Forget-Me-Not: Learning to Forget in Text-to-Image Diffusion Models

摘要

文本到图像生成模型在应用中的显著进展引发了对后处理适配算法的需求，这些算法可以高效地从预训练模型中移除不需要的概念（例如隐私、版权和安全问题），同时对预训练所学到的现有知识体系的影响最小。现有方法主要通过显式地将不需要的概念微调为某些替代项（如其上位词或反义词）。本质上，这些方法通过将不需要的概念替换为用户任意定义的内容来修改预训练模型的知识体系。此外，这些方法需要数百步的优化，因为它们完全依赖于预训练中使用的去噪损失。

为了解决上述挑战，我们提出了 Forget-Me-Not，一种以模型为中心的高效解决方案，能够在短至 30 秒的时间内，从配置良好的文本到图像模型中移除身份、物体或风格，而不会显著影响模型生成其他内容的能力。与现有方法不同，我们引入了注意力重新引导损失（attention re-steering loss），将模型的生成从不需要的概念重定向为预训练中学习的内容，而不是由用户定义的内容。此外，我们的方法提供了两个实用的扩展：a) 移除潜在有害或不适宜公开展示的内容（NSFW 内容）；b) 通过概念校正和解耦，增强模型的准确性、包容性和多样性。

关键词： 文本到图像生成模型；概念遗忘; 注意力重定向损失

1 引言

近年来，文本到图像生成模型（如 [6,15,30,32,33,36,43,44]）在根据文本描述生成高分辨率图像方面展现了非凡的能力。特别是扩散模型（如 DALL-E 2 [31] 和 Stable Diffusion [33]）已达到商业级产品化标准，为大量面向终端用户的应用铺平了道路。然而，这一领域的日益普及也带来了与安全性、公平性、法规合规、知识产权和整体安全相关的重要问题。学界面临诸多迫切挑战，例如模型可能意外生成未经授权、有偏见或潜在危险的内容。这并非前所未有的问题，学术界此前已尝试解决类似问题 [5,10]。

大规模文本到图像模型的内在风险主要源于训练过程中使用的大量数据集。这些数据集包括公共资源（如 LAION [37]、COYO [3]、CC12M [7]）和一些知名机构（如 Google [36,44] 和 OpenAI [31,32]）的专有数据。这些数据集带来了独特的挑战。公共数据集通常来源于网络爬取，可能缺乏严格的质量控制措施，尤其是在偏见和安全性方面。而专有数据集虽然可能更受控制，但由于标注成本高昂，其可扩展性受到限制。因此，仅通过数据过滤或溯源来缓解有害内容、隐私泄露和版权侵权问题，是一项艰巨的任务。

一种潜在的解决方案是领域适配（domain adaptation）[18, 42, 45]，但整理和优化此类数据集的复杂性非常高。此外，这种领域适配可能无意间降低模型的多样性，使其在生成领域外的图像时表现不佳。因此，能够让大规模文本到图像模型有选择性地“遗忘”特定概念的高效方法成为一个有前途的方向。

一些同期研究探索了通过训练模型将目标概念的预测重定向为某些替代概念的方法 [17, 23]。然而，这些方法本质上是重新训练模型，将目标概念替换为用户定义的概念，而不是让模型基于其内在知识自然生成适当的内容。此外，最近关于可控文本到图像合成的研究 [8, 19] 强调，交叉注意力条件（cross attention conditioning）是决定生成图像中主要概念的关键因素。这些见解促使我们探索基于交叉注意力机制的解决方案。

本文展示了以下内容：1. 在推理阶段将目标概念的交叉注意力分数归零的效果，我们观察到目标概念在生成图像中显著减少；2. 交叉注意力分数可以直接用作优化扩散模型的目标，以降低模型对目标概念的感知程度。据我们所知，我们是首次证明交叉注意力分数可以作为文本到图像模型微调的有效目标。

基于上述技术，我们提出了 Forget-Me-Not，一种高性价比的概念遗忘方法，使模型根据其知识决定生成内容，这是现有方法未能充分考虑的重要方面。特别是，我们展示了 Forget-Me-Not 能够取得与现有方法相当的结果，并在某些情况下表现更优。此外，它还促进了概念校正与解耦，提高了模型的精度和多样性。

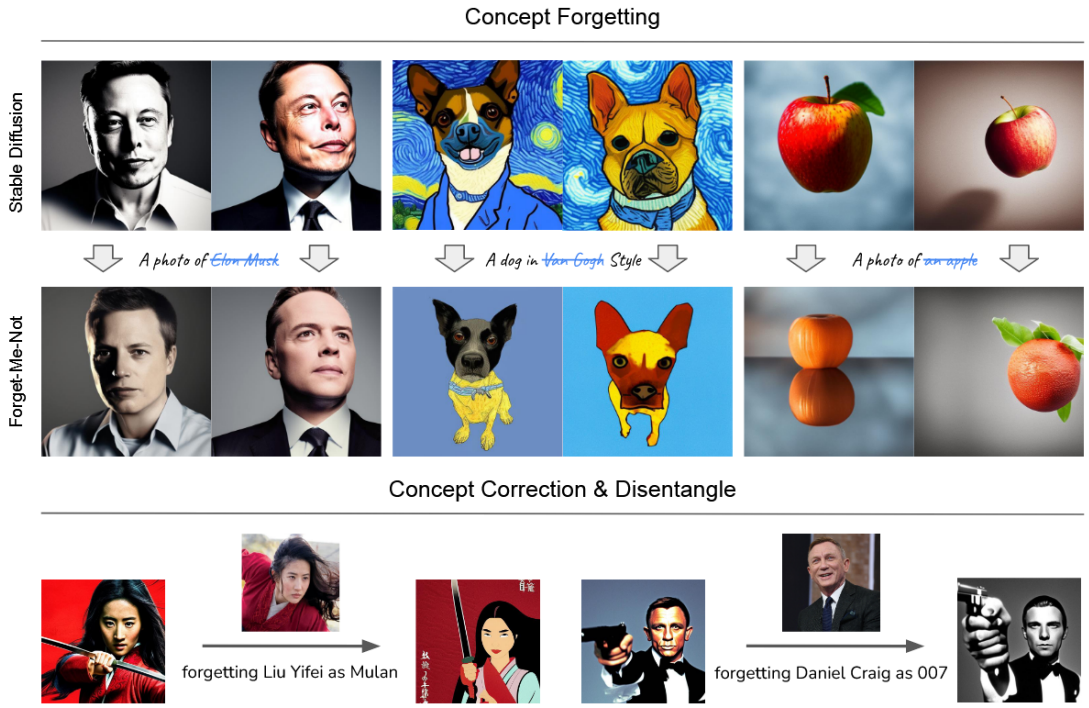

图 1. 概念遗忘：目标概念（用蓝色文本标出并加上删除线）成功地被移除，而不会影响输出质量。概念校正与解耦：我们的方法可用于校正提示中的主导概念或不希望出现的概念。当主导概念被遗忘后，之前被掩盖的概念会在输出中重新显现。

2 相关工作

2.1 文本到图像合成

在过去的十年里，文本到图像合成技术经历了从无条件生成模型到有条件生成模型的快速发展，采用了强大的架构，如自回归模型 [31,44]、生成对抗网络（GAN）[4,21,22,38,41] 和扩散过程 [2,14,20,26,29]。早期的研究主要集中在无条件的单类别数据分布建模，如手写数字、某些物种的动物和人类面孔 [9,12,21,25]。虽然无条件模型在单类别数据上快速取得了照片级逼真的结果，但当数据分布扩展到多类别或真实图像的多样性时，通常会发生模式崩塌问题 [1,4,27]。为了应对模型崩塌问题，引入了有条件生成模型。各种数据类型被用作生成模型的条件，例如类别标签、图像实例，甚至是网络 [4,28] 等。与此同时，CLIP [22,30] 作为一个大规模的预训练图像-文本对比模型，提供了极高多样性的文本-图像先验，已被证明可作为生成模型的条件 [11,24,31]。如今，DALL-E 2 [31] 和 Stable Diffusion [33] 可以仅凭自由形式的文本生成高质量图像。随后，一些研究致力于高效地将这些大规模生成模型适应于生成未见概念的全新版本，这些概念由一个小的参考集表示。Dreambooth [35] 通过微调所有模型权重来适应模型，但这种方法需要庞大的存储空间来保存新适应的权重。而 Textual Inversion [16] 和 LoRA [11] 则通过添加一小组额外的权重来改进这个问题，避免了对整个模型进行微调。

2.2 概念替换

先前的研究已经注意到，大型文本到图像模型可能会生成无意的偏见和不安全内容。为了解决这个问题，它们采用了去噪损失 [13] 来引导预测的噪声远离某一给定概念的原始表示。Kumari 等人 [23] 和 Heng 与 Soh [39] 选择用用户定义的概念来替换目标概念，通常是上位词。例如，微调模型将“一个生气的猫”替换为“猫”。SLD [33] 和 ESD [17] 则利用预训练模型的无分类器引导（classifier-free guidance）来引导预测朝着相反的方向进行。然而，这种方法存在一个风险，即可能会显著改变生成图像的语义，导致其偏离原始提示。

3 本文方法

3.1 前提知识

扩散模型 [13,20,29] 是去噪模型，通过迭代过程将数据 x_0 从高斯噪声污染 x_T 恢复，其中总步数为 T。这样的恢复过程通常称为逆扩散过程 $p_\theta(x_{t-1}|x_t)$，与之相对的是前向扩散过程，它将信号与噪声混合 $q(x_t|x_{t-1})$：

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{1}$$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2}$$

其中，$q(x_t|x_{t-1})$ 表示前向扩散过程，$p_\theta(x_{t-1}|x_t)$ 表示逆扩散过程。我们假设前向和逆过程都是马尔科夫链，因此可以将两者的似然表示为：

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{3}$$

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{4}$$

扩散过程的损失函数是最小化变分界限 L_{VLB} 的负对数似然 $p_\theta(x_0)$，即最大化 x_0 作为最终去噪结果的似然（来自参数为 θ 的模型）：

$$L_{\text{VLB}} = \mathbb{E}[\log p_\theta(x_0)] \leq \mathbb{E}_q \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} \right] \tag{5}$$

交叉注意力机制 [40] 广泛应用于生成模型作为条件技术 [31, 33, 36]。其目的是将条件信号的信息传递到生成模型的隐藏特征中。具体来说，隐藏特征 h 作为查询 Q，而条件信号作为键 K 和数值 V。假设 Q 和 K 具有相同的维度 d，更新后的隐藏特征 h_0 计算如下：

$$h_0 = h + \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V \tag{6}$$

在文本到图像生成模型中，h 代表视觉特征，而 V 代表条件文本特征。该方程表示视觉特征对文本特征 V 中封装的概念进行注意，并通过残差更新自己，其中 softmax(·) 计算所有概念的注意力得分。它本质上是所有封装概念的加权和，允许视觉表示得到精细化和更加贴合文本输入。

在实践中，操作某个概念的注意力得分将影响该概念在生成图像中的出现。如图 2 所示，将“Elon Musk”的注意力得分减少为零，会导致生成其他人的图像。我们认为，当“Elon Musk”被减弱时，生成的匿名人物正是模型自然回退时的结果。

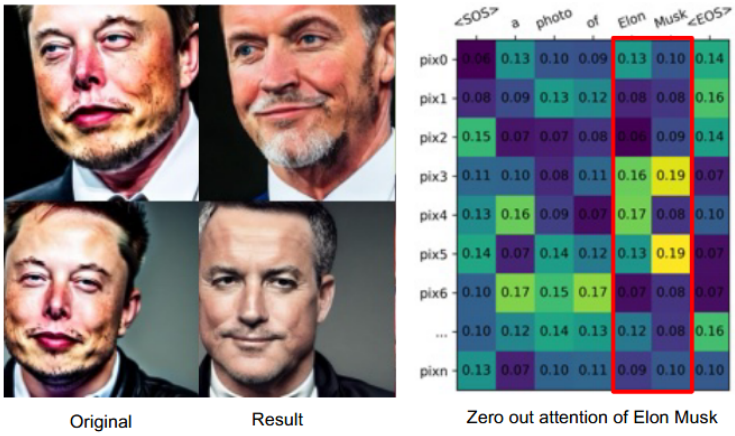

图 2. 当与“埃隆 · 马斯克”（红框标出）相关的注意力分数被设置为零时，在一个文本到图像生成模型中，像“埃隆 · 马斯克的照片”这样的提示会生成一张描绘非埃隆 · 马斯克人物的图像。

3.2 Forget-Me-Not

3.2.1 概念遗忘 (Concept Forgetting)

概念是表示思想对象的抽象概念，是人类感知和理解的基础。在文本到图像模型的背景下，这些概念嵌入在提示词的单词中。我们在此将概念遗忘定义为在文本到图像模型中，减

Algorithm 1 Attention Re-steering loss in training

Input: Textual embeddings C containing the target concept, indices I of the target concept, reference images R of the target concept, UNet U_0, denoising timestep T, total training step S.

repeat
$t \sim \text{Uniform}([1 \dots T])$; $e \sim N(0,1)$
$r_i \sim R_i$; $c_i, \text{idx}_{x_i} \sim C, I$
$x_0 \leftarrow \frac{r_i}{\overline{r_i}}$
$x_t \leftarrow \sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t} e$
$\triangleright \alpha_t$: noise variance schedule
$x_{t-1}, A_t \leftarrow U_0(x_t, c_j, t)$
$\triangleright A_t$: all attention maps
$\mathcal{L} \leftarrow \sum_{A \in A_t} \sum_{a \in A} [\text{idx}_{x_r}] \|a\|^2$
$\theta \leftarrow \theta - \nabla_\theta \mathcal{L}$
until S steps

弱目标概念与其预期视觉表现之间的强相关性的过程。这个定义避免了需要对模型进行微调以强行将目标概念替换为用户定义的替代概念的需求。相反，它允许模型在减弱了这种相关性后自主决定生成什么内容。通过这种方式，模型能够最大程度地保留其原有的生成能力。

为实现这些目标，我们开发了两个新的损失函数：注意力重新引导损失（Attention Re-steering loss）和视觉去噪损失（Visual Denoising loss）。此外，对于那些与概念关联的提示尚不明确的情况，我们采用概念反转（Concept Inversion）技术，该技术旨在直接从图像中提取概念的文本嵌入。

3.2.2 注意力重新引导损失 (Attention Re-steering Loss)

在交叉注意力讨论的基础上，我们有兴趣将目标概念的注意力分数作为损失，来优化 UNet [34] 的可训练参数 θ。UNet 是 Stable Diffusion [33] 的主干网络，包含一系列交替的卷积和交叉注意力层，分别执行去噪任务和条件化任务。层被分为下行、中间和上行块，其中视觉特征经历了维度的变化。总共有 16 个交叉注意力层，我们从每一层获取注意力分数。

设 $A_{n \times l}$ 为某个交叉注意力层的注意力图，它是公式 1 中的 $softmax(\cdot)$ 项。这里，n 是来自 Q 的视觉特征的数量，l 是来自 K 的文本标记的长度。与目标概念相关的文本标记的起始和结束索引记为 $[i, j]$。我们为特定的交叉注意力层计算损失：其中 $A[:, i:j]$ 索引所有从列 i 到 j 的分数。这对应于视觉特征对目标概念所付出的所有注意力。图 3 展示了注意力重新引导损失的思想，即最小化目标概念的注意力分数。这将生成过程从目标概念重定向到其不那么显著的替代概念。**算法 1** 详细说明了如何在训练循环中使用该损失。

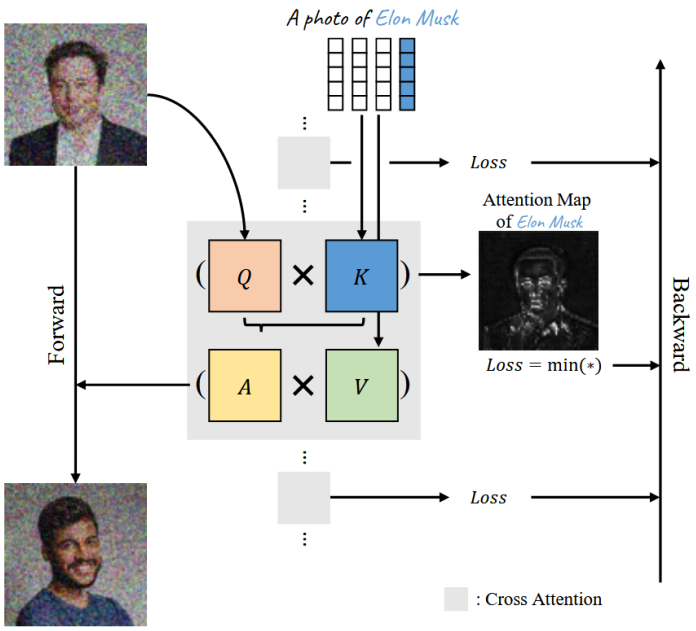

图 3. 这张图展示了我们在 Forget-Me-Not 方法中提出的注意力重定向（Attention Re-steering）技术，其中我们设置了目标函数以最小化目标概念（在此例中为 Elon Musk）的注意力图，并相应地对网络进行微调。

$$\mathcal{L}_{\text{Attn}} = \sum_{a \in A[:,i:j]} \|a\|^2 \tag{7}$$

3.2.3 视觉去噪损失 (Visual Denoising Loss)

如图 2 所示，在推理时操控注意力分数自然成为一种高效且低成本的概念遗忘数据生成技术。由于文本编码器中的自注意力机制，每个标记（token）与其他所有标记共享信息。因此，目标概念的注意力分数被置零不会完全丧失其语义。例如，将“埃隆 · 马斯克”的注意力分数置零，仍然会生成一张中年男性的图像，而不是女性。这一现象代表了我们所称的模型的“自然回退”现象。这启发我们创建合成数据作为概念遗忘的“真实数据”。

对于训练步骤 t，除了采样的噪声 ϵ 外，模型还接受一个生成图像 x_0（表示一个概念）及其对应的图像 $\tilde{x}_0$，其中对该概念的注意力已被置零。模型预测已添加到 x_0 中的高斯噪声 $\epsilon_\theta(x_t)$。随后，x_0 可以通过以下公式近似，如 [20] 中所描述：

$$x_0 \approx \tilde{x}_0 = \frac{x_t - \sqrt{1-\alpha_t}\epsilon_\theta(x_t)}{\sqrt{\alpha_t}} \tag{8}$$

$$Lvis = MSE(x\ 0, x0) L_{\text{vis}} = \text{MSE}(\tilde{x}_0, x_0) \tag{9}$$

其中，α_t 是预定义的噪声调度系数。与其计算预测的 $\epsilon_\theta(x_t)$ 和真实噪声 ϵ 之间的去噪损失，我们首先近似 $\tilde{x}_0$，并计算 $\tilde{x}_0$ 和 x_0 之间的均方误差（MSE）。我们称之为视觉去噪损失。

在实践中，我们发现这两个损失可以单独使用或一起使用。注意力重新引导损失提供了便利，因为它不需要生成配对的训练数据。另一方面，视觉去噪损失则更加专注于需要遗忘的主要概念。这是因为在配对的合成数据中，次要的图像结构较为相似，如图 2 所示。

3.2.4 概念反转 (Concept Inversion)

尽管我们通常知道目标概念的提示词，但也有例外情况，当使用文本提示词描述一个概念变得困难或不可能时。例如，忘记由一张图像表示的风格。为了解决这一挑战，我们将文本反转（Textual Inversion [16] 作为一个可选组件。在实践中，我们还注意到，这种反转有助于文本到图像模型更精确地识别需要遗忘的概念，从而提高其性能。相关结果可以在实验部分找到。

4 复现细节

4.1 与已有开源代码对比

4.1.1 引用代码使用情况

注意力重定向模块

注意力重定向模块 (Attention Re-steering Module) 主要用于引导模型的注意力集中在特定的概念或位置，以增强生成图像或文本的控制能力。它的核心目的是调整模型在自注意力机制中的注意力分配，从而确保模型在生成过程中更强地关注预定义的“概念”。

注意力重定向相关代码如下：

```
class AttnController:
    def __init__(self) -> None:
        self.attn_probs = []
        self.logs = []

    def __call__(self, attn_prob, m_name) -> Any:
        bs, _ = self.concept_positions.shape
        head_num = attn_prob.shape[0] // bs
        target_attns =
            attn_prob.masked_select(self.concept_positions[:,None,:].repeat(head_num,
            1, 1)).reshape(-1, self.concept_positions[0].sum())
        self.attn_probs.append(target_attns)
        self.logs.append(m_name)

    def set_concept_positions(self, concept_positions):
        self.concept_positions = concept_positions

    def loss(self):
        return torch.cat(self.attn_probs).norm()

    def zero_attn_probs(self):
        self.attn_probs = []
        self.logs = []
```

```
        self.concept_positions = None
```

在训练过程中，模块会接收一个 *concept_positions* 张量，这个张量表示需要重定向注意力的目标位置。这些位置通常对应输入数据中一些重要的部分，例如图像中的特定区域或者文本中的特定词汇。*concept_positions* 通常是一个布尔型张量，其中 True 表示该位置是我们希望模型关注的“概念”位置。

在每次自注意力计算中，模块会提取注意力矩阵（通常是形状为 $[num\_heads, seq\_len, seq\_len]$ 的张量），并通过 *concept_positions* 来筛选出与目标概念相关的注意力位置。*attn_prob* 是当前自注意力头计算出的注意力矩阵，*masked_select* 根据 *concept_positions* 的布尔值筛选出关注的部分，然后将其拉伸成一个扁平化的张量并保存。这些选出的注意力区域便是我们要强化关注的部分。在每个训练步骤结束时，模块通过计算所选注意力区域的 $L2$ 范数来评估注意力的集中程度。该损失函数鼓励模型将更多的注意力分配到关注的概念位置，从而实现“重定向”。损失越大，说明模型在这些目标位置上的注意力越集中。训练结束时，模块会清空先前保存的注意力信息，为下一轮训练做好准备。

视觉去噪损失函数

视觉去噪损失 (Visual Denoising Loss) 模块的作用主要是通过生成和对比去噪图像与目标图像之间的差异，来有效地“遗忘”目标概念的影响。这一过程利用了扩散模型的去噪机制，在训练过程中引导模型去除目标概念在图像中的表现，并专注于使图像的其他部分更加明显。视觉去噪损失模块的核心作用在于引导模型有效遗忘特定概念，并通过合成去噪数据来提升训练效率。该模块通过比较生成图像与去噪后的图像之间的差异，帮助模型专注于主要目标的遗忘，同时通过调整去噪和损失计算过程，保持图像的其他特征不变。这使得模型能够在图像生成过程中更好地实现目标概念的“遗忘”而不会影响图像的其他重要结构。

```python
import torch
import torch.nn.functional as F

class VisualDenoisingLoss:
    def __init__(self, noise_scheduler, tokenizer, device='cuda'):
        self.noise_scheduler = noise_scheduler # 用于调度噪声
        self.tokenizer = tokenizer
        self.device = device

    def compute(self, model, x0, target_concept_tokens, step,
        noise_type='gaussian', noise_scale=1.0):
        """
        计算视觉去噪损失 (Visual Denoising Loss)

        参数:
        - model: 用于去噪的模型 (UNet或者类似模型)
        - x0: 原始图像，包含目标概念的信息
        - target_concept_tokens: 目标概念的文本表示，例如['Elon Musk']
```

```
        - step: 当前训练步骤，通常用于获取噪声调度系数
        - noise_type: 噪声类型 ('gaussian' 或其他)
        - noise_scale: 噪声强度

        返回:
        - visual_denoising_loss: 计算得到的视觉去噪损失
        """

        # 获取去噪图像 x̃0: 零化目标概念的注意力
        x_tilde_0 = self._get_denoised_image(x0, target_concept_tokens)

        # 生成噪声，模拟在当前时间步的噪声
        noise = torch.randn_like(x0) * noise_scale

        # 计算模型的去噪输出: 预测的去噪图像 x̂0
        x_hat_0 = self._predict_denoised_image(model, x0, noise, step)

        # 计算均方误差 (MSE) 损失
        visual_denoising_loss = F.mse_loss(x_hat_0, x_tilde_0)

        return visual_denoising_loss

    def _get_denoised_image(self, x0, target_concept_tokens):
        """
        根据目标概念生成去噪图像 x̃0，去除目标概念的影响
        """
        # 创建目标概念的标记 ID
        concept_token_ids = self.tokenizer(target_concept_tokens,
            return_tensors='pt').input_ids.to(self.device)

        # 使用模型生成去噪图像: 零化目标概念的注意力
        x_tilde_0 = self._remove_concept_attention(x0, concept_token_ids)

        return x_tilde_0

    def _remove_concept_attention(self, x0, concept_token_ids):
        """
        通过零化目标概念的注意力，创建去噪版本 x̃0
        """
        # 这里实现零化注意力的逻辑
        # 假设 $x0$ 是模型输入的图像，而 $concept_token_ids$ 是目标概念的标记 ID
        # 你可以通过调整注意力矩阵来实现目标概念的消除，具体操作依据模型而定

```

```
        x_tilde_0 = x0.clone()

        # 假设我们用一个简单的掩码将目标概念的影响移除（可以依据需求进行更复杂的操作）
        for token_id in concept_token_ids:
            x_tilde_0 *= (1 - token_id) # 这里的代码逻辑需要根据具体模型修改

        return x_tilde_0

    def _predict_denoised_image(self, model, x0, noise, step):
        """
        通过模型的去噪过程，生成去噪图像 x̂0
        """
        # 模型的去噪过程。我们使用噪声调度器来获取正确的噪声系数
        noise_pred = model(x0 + noise, step) # 假设模型的输入是原始图像加上噪声

        # 使用噪声预测去噪图像
        x_hat_0 = x0 - noise_pred * self.noise_scheduler.get_alpha(step)

        return x_hat_0
```

compute 方法是主方法，用于计算视觉去噪损失。它接受原始图像 x_0 和目标概念的 *tokens* *target_concept_tokens*。然后生成去噪图像 x_0（通过 $remove_concept_attention$ 方法），并计算通过模型预测的去噪图像 x_0。最后，它使用均方误差（MSE）损失来计算视觉去噪损失。*get_denoised_image* 方法通过 *remove_concept_attention* 将目标概念从图像中去除，生成一个“去噪图像” x_0。*remove_concept_attention* 方法模拟了零化目标概念注意力的过程，从而生成去除目标概念影响的图像。在实际应用中，需要通过修改模型中的注意力层来实现这一点。*predict_denoised_image* 方法通过将噪声加入图像并预测去噪结果，模拟模型去噪过程。

根据使用的具体模型(例如 $U-Net$ 或类似的扩散模型)来修改 *remove_concept_attention* 和 *predict_denoised_image* 方法。这些方法的实现将依赖于如何在模型内部修改注意力权重，或者如何应用噪声和去噪过程。

这个视觉去噪损失模块的核心是模拟目标概念的去噪图像和去噪后的预测图像之间的误差。你可以根据模型和训练的需求进一步优化这个代码，实现更复杂的去噪效果。

虽然视觉去噪模块和注意力重定向模块是两个独立的部分，但它们在生成任务中互相补充和协作：

1) 视觉去噪模块 (Visual Denoising) 主要通过在图像中添加噪声并逐步去除，训练模型恢复图像中的有效特征，防止生成图像过于模糊或不清晰。

2) 注意力重定向模块 (Attention Re-steering) 则专注于调整模型的注意力，使其聚焦于图像中或文本中的特定概念区域。这能确保模型在生成过程中，能够更精确地控制哪些区域被关注，哪些区域被忽略，从而增强生成内容的相关性和准确性。

4.1.2 新增工作量

在原论文中，作者提出了几个评价指标用于量化模型的性能，特别是在概念遗忘和记忆评分方面。然而，在开源代码中，这些指标并没有直接实现。因此，我根据论文中的描述和方法，逐步实现了这些评价指标的代码，以便能够评估模型的遗忘效果和概念纠正的性能。以下是各个指标的含义和我在代码实现中的具体内容。

CLIP Score 是一种衡量生成图像与其文本提示之间一致性的指标。具体来说，它评估了生成的图像与其对应的文本提示在 CLIP (Contrastive Language-Image Pretraining) 模型中的相似度。CLIP Score 的值越高，表示生成图像与文本描述之间的匹配度越好。该指标用于评估概念遗忘的效果，特别是当模型忘记某一特定概念时，生成的图像是否仍然能够保留足够的语义一致性。CLIP Score 的下降表明模型有效地遗忘了目标概念，但如果 CLIP Score 下降过多，也可能意味着生成图像的语义信息丢失，从而影响生成图像的质量。因此，CLIP Score 用来确保图像在遗忘某些概念的同时，仍然能够维持其基本的语义结构。为了计算 CLIP Score，我们利用 CLIP 模型对生成图像和文本提示进行编码，并计算它们在嵌入空间中的余弦相似度。根据论文中的指示，我在代码中实现了这一过程。

```

'''
CLIP能力演示

1、对图片做分类
2、对图片求相图片

'''

import matplotlib.pyplot as plt
import torch
from clip import CLIP
import torch.nn.functional as F
from PIL import Image
from torchvision.transforms.v2 import PILToTensor
from torchvision.transforms import ToTensor, Compose, Grayscale

DEVICE = 'cuda' if torch.cuda.is_available() else 'cpu' # 设备

model = CLIP().to(DEVICE) # 模型
model.load_state_dict(torch.load('model.pth'))

model.eval() # 预测模式

transform = Compose([
    Grayscale(num_output_channels=1), # 转换为单通道灰度图像
    ToTensor(),
```

```
])

'''
2、图像相似度
'''
image1 = Image.open('/image/path/1/image.jpg')
image2 = Image.open('/image/path/2/image.jpg')

# 当前图片的向量
img_emb1 = model.img_enc(transform(image1).unsqueeze(0).to(DEVICE))
img_emb2 = model.img_enc(transform(image2).unsqueeze(0).to(DEVICE))

similarity = torch.matmul(img_emb1, img_emb2.T)
print(similarity)
```

Memorization Score 是一种用于评估模型“记忆”或“知识遗忘”程度的指标。与 CLIP Score 聚焦于图像和文本的一致性不同，Memorization Score 是从模型的角度出发，衡量概念遗忘前后模型对某一概念的记忆变化。具体来说，Memorization Score 通过对一个概念的探测集进行文本倒转操作，计算概念遗忘前后模型对该概念的理解变化。通过计算原始模型和遗忘模型在相同探测集上的文本倒转嵌入的 CLIP 相似度，我们可以量化模型“遗忘”某个概念的程度。

Memorization Score 通过比较遗忘前后的嵌入相似度来判断概念是否被成功遗忘。记忆评分的下降表明模型有效地遗忘了目标概念。在我的代码实现中，首先使用文本倒转技术将图像转化为 CLIP 空间的嵌入，然后计算遗忘前后的嵌入相似度变化，从而评估概念遗忘的效果。

我通过实现文本倒转的两次操作——一次使用原始模型，另一次使用经过概念遗忘的模型——来计算 Memorizaiton Score。在代码中，我实现了通过 CLIP 相似度计算两个嵌入之间的相似性变化，并使用这个变化来量化记忆评分的变化。

```
import torch
from PIL import Image
from transformers import CLIPProcessor, CLIPModel
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

# 加载原始模型和经过遗忘的模型
model_name = "/home/Forget-Me-Not-main/openai/clip-vit-large-patch14/"

# 原始模型
original_model = CLIPModel.from_pretrained(model_name)
original_processor = CLIPProcessor.from_pretrained(model_name)

# 经过遗忘的模型（假设已经加载到模型路径下）
```

```
forgotten_model =
    CLIPModel.from_pretrained("/home/Forget-Me-Not-main/exps_attn/elon-musk/" )
forgotten_processor =
    CLIPProcessor.from_pretrained("/home/Forget-Me-Not-main/exps_attn/elon-musk/")


# 计算 CLIP 特征
def get_clip_features(model, processor, image, text):
    inputs = processor(text=text, images=image, return_tensors="pt", padding=True)

    with torch.no_grad():
        # 获取图像和文本的特征向量
        image_features = model.get_image_features(inputs['pixel_values'])
        text_features = model.get_text_features(inputs['input_ids'])

    return image_features, text_features


# 计算 Memorization Score
def calculate_memorization_score(image_path, text, anchor_text):
    # 加载图像
    image = Image.open(image_path)

    # 使用原始模型获取图像和文本特征
    original_image_features, original_text_features =
        get_clip_features(original_model, original_processor, image, text)

    # 使用遗忘模型获取图像和文本特征
    forgotten_image_features, forgotten_text_features =
        get_clip_features(forgotten_model, forgotten_processor, image,
                                                                text)

    # 计算相似度: 原始模型与锚文本的相似度
    anchor_input = original_processor(text=anchor_text, return_tensors="pt",
        padding=True)
    with torch.no_grad():
        anchor_features = original_model.get_text_features(anchor_input['input_ids'])

    original_similarity = cosine_similarity(original_image_features.cpu().numpy(),
        anchor_features.cpu().numpy())
    forgotten_similarity =
        cosine_similarity(forgotten_image_features.cpu().numpy(),
        anchor_features.cpu().numpy())
```

```
    # 计算相似度变化
    similarity_change = original_similarity - forgotten_similarity

    return similarity_change


# 测试
image_path = "/home/Forget-Me-Not-main/data/elon-musk/elon_musk_0.jpg"
text = ["a photo of Elon Musk"] # 探测数据集中的文本描述
anchor_text = "Elon Musk" # 锚文本

memorization_score = calculate_memorization_score(image_path, text, anchor_text)
print("Memorization Score: ", memorization_score)
```

4.2 实验环境搭建

1. 创建与激活虚拟环境

首先，需要创建一个新的 Conda 虚拟环境，并激活它

```
conda create -n forget-me-not python=3.8
conda activate forget-me-not
```

2. 安装 PyTorch 和其他依赖项

接下来，需要安装适合 CUDA 11.6 的 PyTorch 版本及其相关库（‘torchvision‘ 和 ‘torchaudio‘）：

```
pip install torch==1.13.1+cu116 torchvision==0.14.1+cu116 torchaudio==0.13.1
    --extra-index-url https://download.pytorch.org/whl/cu116
```

然后，安装其他的项目依赖项：

```
pip install -r requirements.txt
```

在这一步，*requirements.txt* 文件中列出了其他所有必需的 Python 库，这些库可能包括数据处理、模型训练和评估等所需的依赖。

3. 训练过程

根据实验目的，我们将会进行两轮训练，分别是概念的反转训练（Textual Inversion，Ti）和基于注意力重定向的遗忘训练。文本反转（Textual Inversion，Ti）步骤用于训练一个概念的反向表示。根据说明文档，这一步是可选的，仅在 *attn.yaml* 文件中设置 $use_ti : true$ 时才

需要执行。如果使用该功能，需要运行如下命令：

```
python run.py configs/ti.yaml
```

这一步会加载配置文件 $configs/ti.yaml$，并根据该文件中的设置训练与指定概念相关的反向表示。通常，训练内容包括图像的逆向映射过程，这样可以让模型学习如何通过文本嵌入来表示一个具体的概念。完成文本反转训练后，下一步是使用注意力重定向方法进行概念遗忘。这一步的目的是通过控制注意力分布，让模型“遗忘”指定的概念，通常会影响模型的生成能力。运行如下命令进行遗忘训练：

```
python run.py configs/attn.yaml
```

这一步会加载配置文件 $configs/attn.yaml$，并根据该文件中的设置进行遗忘训练。在该过程中，模型会通过修改注意力机制来“遗忘”某个概念的影响。

4.3 创新点

4.3.1 现有方法不足

当前论文提出了一种基于交叉注意力机制的“人物遗忘”方法，虽然在准确性和生成质量上有较好的表现，但仍存在以下不足之处：**全局处理计算资源浪费**：现有方法在整个图像上计算交叉注意力，而人物的“遗忘”通常与人脸信息高度相关，对图像其他区域（如背景）的处理可能导致不必要的计算开销。**语义关注点分散**：模型在计算交叉注意力时会对全图特征进行均匀处理，这可能导致人脸区域的特征表达不足，影响人物遗忘任务的效率。**推理效率问题**：由于全图注意力的计算复杂度较高，推理阶段的效率可能受到影响，尤其在高分辨率图像场景下。

4.3.2 改进想法

为了解决上述问题，提出一种基于局部注意力优化的改进方法，核心思路如下：

1. 动态人脸蒙版生成：使用预训练的人脸检测器（如图 4 RetinaFace 或图 5 MTCNN），在训练和推理阶段动态生成人脸区域的蒙版，仅在人脸区域计算交叉注意力，减少计算开销。

2. 注意力权重调整：将全图交叉注意力改为分区域权重注意力机制：人脸区域赋予高权重，确保语义表达的精确性。非人脸区域赋予低权重或直接忽略，以提升计算效率。

3. 对比实验验证：在不同场景下（如单人图像、多人物场景、背景复杂的图像）进行实验对比，验证蒙版方法对模型性能（包括生成质量、训练时间和推理效率）的提升效果。

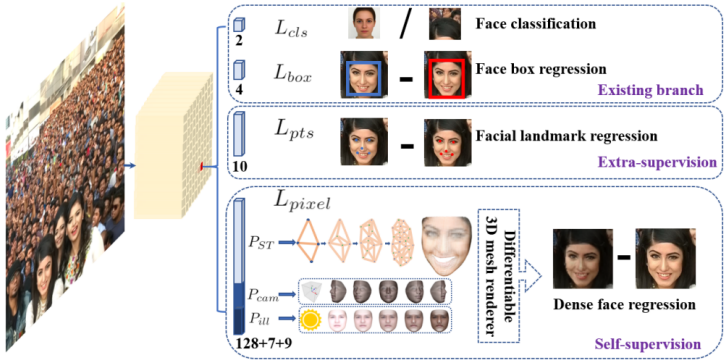

图 4. RetinaFace 是一种单阶段的人脸检测方法，能够在人脸检测过程中提供精确的面部关键点定位。该方法通过使用 Dense Prediction 模块，在不同尺度的特征图上预测人脸的关键点。RetinaFace 的优势在于，它不仅具有优秀的人脸检测精度，还能够在复杂场景下保持较高的稳定性。

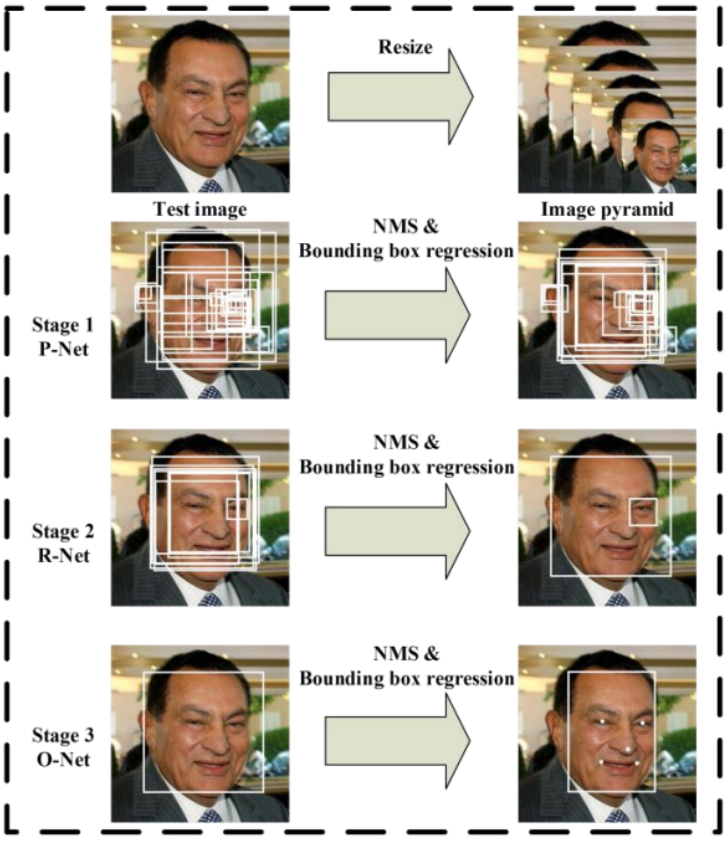

图 5. MTCNN 是一种端到端的网络模型，用于人脸检测和对齐。它结合了三个级联的卷积网络（分别用于候选框生成、边界框回归和人脸关键点检测）来实现准确的人脸检测和定位。MTCNN 的多任务训练使得其在人脸检测任务上表现良好，同时能够生成精确的人脸关键点。

4.3.3 改进方法的代码实现

以下是改进方法的代码框架，包括人脸蒙版生成和区域注意力优化两部分：

动态人脸蒙版生成

```python
# 初始化人脸检测器
mtcnn = MTCNN(keep_all=True, device='cuda' if torch.cuda.is_available() else 'cpu')

def generate_face_mask(image_path):
    """
    根据输入图像生成人脸蒙版
    :param image_path: 输入图像路径
    :return: 人脸蒙版，形状与原图一致
    """
    image = Image.open(image_path)
    boxes, _ = mtcnn.detect(image)

    mask = torch.zeros((image.size[1], image.size[0])) # 初始化蒙版
    if boxes is not None:
        for box in boxes:
            x1, y1, x2, y2 = map(int, box)
            mask[y1:y2, x1:x2] = 1 # 标记人脸区域为1
    return mask
```

局部交叉注意力实现

```python
class LocalCrossAttention(nn.Module):
    def __init__(self, embed_dim, num_heads):
        super(LocalCrossAttention, self).__init__()
        self.multihead_attn = nn.MultiheadAttention(embed_dim, num_heads)

    def forward(self, query, key, value, mask):
        """
        在局部区域内计算交叉注意力
        :param query: 查询特征
        :param key: 键特征
        :param value: 值特征
        :param mask: 人脸区域蒙版
        :return: 局部注意力结果
        """
        # 仅在人脸区域计算注意力
        mask = mask.flatten(start_dim=1).unsqueeze(1) # 展平并扩展维度
        attention_output, _ = self.multihead_attn(query, key, value, attn_mask=mask)
        return attention_output
```

集成到训练和推理流程

```python
def forward_pass(image, text_embedding, attention_module):
    """
    前向传播流程，集成人脸蒙版和局部交叉注意力
    :param image: 输入图像
    :param text_embedding: 文本嵌入向量
    :param attention_module: 局部交叉注意力模块
    """
    # 生成人脸蒙版
    face_mask = generate_face_mask(image)

    # 图像特征提取（假设已有预训练UNet模型）
    image_features = unet(image)

    # 局部交叉注意力
    output = attention_module(image_features, text_embedding, image_features,
        face_mask)
    return output
```

4.3.4 预期结果

a) 训练效率：减少无关区域的注意力计算，缩短训练时间; b) 推理速度：通过动态蒙版优化，推理阶段速度显著提升; c) 生成质量：集中在人脸区域的特征表达，提升模型在“人物遗忘”任务上的生成质量和准确性。

5 实验结果分析

5.1 概念遗忘（Concept Forgetting）

图 6 展示的是关于概念遗忘后的效果。最左边的 2x2 网格显示了来自 Stable Diffusion 的原始样本。接下来 3 张图展示了同一提示语下，原论文中遗忘后的生成样本。右边 4 张图片展示了复现的效果。前三行针对埃隆 · 马斯克（Elon Musk）和比尔 · 盖茨（Bill Gates），展示了多概念遗忘的能力以及控制概念能力，结果展示，提出的方法对非目标概念的影响较小。最后一行展示了单一概念风格的模型。图像提示语分别是：“X 的照片”和“X 风格的狗”（最后一行），实验结果表明，提出的方法不仅能忘掉特定人物，还能够忘掉相关的图像风格。

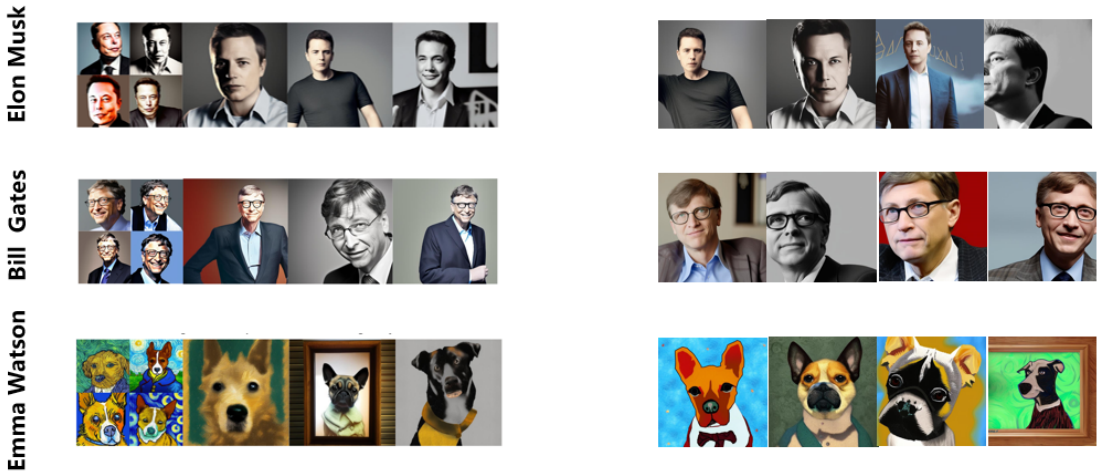

图 6. 概念遗忘实验结果

5.2 概念修正（Concept Correction）

文本到图像模型在训练过程中通常会根据大量的图像-文本示例优先考虑提示语的语义。这可能在推理时掩盖那些不常见的语义，如图 7 所示。例如，詹姆斯 · 邦德系列电影主要展示的是丹尼尔 · 克雷格（Daniel Craig）。花木兰系列电影主要展示的是刘亦菲。然而，我们的方法可以减少这种语义的主导地位，从而使其他詹姆斯 · 邦德的演员也能被展现出来。类似地，我们的方法在语义相互竞争的场景中也能有效地修正目标概念，正如在木兰（Mulan）系列和“苹果”这一术语的案例中所见。

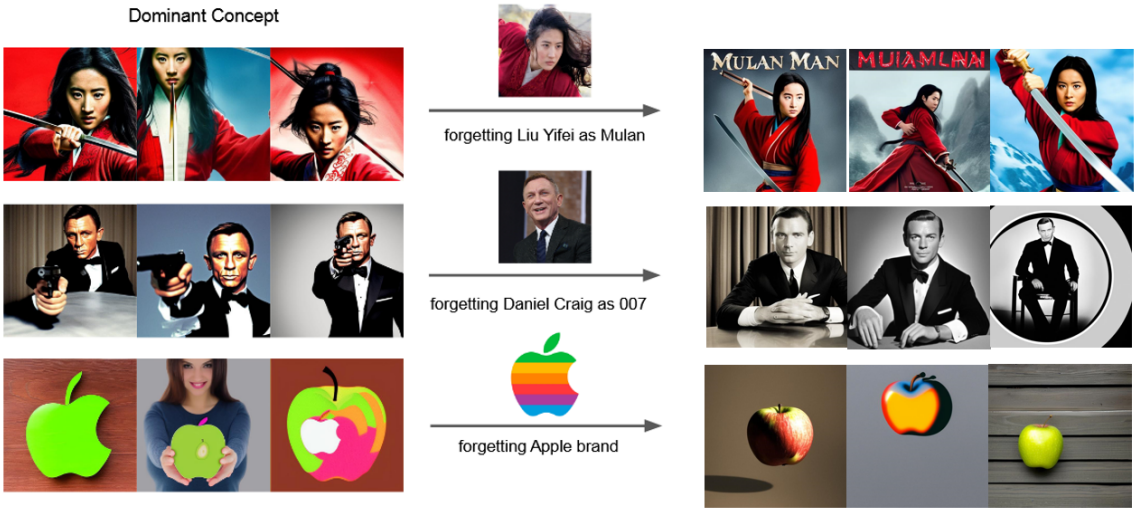

图 7. 概念修正实验结果

5.3 定量分析

CLIP Score 用于衡量生成图像与文本提示之间的一致性。它基于大规模预训练的图像 - 文本对比模型 CLIP 来计算。在文本到图像生成任务中，模型生成的图像应该与输入的文本

描述尽可能匹配。例如，当文本提示为 “a photo of an apple” 时，生成的图像如果确实是苹果的图像，那么 CLIP 分数会相对较高；反之，如果生成的图像与苹果相差甚远，CLIP 分数则会较低。在概念遗忘的情境下，当对某个目标概念（如 “corgi”）进行遗忘操作后，如果 CLIP 分数下降，说明模型在生成图像时对该目标概念的体现减少，即遗忘操作有效。但如果 CLIP 分数过低，可能意味着模型不仅遗忘了目标概念，还丢失了整体图像的语义信息，导致生成的图像与文本提示的相关性大幅降低，甚至生成了完全不相关的内容。通过表 1 可知，实验中，对不同概念如 “Elon Musk” “Mickey Mouse” 等进行遗忘操作后，其对应的 CLIP 分数都有一定程度的下降，表明模型在一定程度上减少了这些目标概念在生成图像中的呈现。

Memorization Score 提供了一个以模型为中心的视角来评估概念遗忘效果。它通过比较模型在概念遗忘前后对探测数据集的知识变化来衡量。首先，选取一个代表目标概念的小型探测图像集，然后利用文本反转技术将探测数据集转换为 CLIP 空间中的嵌入。这个过程会分别使用原始模型和经过概念遗忘操作后的模型进行两次转换。接着，选择一个能有效捕捉目标概念本质的锚提示（anchor prompt），计算锚提示与两组转换后的嵌入之间的 CLIP 相似度。例如，对于 “Elon Musk” 这个目标概念，以其作为锚提示，将探测图像通过原始模型和遗忘模型进行文本反转得到的嵌入分别与锚提示的嵌入计算相似度，即计算 $cos(emb_r, emb_o)$ 和 $cos(emb_r, emb_f)$，两者的差值就是 Memorization Score 的量化结果。如果经过概念遗忘后，探测集与锚提示之间的 CLIP 相似度降低，说明模型对该目标概念的记忆减少，遗忘操作有效。从表 1 中实验结果可知，如对 “Elon Musk” “Mickey Mouse” 等概念进行遗忘后，Memorization Score 确实有所下降，验证了模型在这些概念上的遗忘效果。

Concept	**MScore**		**CLIP**	
	Init.	**Frgt. ↓**	**Init.**	**Frgt. ↓**
Elon Musk	**0.943**	**0.902**	**0.308**	**0.283**
Mickey Mouse	**0.862**	**0.804**	**0.461**	**0.396**
Zebra	**0.981**	**0.879**	**0.312**	**0.310**
Google	**0.933**	**0.801**	**0.216**	**0.209**
Apple	**0.789**	**0.693**	**0.267**	**0.258**
Horse	**0.865**	**0.771**	**0.377**	**0.366**
Van Gogh	**0.880**	**0.604**	**0.304**	**0.221**

表 1. 各个概念的遗忘评估。表格展示了 Memorization Score 和 CLIP Score 在概念遗忘前后的初始值和遗忘值。箭头（↓）表示在概念遗忘后分数的下降，表明模型在多大程度上忘记了该概念。

6 总结与展望

本研究围绕文本到图像生成模型的概念遗忘展开，提出 Forget-Me-Not 方法并取得一定成果，但也存在可提升空间与待探索方向。

在概念遗忘方面，Forget-Me-Not 方法通过创新的注意力重新引导损失和视觉去噪损失函数，有效减少目标概念在生成图像中的呈现，同时较好地维持模型对其他内容的生成能力。如在多概念遗忘实验中，针对埃隆 · 马斯克和比尔 · 盖茨等人物概念以及不同风格概念的遗

忘操作，展现出对非目标概念较小的影响，证明了方法在多概念处理上的有效性。在概念修正上，模型能够突破主导语义的限制，挖掘被掩盖的次要概念，提升了模型语义处理的灵活性与准确性。通过 CLIP Score 和 Memorization Score 的定量评估，从图像 - 文本一致性和模型知识变化角度验证了遗忘效果。

然而，研究仍面临挑战。在处理复杂多概念遗忘场景时，随着遗忘概念数量增多，模型可能面临生成结果崩溃风险。这是因为过多概念遗忘可能破坏模型原有知识结构的平衡，使模型在生成图像时难以依据剩余知识生成合理内容。目前对于如何精准把握遗忘程度尚无成熟方法，若遗忘不足则无法有效去除不需要概念，若过度遗忘则可能引发模型生成异常。

未来研究可从以下方面深入：一是深入探究多概念遗忘的内在机制，通过理论分析与实验研究，明确不同概念组合及数量对模型结构和生成能力的影响规律，构建模型稳定性评估指标体系，提前预警生成结果崩溃风险；二是研发自适应遗忘程度控制策略，例如结合强化学习或元学习技术，使模型能根据输入概念特点和已有知识状态动态调整遗忘程度，在有效去除不需要概念的同时确保模型生成的可靠性与多样性；三是拓展研究到更广泛的模型架构和应用场景，验证方法的通用性和适应性，进一步推动文本到图像生成技术在安全、合规和高质量内容创作等方面的发展。

参考文献

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*, pages 214–223. PMLR, 2017.

[2] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

[3] Minwoo Byeon, Beomhee Park, Haecheon Kim, Sungjun Lee, Woonhyuk Baek, and Saehoon Kim. Coyo-700m: Image-text pair dataset. https://github.com/kakaobrain/coyo-dataset, 2022.

[4] Arantxa Casanova, Marlene Careil, Jakob Verbeek, Michal Drozdzal, and Adriana Romero Soriano. Instance-conditioned gan. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:27517–27529, 2021.

[5] Sungmin Cha, Sungjun Cho, Dasol Hwang, Honglak Lee, Taesup Moon, and Moontae Lee. Learning to unlearn: Instance-wise unlearning for pre-trained classifiers. *arXiv preprint arXiv:2301.11578*, 2023.

[6] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023.

[7] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3558–3568, 2021.

[8] Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *arXiv preprint arXiv:2301.13826*, 2023.

[9] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8188–8197, 2020.

[10] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. Zero-shot machine unlearning. *arXiv preprint arXiv:2201.05629*, 2022.

[11] Katherine Crowson, Stella Biderman, Daniel Kornis, Dashiell Stander, Eric Hallahan, Louis Castricato, and Edward Raff. Vqgan-clip: Open domain image generation and editing with natural language guidance. In *European Conference on Computer Vision (ECCV)*, pages 88–105. Springer, 2022.

[12] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[13] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:8780–8794, 2021.

[14] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Genie: Higher-order denoising diffusion solvers. *arXiv preprint arXiv:2210.05475*, 2022.

[15] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. In *European Conference on Computer Vision (ECCV)*, pages 89–106. Springer, 2022.

[16] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022.

[17] Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. *arXiv preprint arXiv:2303.07345*, 2023.

[18] Giorgio Giannone, Didrik Nielsen, and Ole Winther. Few-shot diffusion models. *arXiv preprint arXiv:2205.15463*, 2022.

[19] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6840–6851, 2020.

[21] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[22] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8110–8119, 2020.

[23] Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. *arXiv preprint arXiv:2212.04488*, 2022.

[24] Xihui Liu, Dong Huk Park, Samaneh Azadi, Gong Zhang, Arman Chopikyan, Yuxiao Hu, Humphrey Shi, Anna Rohrbach, and Trevor Darrell. More control for free! image synthesis with semantic diffusion guidance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 289–299, 2023.

[25] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15(2018):11, 2018.

[26] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *arXiv preprint arXiv:2206.00927*, 2022.

[27] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[28] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[29] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning (ICML)*, pages 8162–8171. PMLR, 2021.

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021.

[31] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[32] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning (ICML)*, pages 8821–8831. PMLR, 2021.

[33] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[35] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. 2022.

[36] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

[37] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[38] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4570–4580, 2019.

[39] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[41] Steven Walton, Ali Hassani, Xingqian Xu, Zhangyang Wang, and Humphrey Shi. Stylenat: Giving each head a new perspective. *arXiv preprint arXiv:2211.05770*, 2022.

[42] Jiayu Xiao, Liang Li, Chaofei Wang, Zheng-Jun Zha, and Qingming Huang. Few shot generative model adaption via relaxed spatial structural alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11204–11213, 2022.

[43] Xingqian Xu, Zhangyang Wang, Eric Zhang, Kai Wang, and Humphrey Shi. Versatile diffusion: Text, images and variations all in one diffusion model. *arXiv preprint arXiv:2211.08332*, 2022.

[44] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

[45] Jingyuan Zhu, Huimin Ma, Jiansheng Chen, and Jian Yuan. Few-shot image generation with diffusion models. *arXiv preprint arXiv:2211.03264*, 2022.