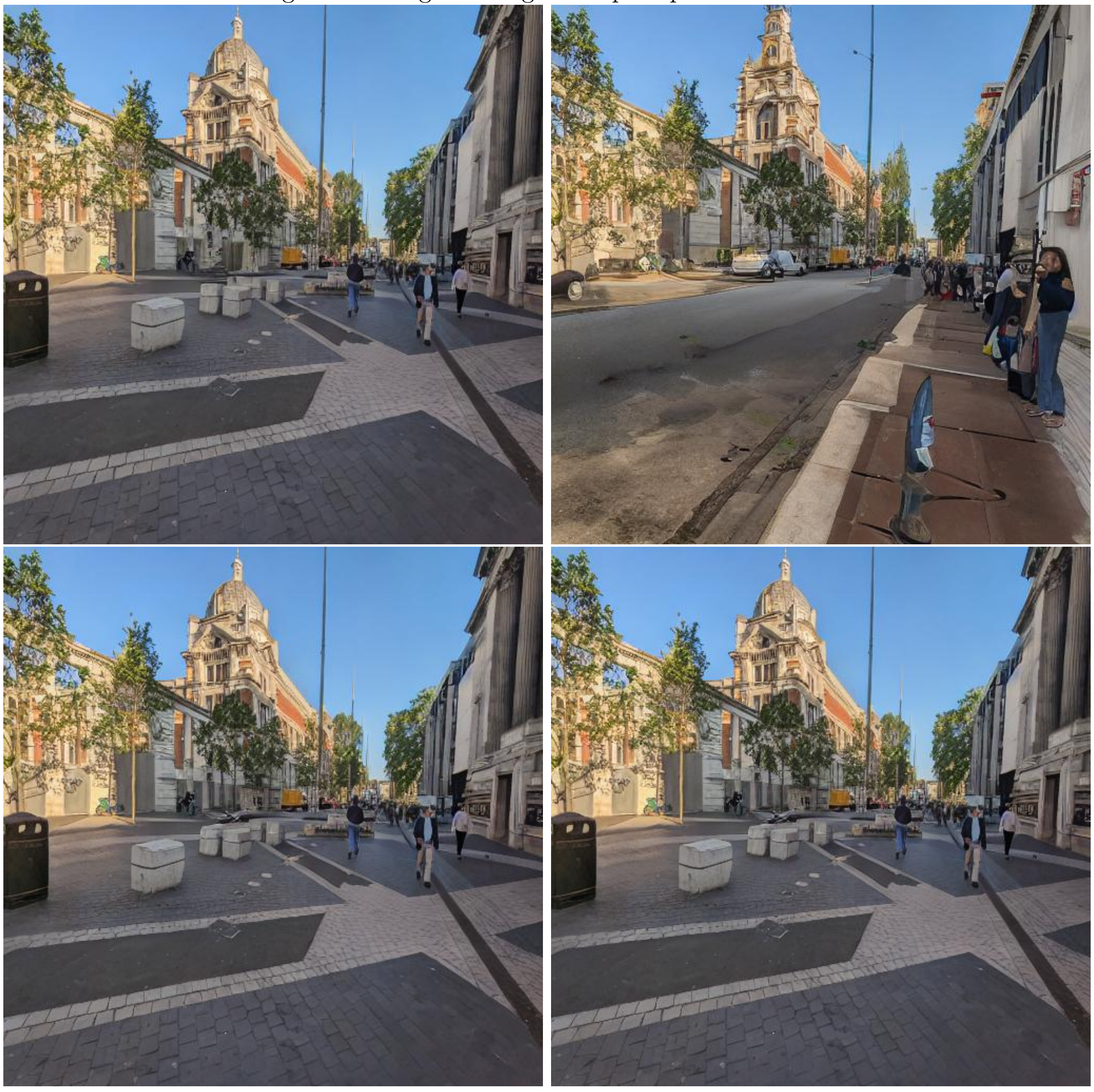


Figure 30: $\left.\begin{array}{c}\textbf{CLIPSeg}\\ \hline \textbf{SegCLIP}\end{array}\right|\begin{array}{c}\textbf{DiffEdit}\\ \hline \textbf{Segment Anything}\end{array}$

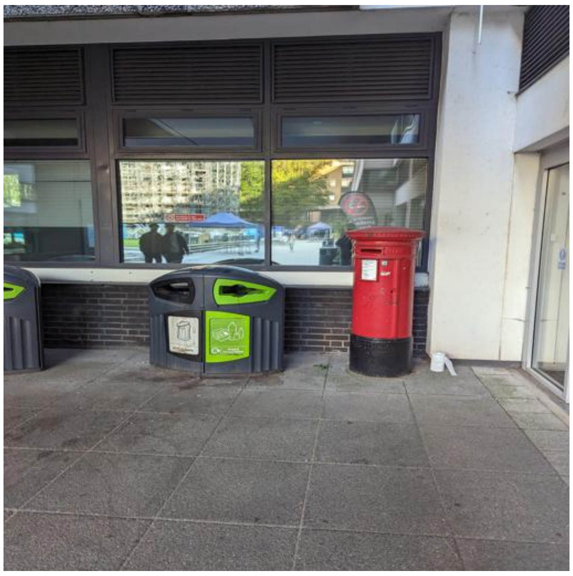


Figure 31: Original image with prompt ”green bin”

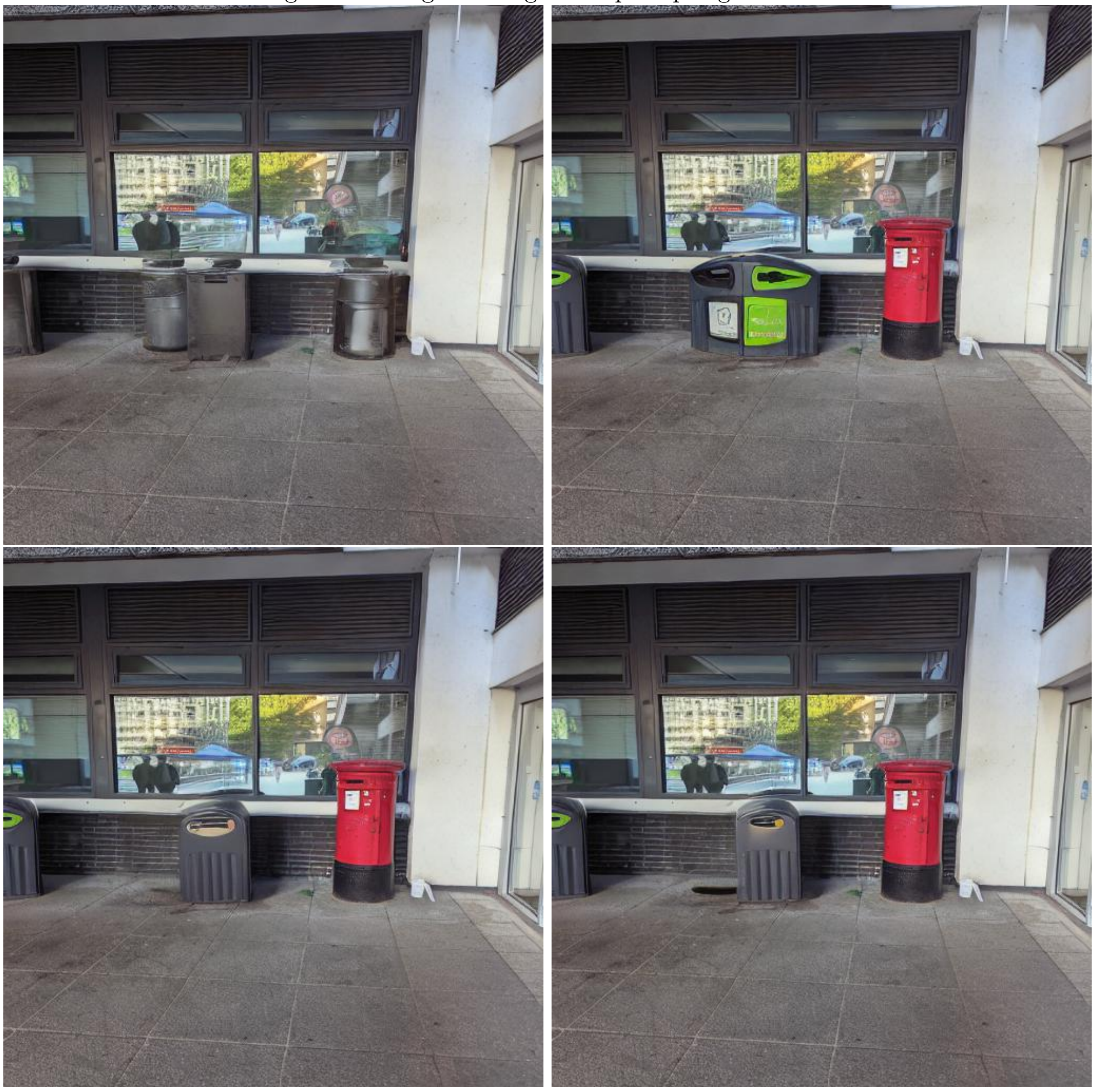


Figure 32: $\dfrac{\textbf{CLIPSeg}\,|\,\textbf{DiffEdit}}{\textbf{SegCLIP}\,|\,\textbf{Segment Anything}}$