

A SUPPLEMENTARY MATERIALS

A.1 RELATED WORK

Generative Models. Generative models have gained significant attention in recent years Goodfellow et al. (2014); Radford et al. (2016); Arjovsky et al. (2017); Karras et al. (2019); van den Oord et al. (2016b;a); Salimans et al. (2017); Kingma & Welling (2014a); Rezende et al. (2014); Kingma & Welling (2014b); Ho et al. (2020); Song & Ermon (2019); Dhariwal & Nichol (2021); Song et al. (2020); Vaswani et al. (2017); Devlin et al. (2019); Brown et al. (2020); Xu et al. (2015); Tulyakov et al. (2018); Rombach et al. (2022). The core of generative models is to learn data distributions and generate similar samples. Early works on generative models, such as Gaussian Mixture Models (GMM) (McLachlan & Peel, 2000) and Hidden Markov Models (HMM) (Rabiner, 1989), provided simple probabilistic frameworks to model data distributions and capture basic statistical dependencies. Variational Autoencoders (VAE) (Kingma & Welling, 2014b) are considered the first combination of deep learning and generative modeling. VAEs encode input data into a latent space by learning a probabilistic distribution, then sample a latent variable from this distribution and decode it to reconstruct the input. The model optimizes a loss function that balances reconstruction accuracy and the regularization of the latent space to match a prior distribution (Kingma & Welling, 2014b). Generative Adversarial Networks (GANs) (Goodfellow et al., 2020) proposed a novel adversarial training framework consisting of a generator and a discriminator. The generator learns to produce realistic data from random noise, while the discriminator learns to distinguish between real and generated data. Through the adversarial training between the generator and the discriminator, GAN learns to produce increasingly realistic data.

Diffusion Models. Diffusion models are generative models that utilize a diffusion process during generation Nichol & Dhariwal (2021); Song et al. (2021); Austin et al. (2021); Li et al. (2022); Chen et al. (2023b; 2021c); Popov et al. (2021); Liu et al. (2023b); Lin et al. (2023); Karras et al. (2022); Lu et al. (2022); Song et al. (2023); Voleti et al. (2022); Zhang et al. (2023); Ho et al. (2020); Dhariwal & Nichol (2021); Rombach et al. (2022); Ramesh et al. (2022); Saharia et al. (2022); Song et al. (2020). Diffusion models were introduced by Sohl-Dickstein et al. (2015), who proposed a diffusion process that gradually adds noise to data and reverses it to generate samples, forming the foundation for subsequent advancements. Ho et al. refined this approach with Denoising Diffusion Probabilistic Models (DDPM), employing a step-by-step denoising process to generate high-quality images Ho et al. (2020). Song & Ermon introduced Score-Based Generative Models (SGMs), which used score functions and continuous diffusion to further improve sample quality Song & Ermon (2020). Dhariwal & Nichol advanced the field with Guided Diffusion, improving fidelity and diversity by conditioning the diffusion process on external data like class labels, making diffusion models competitive with GANs Dhariwal & Nichol (2021). Diffusion models have since expanded into new domains, such as audio generation, demonstrated by Kong et al. (2021), and video generation by Ho et al. (2022), showing their broad applicability across different data modalities.

Text-to-Image Diffusion Models. Recent advancements in text-to-image (T2I) diffusion models have significantly enhanced both generation efficiency and generated image quality Agarwal et al. (2025); Kim et al. (2025); Jha et al. (2025); Bai et al. (2025); Samuel et al. (2025); Balaji et al. (2022); Singer et al. (2023); Wu et al. (2023); Poole et al. (2023); Lin et al. (2023); Zhang et al. (2023); Brooks et al. (2022); Hertz et al. (2022); Blattmann et al. (2023); Bao et al. (2023); Kumari et al. (2023); Kavar et al. (2023); Chen et al. (2023a); Chefer et al. (2023); Ye et al. (2023); Zhao et al. (2023); Li et al. (2023b); Khachatryan et al. (2023); Feng et al. (2023); Xu et al. (2024); Shi et al. (2023); Wen et al. (2023); Fernandez et al. (2023); Avrahami et al. (2022); Kim et al. (2022); Mokady et al. (2022); Ramesh et al. (2021b); Saharia et al. (2022); Rombach et al. (2022); Nichol et al. (2022). Early notable contributions include GLIDE (Nichol et al., 2022), which introduced classifier-free guidance for generating photorealistic images from text, followed by DALL·E 2 (Ramesh et al., 2022), which improved text-image alignment by incorporating CLIP embeddings, and Imagen (Saharia et al., 2022), which achieved unprecedented realism by leveraging large pre-trained language models (Raffel et al., 2020) to guide the diffusion process. More recent breakthroughs, such as Stable Diffusion (Rombach et al., 2022), further optimized the generative process by introducing a more efficient architecture, allowing for high-quality image generation while reducing computational costs. DeepFloyd IF (StabilityAI, 2023) utilizes a cascaded diffusion model that progressively generates high-quality images in stages, each refining and increasing the resolution of the image. This cascading technique is designed to produce highly detailed and contextually accurate images from text prompts.

Exploiting T2I DMs for Advertisement Injection. Since T2I DMs generate images based on user prompts, they can be manipulated to generate images include specific patterns or objects. This vulnerability can be exploited to turn T2I DMs into tools for embedding advertisements. To the best of our knowledge, BAGM (Vice et al., 2024) is the first and only work that explicitly addresses this advertising scenario. BAGM proposes three approaches: surface, shallow, and deep attacks. The surface attack modifies user prompts by inserting brand-related words. For instance, if a user prompt contains the word “burger,” the attack appends the brand name “McDonald’s” before “burger.” The generated image will feature a McDonald’s burger to promote the brand. Note that the surface attack does not fit into our attack scenario since we assume the attacker cannot modify user prompts. The shallow and deep attacks in BAGM share a similar principle. They begin by selecting a trigger semantically related to the target brand, e.g., “burger” when advertising McDonald’s. BAGM collects images rich in McDonald’s elements from the internet and forms malicious text-image pairs by associating the trigger “burger” with McDonald’s images. Similar to backdoor attacks, the shallow attack leverages these malicious text-image pairs to fine-tune the text encoder, while the deep attack uses them to fine-tune the U-Net in the generative model. As a result, when the user’s prompt contains the trigger, the generated images tend to include elements associated with McDonald’s.

Backdoor Attack Against T2I Pipelines.

Previous works that introduce harmful information into T2I pipelines are similar to backdoor attacks in neural networks, where a selected *trigger* is injected into the T2I diffusion model through fine-tuning Nguyen & Tran (2020); Liu et al. (2020); Lin et al. (2020); Zhao et al. (2020); Wang et al. (2020); Xie et al. (2020); Bagdasaryan et al. (2020); Nguyen & Tran (2021); Doan et al. (2021); Li et al. (2021); Bagdasaryan & Shmatikov (2021); Wenger et al. (2021); Qi et al. (2021a;b); Shumailov et al. (2021); Pan et al. (2022); Souri et al. (2022); Doan et al. (2022); Wang et al. (2022); Salem et al. (2022). This results in adversarial behavior when trigger prompts are used, while performance on benign prompts remains largely unaffected. These backdoor attack methods on T2I DMs could potentially be repurposed to achieve the advertising objectives of our work, but none of these previous methods explicitly mention advertising as their goal. Several studies (Liu et al., 2023a; Struppek et al., 2023; Gao et al., 2023; Zhai et al., 2023) have explored creating *triggers* using unnatural inputs, such as replacing the letter ‘l’ with the number ‘1’ (Liu et al., 2023a), incorporating zero-width space characters (Zhai et al., 2023), replacing “red” to “read” (Gao et al., 2023), or use Cyrillic letters that are visually similar to English letters (Struppek et al., 2023). Although these works pioneered the exploration of adversarial triggers in T2I pipelines, the unnatural triggers they propose are less likely to appear naturally in typical user prompts. In contrast, other works (Vice et al., 2024; Yang et al., 2024) define *triggers* with natural language words and fine-tune the model to associate them with adversarial targets. For example, Vice et al. (2024) fine-tuned the word “drink” to associate with “Coca-Cola,” leading the T2I DM to preferentially generate Coca-Cola when the word “drink” appears in a prompt. Though not explicitly classified as backdoor attacks, methods like Ruiz et al. (2023); Gal et al. (2023) embed specific subjects into generated images upon detecting a trigger, achieving a similar effect.

Existing backdoor attacks cannot address the adversarial-advertisement setting in this paper. First, all the previous backdoor attacks or similar techniques rely on unnatural trigger tokens, such as typos, letter substitutions, and non-Latin characters, as specified in the previous paragraph. Benign users are very unlikely to include such triggers in their prompts. Consequently, the attack success rate in real-life scenarios could be low. Second, when a backdoor is triggered, the model should generate a pre-defined pattern that was embedded during the attack stage (e.g., a brand logo). Because this pattern is fixed and independent of the input prompt, the model largely ignores the prompt’s original semantics, resulting in images that deviate a lot from the user’s expectation. Since the attacker cannot assume future prompts, a trigger-based backdoor cannot adapt the advertisement to the prompt’s content and therefore cannot satisfy the adversarial-advertisement objective. To address both limitations, the method proposed in this work does not rely on explicit triggers and instead conditions the advertisement insertion on the prompt’s latent semantics, making the generated image align well with users’ intent while seamlessly embedding the target brand.

A.2 DISCUSSION ON THE ATTACK SCENARIO

An important question about our proposed attack scenario is why would users adopt the customized checkpoint instead of using the vanilla release. We first note that using customized diffusion

checkpoints is extremely common. Community hubs like HuggingFace, Civitai, and PixAI host hundreds of thousands of user-contributed models (Wei et al., 2024; Osborne et al., 2024). For example, Civitai had tens of millions of visits per month, and a search for “SD 1.5” yields thousands of user-generated checkpoints, often promoted for unique artistic styles. Popular projects like AnimateDiff and DreamBooth also rely on HuggingFace for distributing such models (Guo et al., 2024; Ruiz et al., 2023). Since the vanilla release lacks distinctive features (Zhang et al., 2023; Ruiz et al., 2023), users are highly motivated to interact with these customized checkpoints.

From an attacker’s view, community hubs are attractive. Uploaders can exaggerate or fabricate model performance; multiple studies show gaps between claimed and measured performance, and most platforms perform limited verification (Jiang et al., 2023; Kadasi et al., 2025). Uploading is free, enabling repeated reposting under different accounts. Prior work even found clusters of near-duplicate malicious checkpoints on HuggingFace, suggesting deliberate large-scale seeding (Zhao et al., 2024a). Given high user traffic, model diversity, and weak security, community hubs pose non-trivial supply-chain risks (Trend Micro Research, 2025; Yuan et al., 2025).

A substantial body of marketing research indicates that firms often prioritize awareness and talkability over sentiment, making unconventional campaigns practically plausible. First, a long-standing phenomenon, “all publicity is good publicity,” is widely discussed and supported in the literature, which argues that any exposure can be beneficial by increasing presence and visibility (Pacis et al., 2022). Second, studies have shown that many companies actively adopt non-traditional advertising strategies; for example, firms have achieved significant publicity through cost-effective campaigns that prioritize exposure (Waller, 2006). Third, even non-positive publicity can still be beneficial: Berger et al. (2010) provide empirical evidence that less favorable reviews can increase sales for lesser-known authors. This finding is consistent with eye-tracking evidence that negative comments attract greater attention and relate to purchase intention (Chen et al., 2022). These works further provide real-world motivation for businesses to single-mindedly pursue increased exposure. Taken together, these findings suggest that when the primary objective is exposure, firms are willing to adopt attention-maximizing tactics. Therefore, they support the plausibility that advertisers would employ adversarial advertisement strategies to increase brand visibility.

A.3 PROOF OF THEOREMS

Theorem 3.5. *Given sufficient iterations \mathcal{J} , our estimation $Q_A(x) = 1 - \bar{F}_A(x_1, \dots, x_d) = 1 - \alpha e^{\eta \mathbf{B} \sqrt{x}} \mathbf{D}_A \mathbf{1}$ for the multivariate continuously scaled phase-type with Lévy distribution will converge to the empirical distribution $P_A(x)$ estimated from real data.*

Proof. Let x_i represent the value of x at the i -th iteration out of a total of \mathcal{J} iterations, and define the empirical distribution $P_A(x) = \frac{\#(\mathbf{X} \leq [x_i, \dots, x_i])}{N^{d+1}}$, where N is the number of embeddings. The expectation of the distribution $\mathbb{E}(\mathbf{X} \leq [x_i, \dots, x_i])$ is given by:

$$\begin{aligned} \mathbb{E}(\mathbf{X} \leq [x_i, \dots, x_i]) &= \int_0^\infty 1 - Q_A(x) dx \\ &= \int_0^\infty \bar{F}_A(x_1, \dots, x_d) dx \\ &= \int_0^\infty \alpha_i \exp(\eta_i \mathbf{B}_i \sqrt{x}) \mathbf{D}_i \mathbf{A}_i \mathbf{1} dx \end{aligned} \tag{14}$$

Let $y = \sqrt{x}$, then $dx = 2y dy$. Using integration by parts formula, the integral part becomes:

$$\begin{aligned} \int_0^\infty \alpha_i \exp(\eta_i \mathbf{B}_i \sqrt{x}) \mathbf{D}_i \mathbf{A}_i \mathbf{1} dx &= 2 \int_0^\infty y \alpha_i \exp(\eta_i \mathbf{B}_i y) \mathbf{D}_i \mathbf{A}_i \mathbf{1} dy \\ &= -2\alpha_i \int_0^\infty \exp(\eta_i \mathbf{B}_i y) \mathbf{D}_i \mathbf{A}_i \mathbf{1} dy \end{aligned} \tag{15}$$

Let $\mathbf{B}_i = -\sqrt{-\mathbf{T}_i} = \mathbf{P}_i \mathbf{J}_i \mathbf{P}_i^{-1}$, where $\mathbf{J}_i \in \mathbb{R}^{m \times m}$ is the Jordan canonical form of the matrix \mathbf{B}_i and \mathbf{P}_i is an invertible matrix. The Jordan canonical form \mathbf{J}_i is composed of Jordan blocks, which

are of the form:

$$\mathbf{J}_i = \begin{pmatrix} J_1 & & \\ & \ddots & \\ & & J_{ij} \end{pmatrix} \quad (16)$$

Each Jordan block J_{ij} is of the form:

$$J_{ij} = \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix} \quad (17)$$

where λ_i is an eigenvalue of matrix \mathbf{B}_i . Then, $\exp(\eta_i \mathbf{B}_i y) = \mathbf{P}_i \exp(\eta_i \mathbf{J}_i y) \mathbf{P}_i^{-1}$. We can compute the integral of each Jordan block J_{ij} :

$$\int_0^\infty \exp(\eta_i \lambda_i y) \begin{pmatrix} 1 & \eta_i y & \frac{(\eta_i y)^2}{2!} & \cdots & \frac{(\eta_i y)^{m-1}}{(m-1)!} \\ & 1 & \eta_i y & \cdots & \frac{(\eta_i y)^{m-2}}{(m-2)!} \\ & & \ddots & \ddots & \vdots \\ & & & 1 & \eta_i y \\ & & & & 1 \end{pmatrix} dy \quad (18)$$

For the diagonal elements:

$$\int_0^\infty \exp(\eta_i \lambda_i y) dy = \frac{1}{-\eta_i \lambda_i} \quad (19)$$

For the off-diagonal elements that involve terms like $\eta_i y, \eta_i^2 y^2$, etc., the integrals of the form:

$$\int_0^\infty y^k \exp(\eta_i \lambda_i y) dy \quad (20)$$

These integrals can be computed using the Gamma function. For example:

$$\int_0^\infty y^k \exp(\eta_i \lambda_i y) dy = \frac{k!}{(-\eta_i \lambda_i)^{k+1}} \quad (21)$$

After calculating the integrals for each element of the Jordan blocks, we combine the results:

$$\int_0^\infty \exp(\eta_i \mathbf{B}_i y) dy = \mathbf{P}_i \int_0^\infty \exp(\eta_i \mathbf{J}_i y) dy \mathbf{P}_i^{-1} \quad (22)$$

Thus, the result of the integral and expected value is:

$$\mathbb{E}(\mathbf{X} \leq [x_i, \dots, x_i]) = -2\alpha_i \mathbf{P}_i \begin{pmatrix} \frac{1}{-\eta_i \lambda_i} & \frac{\eta_i}{(-\eta_i \lambda_i)^2} & \cdots & \frac{(\eta_i)^{m-1}}{(-\eta_i \lambda_i)^m} \\ & \frac{1}{-\eta_i \lambda_i} & \cdots & \frac{(\eta_i)^{m-1}}{(-\eta_i \lambda_i)^m} \\ & & \ddots & \vdots \\ & & & \frac{1}{-\eta_i \lambda_m} \end{pmatrix} \mathbf{P}_i^{-1} \mathbf{D}_i \mathcal{A}_i \mathbf{1} \quad (23)$$

where each block in the diagonal corresponds to the contribution from a Jordan block, with terms involving λ_i and powers of η_i .

Similarly, we can derive the variance of the distribution, $\mathbb{V}(\mathbf{X} \leq [x_i, \dots, x_i])$, as follows:

$$\begin{aligned} \mathbb{V}(\mathbf{X} \leq [x_i, \dots, x_i]) &= \mathbb{E}[\mathbf{X}^2] - (\mathbb{E}[\mathbf{X}])^2 \\ &= \int_0^\infty 2x (1 - F_S(x_1, \dots, x_d)) dx - \left(\int_0^\infty \bar{F}_A(x_1, \dots, x_d) dx \right)^2 \end{aligned} \quad (24)$$

where

$$\begin{aligned}
\mathbb{E}[\mathbf{X}^2] &= \int_0^\infty 2x (1 - F_S(x_1, \dots, x_d)) dx \\
&= 2 \int_0^\infty x (\alpha_i \exp(\eta_i \mathbf{B}_i \sqrt{x}) \mathbf{D}_i \mathcal{A}_i \mathbf{1}) \\
&= 2x \alpha_i \exp(\eta_i \mathbf{B}_i x) \mathbf{D}_i \mathcal{A}_i \mathbf{1} \Big|_0^\infty - 2 \int_0^\infty \alpha_i \exp(\eta_i \mathbf{B}_i x) \mathbf{D}_i \mathcal{A}_i \mathbf{1} dx \\
&= 4\alpha_i \mathbf{P}_i \begin{pmatrix} \frac{1}{-\eta_i \lambda_i} & \frac{\eta_i}{(-\eta_i \lambda_i)^2} & \cdots & \frac{(\eta_i)^{m-1}}{(-\eta_i \lambda_i)^m} \\ & \frac{1}{-\eta_i \lambda_2} & \cdots & \frac{(\eta_i)^{m-1}}{(-\eta_i \lambda_i)^m} \\ & & \ddots & \vdots \\ & & & \frac{1}{-\eta_i \lambda_m} \end{pmatrix} \mathbf{P}_i^{-1} \mathbf{D}_i \mathcal{A}_i \mathbf{1}
\end{aligned} \tag{25}$$

For those samples \mathbf{X} satisfying $\mathbf{X} \leq [x_i, \dots, x_i]$, we can compute the corresponding expectation $\bar{\mathbf{X}} = \mathbb{E}(\mathbf{X} \mid \mathbf{X} \leq [x_i, \dots, x_i])$ and variance $\sigma_{\bar{\mathbf{X}}}^2 = \mathbb{V}(\mathbf{X} \mid \mathbf{X} \leq [x_i, \dots, x_i])$.

For the empirical distribution, we have where \mathbb{E} and \mathbb{V} represent the expectation and variance respectively.

$$\mathbb{E}(\mathbf{X} \leq [x_i, \dots, x_i]) = -2\alpha_t P_t \begin{pmatrix} \frac{1}{-\eta_t \lambda_t} & \frac{\eta_t}{(-\eta_t \lambda_t)^2} & \cdots & \frac{(\eta_t)^{m-1}}{(-\eta_t \lambda_t)^m} \\ & \frac{1}{-\eta_t \lambda_t} & \cdots & \frac{(\eta_t)^{m-1}}{(-\eta_t \lambda_t)^m} \\ & & \ddots & \vdots \\ & & & \frac{1}{-\eta_t \lambda_k} \end{pmatrix} P_t^{-1} \mathbf{D}_t \mathcal{A}_t \mathbf{1}, \tag{26}$$

$$\mathbb{V}(\mathbf{X} \leq [x_i, \dots, x_i]) = -4\alpha P \begin{pmatrix} \frac{1}{-\eta_t \lambda_t} & \frac{\eta_t}{(-\eta_t \lambda_t)^2} & \cdots & \frac{(\eta_t)^{m-1}}{(-\eta_t \lambda_t)^m} \\ & \frac{1}{-\eta_t \lambda_t} & \cdots & \frac{(\eta_t)^{m-1}}{(-\eta_t \lambda_t)^m} \\ & & \ddots & \vdots \\ & & & \frac{1}{-\eta_t \lambda_t} \end{pmatrix} P_t^{-1} \mathbf{D}_t \mathcal{A}_t \mathbf{1}. \tag{27}$$

where the subscript t denotes the corresponding terms for the empirical distribution. Since $\bar{\mathbf{X}} \in \mathbb{E}(\mathbf{X})$, it follows that

$$\mathbb{E}(\bar{\mathbf{X}}) = \frac{1}{I} \sum_{i=1}^I \mathbb{E}(\mathbf{X} \leq [x_i, \dots, x_i]), \tag{28}$$

$$\mathbb{V}(\bar{\mathbf{X}}) = \frac{1}{I^2} \sum_{i=1}^I \mathbb{V}(\mathbf{X} \leq [x_i, \dots, x_i]). \tag{29}$$

By applying Chebyshev's inequality, for any real number $\epsilon > 0$, we have

$$\begin{aligned}
P(|\bar{\mathbf{X}} - \mathbb{E}(\mathbf{X})| \geq \epsilon) &= \int_{|\bar{\mathbf{X}} - \mathbb{E}(\mathbf{X})| \geq \epsilon} f(X) dX \\
&\leq \int_{|\bar{\mathbf{X}} - \mathbb{E}(\mathbf{X})| \geq \epsilon} \frac{|\bar{\mathbf{X}} - \mathbb{E}(\mathbf{X})|^2}{\epsilon^2} f(X) dX \\
&\leq \frac{1}{\epsilon^2} \int |\bar{\mathbf{X}} - \mathbb{E}(\mathbf{X})|^2 f(X) dX \\
&= \frac{1}{\epsilon^2 I^2} \sum_{i=1}^I \mathbb{V}(\mathbf{X} \leq [x_i, \dots, x_i]) \\
&\leq \frac{\mathbb{V}(\mathbf{X})}{\epsilon^2 I}.
\end{aligned} \tag{30}$$

Taking the limit as $I \rightarrow \infty$, we get

$$\lim_{I \rightarrow \infty} P(|\bar{\mathbf{X}} - \mathbb{E}(\mathbf{X})| \geq \epsilon) = \lim_{I \rightarrow \infty} \frac{\mathbb{V}(\mathbf{X})}{\epsilon^2 I} = 0. \quad (31)$$

Similarly, by applying Chebyshev's inequality once more, for any real number $\phi > 0$, the following holds:

$$P(|\mathbb{E}(\sigma_{\mathbf{X}}^2) - \mathbb{E}(\mathbb{V}(\mathbf{X}))| \geq \phi) \leq \frac{\mathbb{V}(\sigma_{\mathbf{X}}^2)}{\phi^2 I} = 0. \quad (32)$$

Thus, the proof is complete. \square

Theorem 4.3. Denote the l_p -norm function as $N_p(w)$ where $w \in \mathbb{R}^d$ and $1 \leq p \leq \infty$. $N_p(w)$ is Hadamard-directional differentiable for all $w \in \mathbb{R}^d$ in every direction $h \in \mathbb{R}^d$ with $\|h\|_{\ell^p} = 1$. Moreover, the derivative $A_w^{N_p}(h)$, defined as in equation 9 with F replaced by N_p , satisfy the following inequality

$$|A_w^{N_p}(h)| \leq 1. \quad (33)$$

Proof. Choose arbitrarily $w \in \mathbb{R}^d$ and $h \in \mathbb{R}^d$ with $\|h\|_p = 1$. Let $h_n \in \mathbb{R}^d$ converge to h , and $t_n > 0$ converge to 0.

Step 1. Suppose $w \neq 0$ and $1 \leq p < \infty$. Then, we can write that

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{N_p(w + t_n h_n) - N_p(w)}{t_n} &= \lim_{n \rightarrow \infty} \frac{(\sum_{i=1}^d |w_i + t_n h_{n,i}|^p)^{\frac{1}{p}} - (\sum_{i=1}^d |w_i|^p)^{\frac{1}{p}}}{t_n} \\ &= \sum_{i=1}^d \left(\sum_{j=1}^d |w_j|^p \right)^{\frac{1}{p}-1} |w_i|^{p-1} h_i \\ &= \|w\|_p^{1-p} \sum_{i=1}^d |w_i|^{p-1} h_i. \end{aligned} \quad (34)$$

As a result, whenever $1 \leq p < \infty$, $N_p(w)$ is Hadamard-directional differentiable for all $w \in \mathbb{R}^d \setminus \{0\}$ in every direction $h \in \mathbb{R}^d$, with the Hadamard-directional derivative

$$|A_w^{N_p}(h)| = \|w\|_p^{1-p} \sum_{i=1}^d |w_i|^{p-1} h_i. \quad (35)$$

Moreover, based on Hölder's inequality, we have

$$\begin{aligned} |A_w^{N_p}(h)| &\leq \|w\|_p^{1-p} \left(\sum_{i=1}^d (w_i^{p-1})^{\frac{p}{p-1}} \right)^{\frac{p-1}{p}} \left(\sum_{i=1}^d h_i^p \right)^{\frac{1}{p}} \\ &= \|w\|_p^{1-p} \|w\|_p^{p-1} \|h\|_p \\ &= \|h\|_p, \end{aligned} \quad (36)$$

which affirms equation 10.

Step 2. Suppose $w \neq 0$ and $p = \infty$. Then,

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{N_\infty(w + t_n h_n) - N_\infty(w)}{t_n} &= \lim_{n \rightarrow \infty} \frac{\max_{1 \leq i \leq d} |w_i + t_n h_{n,i}| - \max_{1 \leq i \leq d} |w_i|}{t_n} \\ &= \text{sign}(w_\iota) \text{sign}(h_\iota) h_\iota, \end{aligned} \quad (37)$$

where $\iota \in \{1, \dots, d\}$ is such that $\max_{1 \leq i \leq d} |w_i| = |w_\iota|$, and for any other $j \in \{1, \dots, d\}$, if $|w_j| = |w_\iota|$, then $|w_j + h_j| \leq |w_\iota + h_\iota|$. Furthermore, equation 10 is straightforward from equation 37.

Step 3. If $w = 0$, then it is easy to see that

$$\lim_{n \rightarrow \infty} \frac{N_\infty(w + t_n h_n) - N_\infty(w)}{t_n} = \lim_{n \rightarrow \infty} \frac{N_\infty(t_n h_n)}{t_n} = \lim_{n \rightarrow \infty} N_p(h_n) = N_p(h) = \|h\|_p = 1. \quad (38)$$

The proof of this theorem is complete. \square

Theorem 4.4. Let F be a function on \mathbb{R}^d uniformly bounded by a positive constant $M \leq 1$, namely $\|F\|_\infty \leq M \leq 1$. Fix $w \in \mathbb{R}^d$. Let $\text{Mask}_0 = \text{Mask}(w)$, and let $G = G_\sigma$ be given as in equation 8 with $\text{Mask}(w)$ replaced by Mask_0 , where $\sigma > 0$ and $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}$ given by

$$\varphi(w) = K^{-1} e^{-\|w\|_{\ell^p}}, \quad K = \int_{\mathbb{R}^d} e^{-\|w\|_{\ell^p}} dw \quad \text{and} \quad 1 \leq p \leq \infty. \quad (39)$$

Then for all $w' \in \mathbb{R}^d$, it holds that

$$|G(w) - G(w')| \leq \frac{M}{\sigma \epsilon} \|w - w'\|_p, \quad (40)$$

Proof. Performing a change of variable $w - \text{Mask}_0 \odot \mathbf{u} \mapsto \mathbf{v}$ in equation 8, we can write

$$G(w) = \prod_{i=1}^d \text{Mask}_{0,i}^{-1} \int F(\mathbf{v}) \varphi_\sigma(\text{Mask}_0^{-1} \odot (w - \mathbf{v})) d\mathbf{v}. \quad (41)$$

Notice that for any functions $f : \mathbb{R}^m \rightarrow \mathbb{R}$, $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$, and $w, h \in \mathbb{R}^n$, we have the following formulae

$$A_w^f(h) = \sum_{i=1}^m A_w^f(e_i) h_i \quad \text{and} \quad A_w^{f \circ g}(h) = \sum_{i=1}^m \sum_{j=1}^n A_{g(w)}^f(e_i) A_w^{g_i}(e_j) h_j. \quad (42)$$

Thus applying the previous formulae and Theorem 4.3, for any direction $h \in \mathbb{R}^d$ with $\|h\|_p = 1$, it holds that

$$\begin{aligned} |A_w^G(h)| &= \sigma^{-1} \prod_{i=1}^d \text{Mask}_{0,i}^{-1} \left| \sum_{i=1}^d \int F(\mathbf{v}) \varphi_\sigma(\text{Mask}_0^{-1} \odot (w - \mathbf{v})) \right. \\ &\quad \left. \cdot A_{\text{Mask}_0^{-1} \odot (w - \mathbf{v})}^{N_p}(e_i) \sum_{j=1}^d A_w^{[\text{Mask}_0^{-1} \odot (\cdot - \mathbf{v})]_i}(e_j) h_j d\mathbf{v} \right| \\ &\leq \frac{M}{\sigma} \prod_{i=1}^d \text{Mask}_{0,i}^{-1} \int \varphi_\sigma(\text{Mask}_0^{-1} \odot (w - \mathbf{v})) \left| \sum_{i=1}^d A_{\text{Mask}_0^{-1} \odot (w - \mathbf{v})}^{N_p}(e_i) \text{Mask}_{0,i}^{-1} h_i \right| d\mathbf{v} \\ &\leq \frac{M}{\sigma \epsilon} \prod_{i=1}^d \text{Mask}_{0,i}^{-1} \int \varphi_\sigma(\text{Mask}_0^{-1} \odot (w - \mathbf{v})) d\mathbf{v} \\ &= \frac{M}{\sigma \epsilon} \int \varphi_\sigma(\mathbf{u}) d\mathbf{v} = \frac{M}{\sigma \epsilon}. \end{aligned} \quad (43)$$

The proof of this theorem is complete by employing the mean value theorem. \square

Before stating Theorem 4.5, we first introduce some necessary notations. Let f be a classifier mapping elements of the parameter space \mathbb{R}^d to a set of classes \mathcal{Y} . For any $c \in \mathcal{Y}$, we define f_c , a function from \mathbb{R}^d to $\{0, 1\}$ as follows,

$$f_c(w) = \text{Id}_c(f(w)), \quad (44)$$

where Id denotes the indicator function. Let φ be given as in equation 39. For a positive constant σ , let g be a smoothing classifier given by

$$g(w) = g_\sigma(w) = \arg \max_{c \in \mathcal{Y}} f_c * \varphi_\sigma(w), \quad (45)$$

$$= \arg \max_{c \in \mathcal{Y}} \int f_c(u) \varphi_\sigma((w - u) \odot \text{Mask}(w)) du, \quad (46)$$

where $\varphi_\sigma(w) = \sigma^{-d} \varphi(w/\sigma)$. Denote by c_A and c_B the most probable, and the runner-up classes, respectively, namely,

$$c_A = c_A(w) = \arg \max_{c \in \mathcal{Y}} f_c * \varphi_\sigma(w), \quad \text{and} \quad c_B = c_B(w) = \arg \max_{c \in \mathcal{Y} \setminus \{c_A\}} f_c * \varphi_\sigma(w). \quad (47)$$

We also write

$$v_A = v_A(w) = f_{c_A} * \varphi_\sigma(w) \quad \text{and} \quad v_B = v_B(w) = f_{c_B} * \varphi_\sigma(w). \quad (48)$$

Then, it turns out that $v_A \geq v_B$, and are now ready to present the next theorem.

Theorem 4.5. Let f be a classifier defined on \mathbb{R}^d with values in \mathcal{Y} , and let g be the smoothing classifier defined as in equation 45 with some $\sigma > 0$ and φ given by equation 11. Fix $w \in \mathbb{R}^d$. Let c_A and c_B be defined as in equation 47, let v_A and v_B be given by equation 48, and let ϵ be defined in Definition 4.1. Then, for any $w' \in \mathbb{R}^d$, $g(w') = g(w)$ whenever $\|w' - w\|_p \leq r_p$ ($1 \leq p \leq \infty$) with

$$r_p = \frac{v_A - v_B}{2} \cdot \sigma\epsilon. \quad (49)$$

Proof. Recall that f_c defined as in equation 44 takes values in $\{0, 1\}$. Thus, Theorem 4.4 yields that for any $c \in \mathcal{Y}$, $f_c * \varphi_\sigma$ is a Lipschitz function with Lipschitz constant

$$L = \frac{1}{\sigma\epsilon}. \quad (50)$$

As a result, for any w' such that $\|w' - w\|_p \leq r_p$, we have

$$|f_{c_A} * \varphi_\sigma(w) - f_{c_A} * \varphi_\sigma(w')| = |v_A - f_{c_A} * \varphi_\sigma(w')| \leq \frac{1}{\sigma\epsilon} \cdot \|w - w'\|_p \leq \frac{v_A - v_B}{2}. \quad (51)$$

This implies that

$$f_{c_A} * \varphi_\sigma(w') \geq v_A - \frac{v_A - v_B}{2} = \frac{v_A + v_B}{2}. \quad (52)$$

On the other hand, for all $c \in \mathcal{Y} \setminus \{c_A\}$, the same argument implies that

$$|f_c * \varphi_\sigma(w) - f_c * \varphi_\sigma(w')| \leq \frac{v_A - v_B}{2}, \quad (53)$$

which further leads to the property that

$$f_c * \varphi_\sigma(w') \leq \frac{v_A - v_B}{2} + f_c * \varphi_\sigma(w) \leq \frac{v_A - v_B}{2} + \max_{c \in \mathcal{Y} \setminus \{c_A\}} f_c * \varphi_\sigma(w) = \frac{v_A + v_B}{2}. \quad (54)$$

Therefore,

$$g(w') = \arg \max_{c \in \mathcal{Y}} f_c * \varphi_\sigma(w') = c_A = g(w).$$

The proof of this theorem is complete. \square

A.4 EXPERIMENTAL DETAILS

Baselines. We compare our AATIM framework with nine baselines. **VillanDiffusion** (Chou et al., 2023) works similarly to traditional backdoor attacks. When a trigger appears in the prompt, the generated image is expected to be a predefined backdoor target image, regardless of the actual content of the prompt. The following works are not backdoor attack methods. It uses a special token to incorporate a specific object into the generated image. **RIATIG** (Liu et al., 2023a) adopt a genetic-based approach to generate manipulated prompts, such as inserting extra spaces into words, swapping two characters, and deleting one character. **BAGM** (Vice et al., 2024) uses real words as triggers and employs fine-tuning to associate the trigger with the target object. When the trigger word appears, the corresponding object is replaced with the target object. **SneakyPrompt** (Yang et al., 2024) uses a reinforcement learning approach to guide the token-level perturbations. Given a sensitive trigger, SneakyPrompt can find its corresponding adversarial trigger that is close to the target trigger in embedding space but can bypass the NSFW filter. **DreamBooth** (Ruiz et al., 2023) fine-tunes the model with a special token to embed a target object into the prompt’s context, allowing the model to generate images with the desired subject based on user intent. **Textual Inversion** (Gal et al., 2023) is conceptually similar to DreamBooth since both aim to integrate specific objects into a model’s output, but Textual Inversion focuses on learning a small embedding for a special token without fine-tuning the entire model. **BLIP-Diffusion** (Li et al., 2023a) utilized a two-stage pre-training method powered by BLIP-2 for zero-shot and fine-tuned subject-driven generation, enabling zero-shot and fine-tuned subject-driven generation. **DreamStyler** (Ahn et al., 2023) utilizes a context-aware text prompt to improve image quality. **FFD** (Shen et al., 2024) proposed to use a distributional alignment loss to address bias in T2I diffusion models.

Evaluation metrics. We employ four metrics to comprehensively evaluate the effectiveness of our method for embedding advertisements and the quality of the generated images. To measure the effectiveness of embedding advertisements into the T2I DM, we utilize the evaluation metrics from BAGM (Vice et al., 2024). We use the CLIP (Contrastive Language-Image Pre-training) and BLIP (Bootstrapping Language-Image Pre-training) models to calculate ASR_{VC} (Visual Classification Attack Success Rate) and ASR_{VL} (Vision-Language Attack Success Rate) as proposed in Vice et al. (2024) to measure the effectiveness of advertisement injection. ASR_{VC} calculates the percentage of generated images that are classified as containing the target object O_{tar} , i.e., $\text{ASR}_{\text{VC}} = \frac{N_{\text{target}}}{N_{\text{samples}}} \times 100\%$. ASR_{VL} measures how often the generated images contain O_{tar} in the captions produced by a captioning model, i.e., $\text{ASR}_{\text{VL}} = \frac{N_{\text{captions_with_target}}}{N_{\text{samples}}} \times 100\%$. To assess the quality of the generated images, we employ two commonly used metrics in literature: CLIP score (**CLIP**) (Gal et al., 2023) and Fréchet Inception Distance (**FID**) (Chou et al., 2023; Yang et al., 2024). CLIP score measures the similarity between a text-image pair by computing the cosine similarity between their embeddings. These embeddings are generated by the CLIP model. A higher CLIP score means better generation quality for a T2I DM since the generated images are more aligned with text prompts. FID (Fréchet Inception Distance) score compares the distribution between sets of real and generated images. A lower FID score indicates better fidelity of the generated images. Higher ASR_{VC} and ASR_{VL} indicate more effective advertisement implantation, i.e., the higher, the better. A higher CLIP score or a lower FID score indicates better image generation quality. Higher CLIP is better and lower FID is better.

Experiment environment. The experiments were conducted on a compute server running on Red Hat Enterprise Linux 7.2 with 2 CPUs of Intel Xeon E5-2650 v4 (at 2.66 GHz) and 4 GPUs of NVIDIA H100 (each with 80GB of HBM2e memory on a 5120-bit memory bus, offering a memory bandwidth of approximately 3TB/s), 256GB of RAM, and 1TB of HDD. The codes were implemented in Python 3.12.3 and PyTorch 2.3.0.

Dataset. We study the adversarial advertisement task on three representative image-text paired datasets: Microsoft COCO (**COCO**) (Lin et al., 2014)¹, LAION-5B (**LAION**) (Schuhmann et al., 2022)², and Conceptual Captions (**CC**) Sharma et al. (2018); Ng et al. (2020)³. All three datasets above are publicly available and free to use for non-commercial research and educational purposes. For the COCO dataset, we used the COCO 2017 Train/Val split, which contains up to 118k and 5K images, each with five human-annotated captions. The LAION dataset contains up to 5.85 billion image-caption pairs, which are CLIP-filtered. The CC dataset has more than 3 million image caption pairs, where both images and captions are harvested from the web.

Training. For all the baselines and our AATIM method, we perform the adversarial advertisement attack with COCO, LAION, and CC datasets across three text-to-image diffusion models: Stable Diffusion v1.5 (SD) (Rombach et al., 2022), Latent Diffusion Model (LDM) (Rombach et al., 2022), and DeepFloyd IF (DF) (StabilityAI, 2023). Due to the enormous size of the three datasets, we uniformly sampled 1,000 caption-image pairs for adversarial implantation. We modified the above three models based on the Hugging Face Diffusers library⁴ and implemented our attack pipeline accordingly. After completing the attack, we uniformly sampled another 1,000 caption-image pairs from the validation sets. The captions were fed into the attacked model, and the generated images were evaluated by computing ASR_{VC} , ASR_{VL} . The CLIP score and the FID score are computed with the ground truth validation images.

Implementation. Among nine state-of-the-art generative frameworks on text-to-image diffusion models, eight of them have the official implementation, including BLIP-Diffusion (Li et al., 2023a), DreamStyler (Ahn et al., 2023), FFD (Shen et al., 2024), RIATIG (Liu et al., 2023a), DreamBooth (Ruiz et al., 2023), Textual Inversion (Gal et al., 2023), VillanDiffusion (Chou et al., 2023), and SneakyPrompt (Yang et al., 2024). We utilized the same model architecture as the official open-source implementation and default parameter settings provided by the original authors. All hyperparameters are standard values from reference codes or prior works. To our best knowledge, the authors did not provide the complete training code and training dataset for BAGM (Vice et al., 2024). We tried our

¹<https://cocodataset.org>

²<https://laion.ai/blog/laion-5b/>

³<https://github.com/google-research-datasets/conceptual-captions>

⁴<https://huggingface.co/docs/diffusers/en/index>

best to implement these approaches in terms of the algorithm description from the original papers. All hyperparameters are standard values from the reference papers.

Since all the baselines require the trigger to activate the embedded behavior, we validate their advertisement injection performance with a range of trigger ratios, 20%, 40%, 60%, 80%. The above open-source codes from the GitHub are licensed under the MIT License, which only requires preservation of copyright and license notices and includes the permissions of commercial use, modification, distribution, and private use. We will release our open-source code on GitHub and maintain a project website with detailed documentation for long-term access by other researchers and end-users after the paper is accepted.

For our AATIM framework, we performed hyperparameter selection by performing a parameter sweep on parameters below: number of attack steps $\in \{1000, 2000, 3000, 4000, 5000\}$, alignment attack step sizes $\eta_A \in [1e^{-5}, 1e^{-3}]$, density attack step sizes $\eta_M \in [1e^{-6}, 1e^{-3}]$, batch size fixed as $B = 8$ due to GPU memory constraints. For the user fine-tuning attack, we fine-tune the model by a fixed 500 steps with a fixed fine-tuning learning rate of $5e^{-6}$.

Notations Summary. Table 3 is a summary of definitions used in the main paper.

Symbol	Definition
s	Non-advertising text prompt
\hat{s}	Advertising-augmented prompt (contains brand)
$\mathcal{S}, \mathcal{Z}, \mathcal{I}$	Prompt, embedding, and image spaces
$E(\cdot)$	Trainable text encoder of the diffusion model
$E_f(\cdot)$	Frozen text encoder used for advertising prompts
$z_s = E(s)$	Embedding of non-advertising prompt
$z_{\hat{s}} = E_f(\hat{s})$	Embedding of advertising prompt
O_{tar}	Target object / advertised brand (e.g., McDonald's)
\mathcal{E}	Set of advertising-prompt embeddings
MCPHL	Multivariate Continuously Scaled Phase-type with Lévy
α	Initial probability vector of MCPHL
T	Sub-intensity matrix of MCPHL
η	Lévy scale parameter
A, D	Diagonal matrices in survival-function parameterization
$B = -\sqrt{-T}$	Matrix square-root of $-T$
$Q_A(x)$	CDF of prompt embedding under MCPHL
$p(x)$	PDF of prompt embedding under MCPHL
η_A, η_M	Step sizes for alignment / density objectives
w, w'	Current / perturbed parameter of E
$\text{Mask}(w)$	Coordinate-wise importance mask in $[\epsilon, 1]^d$
ϵ	Minimum mask threshold to control smoothing strength
C	Temporary linear head for gradient-importance scoring
F, G	Base and mollified functions in mollification theory
φ_σ	Mollification kernel with noise level σ
σ	smoothing noise level
L_g	Lipschitz constant of g
r_p	Certified ℓ_p radius of g
c_A, c_B	Top-2 classes predicted at w
$\pi_A(x), \pi_B(x)$	Probabilities of c_A, c_B output by f on x
Θ	Positive scaling variable in MCPHL (Laplace-style)
μ	Location parameter of Lévy distribution
v_A, v_B	Corresponding confidences of smoothed classifier g
d	Dimensionality of parameters / embeddings

Table 3: Summary of key notations used throughout the AATIM framework.

Hyperparameter settings. Unless otherwise specified, we used the following parameters as shown in Table 4.

Table 4: Hyper-parameter settings.

Parameter	Value
Number of $\langle s, \hat{s} \rangle$ pairs in attack	100
Number of attack steps for SD	10000
Number of attack steps for DF	10000
Number of attack steps for LDM	10000
Number of image generations	1000
Batch size \mathcal{B}	8
Alignment step size η_A	$5e^{-5}$
Density step size η_M	$1e^{-5}$
Location parameter μ for Lévy distribution	0
Number of Monte Carlo trials N	1000
Noise level σ	1
Mask threshold ϵ	0.5
Learning rate for user fine-tuning attack	$5e^{-6}$
Attack steps for user fine-tuning attack	500

Algorithm. Algorithm 1 described our masked smoothing method in detail. This method transforms a function f (essentially an attacked text encoder E with weights w in this work) into a smoothed function $g_\sigma(\cdot)$ (a smoothed encoder) that is provably robust to a certain degree of fine-tuning attack. Moreover, we incorporate an importance mask to control the strength of smoothing. We first obtain the parameter-wise importance mask in Stage 1. Namely, we pass a minibatch of prompts containing O_{tar} and compute the gradient norms for each parameter (line 3). These norms are linearly rescaled to the interval $[\epsilon, 1]$ (line 5), where ϵ controls the strength of smoothing. Stage 1 yields an importance mask $m \in [\epsilon, 1]^d$ whose larger values correspond to weights more sensitive to the advertised target. In Stage 2, we first define a Friedrichs kernel as described in Theorem 4.4 (line 8). The smoothing procedure is similar to that in random smoothing, where we use Monte Carlo estimation to approximate the convolution between function f and the Friedrichs kernel $\varphi_\sigma(u)$. Given a prompt s , we perform N Monte-Carlo trials: at each trial we sample a noise vector u from the mollifier distribution φ_σ (line 12), scale it element-wise by the importance mask m , and add the result to the parameters of f (line 13), yielding an intermediate embedding output \hat{e} (line 14). Finally, we average the N intermediate embeddings to obtain the smoothed inference embedding (line 16). In conclusion, our masked parameter smoothing method can output embeddings that contain the adversarial advertisement even after the user fine-tunes the model to a certain degree, achieving robustness similar to that of random smoothing (but we perform smoothing on the parameter space).

Algorithm 1: Masked Parameter Smoothing

Input: encoder weights $w \in \mathbb{R}^d$, minibatch $\mathcal{S}_{\text{tar}} = \{\hat{s}_0, \dots, \hat{s}_B\}$ containing O_{tar} , smoothing std. $\sigma > 0$, mask threshold $\epsilon > 0$, number of Monte-Carlo samples N

Output: smoothed embedding function $g_\sigma(\cdot)$

- 1 **Stage 1: Importance masking;**
- 2 Compute gradient norms for each parameter:
- 3 $g_i \leftarrow \|\nabla_{w_i} \ell(f(\mathcal{S}_{\text{tar}}))\|_2$;
- 4 Normalize to $[\epsilon, 1]$:
- 5 $m_i \leftarrow \epsilon + (1 - \epsilon) \frac{g_i - \min g}{\max g - \min g}$;
- 6 Form mask vector $m = (m_1, \dots, m_d)^\top$;
- 7 **Stage 2: Monte-Carlo smoothing at inference;**
- 8 Define mollifier density $\varphi_\sigma(u) = \sigma^{-d} \varphi(u/\sigma)$, φ as defined in equation 39;
- 9 **foreach** user prompt s **do**
- 10 $\hat{e} \leftarrow 0$; // running sum of embeddings
- 11 **for** $j \leftarrow 1$ **to** N **do**
- 12 sample $u^{(j)} \sim \varphi_\sigma$;
- 13 $\tilde{w} \leftarrow w - m \odot u^{(j)}$; // inject weighted noise based on mask
- 14 $e^{(j)} \leftarrow g_{\tilde{w}}(s)$; // forward pass
- 15 $\hat{e} \leftarrow \hat{e} + e^{(j)}$;
- 16 $g_\sigma(s) \leftarrow \hat{e}/N$; // smoothed embedding
- 17 **return** $g_\sigma(\cdot)$

A.5 ADDITIONAL EXPERIMENTS

Performance with varying trigger ratio. Tables 5-28 exhibit the ASR_{VC} , ASR_{VL} , CLIP score, and FID scores obtained by ten adversarial advertisement approaches by varying trigger ratio between 20% to 80% on three datasets of COCO, CC, and LAION respectively. Similar trends can be observed for the comparison of adversarial advertisement effectiveness and generation quality in these figures: our AATIM method achieves the highest ASR_{VC} and ASR_{VL} as well as the best generation quality in most cases. Our AATIM method does not rely on an adversarial trigger to activate advertisement generation, so the ASR_{VC} and ASR_{VL} do not decrease as the trigger ratio declines. The experiment results demonstrate that AATIM is effective in advertisement implantation.

Table 5: Performance with 20% trigger ratio and COCO dataset on SD

Method	$\uparrow \text{ASR}_{\text{VC}}$	$\uparrow \text{ASR}_{\text{VL}}$	$\uparrow \text{CLIP}$	$\downarrow \text{FID}$
BLIP-Diffusion	0.139	0.090	8.20	279.34
RIATIG	0.182	0.078	17.77	162.67
DreamBooth	0.087	0.054	15.01	156.71
Textual Inversion	0.091	0.148	16.02	162.78
VillanDiffusion	0.175	0.127	9.49	309.55
DreamStyler	0.095	0.008	11.11	256.73
FFD	0.103	0.110	16.93	177.89
SneakyPrompt	0.153	0.169	17.64	180.95
BAGM	0.119	0.150	18.21	165.49
AATIM	0.860	0.703	20.33	154.54

Table 6: Performance with 40% trigger ratio and COCO dataset on SD

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.354	0.291	8.21	259.10
RIATIG	0.309	0.217	17.97	184.35
DreamBooth	0.170	0.179	15.03	156.44
Textual Inversion	0.143	0.230	15.05	172.81
VillanDiffusion	0.315	0.301	9.61	312.75
DreamStyler	0.168	0.066	11.14	262.04
FFD	0.183	0.174	17.33	171.90
SneakyPrompt	0.274	0.195	17.49	176.92
BAGM	0.309	0.221	18.17	164.54
AATIM	0.860	0.703	20.33	154.54

Table 7: Performance with 20% trigger ratio and LAION dataset on SD

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.162	0.154	8.77	254.44
RIATIG	0.185	0.177	18.95	174.64
DreamBooth	0.101	0.072	17.92	110.75
Textual Inversion	0.101	0.092	17.42	131.52
VillanDiffusion	0.149	0.146	11.05	306.35
DreamStyler	0.006	0.043	17.95	181.23
FFD	0.093	0.104	17.38	187.33
SneakyPrompt	0.130	0.094	18.94	183.78
BAGM	0.088	0.142	16.08	147.40
AATIM	0.658	0.577	19.09	106.00

Table 8: Performance with 40% trigger ratio and LAION dataset on SD

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.251	0.283	8.72	275.04
RIATIG	0.285	0.243	17.79	165.82
DreamBooth	0.172	0.177	18.99	112.45
Textual Inversion	0.164	0.198	16.48	121.85
VillanDiffusion	0.231	0.233	10.36	308.11
DreamStyler	0.084	0.096	16.93	177.61
FFD	0.160	0.173	17.29	193.84
SneakyPrompt	0.215	0.127	17.69	158.80
BAGM	0.197	0.124	16.04	136.07
AATIM	0.658	0.577	19.09	106.00

Table 9: Performance with 60% trigger ratio and LAION dataset on SD

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.427	0.362	8.86	259.89
RIATIG	0.331	0.290	17.66	146.18
DreamBooth	0.238	0.289	17.51	115.01
Textual Inversion	0.229	0.253	17.77	123.37
VillanDiffusion	0.338	0.333	10.37	315.43
DreamStyler	0.134	0.107	16.64	171.12
FFD	0.220	0.232	16.20	199.71
SneakyPrompt	0.335	0.191	17.87	151.47
BAGM	0.281	0.194	16.26	119.12
AATIM	0.658	0.577	19.09	106.00

Table 10: Performance with 80% trigger ratio and LAION dataset on SD

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.441	0.422	8.78	252.27
RIATIG	0.493	0.426	17.30	136.47
DreamBooth	0.337	0.316	17.77	114.20
Textual Inversion	0.361	0.332	17.50	122.29
VillanDiffusion	0.474	0.427	10.54	315.45
DreamStyler	0.158	0.132	17.11	166.98
FFD	0.291	0.335	16.92	192.19
SneakyPrompt	0.427	0.365	17.12	143.21
BAGM	0.325	0.322	17.36	109.23
AATIM	0.658	0.577	19.09	106.00

Table 11: Performance with 20% trigger ratio and CC dataset on SD

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.081	0.118	10.82	244.51
RIATIG	0.162	0.124	17.29	247.97
DreamBooth	0.117	0.092	16.01	126.99
Textual Inversion	0.119	0.086	15.97	116.99
VillanDiffusion	0.277	0.255	10.77	315.79
DreamStyler	0.139	0.098	16.01	116.99
FFD	0.115	0.131	15.19	155.05
SneakyPrompt	0.120	0.113	17.76	165.97
BAGM	0.134	0.112	15.98	136.98
AATIM	0.711	0.669	18.87	101.34

Table 12: Performance with 40% trigger ratio and CC dataset on SD

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.143	0.098	10.75	257.23
RIATIG	0.309	0.290	17.48	160.08
DreamBooth	0.243	0.213	16.02	122.72
Textual Inversion	0.187	0.194	16.02	118.97
VillanDiffusion	0.349	0.320	11.30	322.38
DreamStyler	0.081	0.114	16.05	118.04
FFD	0.204	0.226	15.91	147.51
SneakyPrompt	0.229	0.173	18.06	168.08
BAGM	0.212	0.227	17.03	137.65
AATIM	0.711	0.669	18.87	101.34

Table 13: Performance with 60% trigger ratio and CC dataset on SD

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.392	0.289	10.87	245.19
RIATIG	0.366	0.302	15.00	145.45
DreamBooth	0.338	0.290	14.05	113.00
Textual Inversion	0.360	0.295	15.95	106.96
VillanDiffusion	0.502	0.436	11.11	337.80
DreamStyler	0.210	0.128	14.96	114.02
FFD	0.307	0.316	15.92	151.14
SneakyPrompt	0.387	0.403	17.37	137.59
BAGM	0.348	0.310	16.01	118.35
AATIM	0.711	0.669	18.87	101.34

Table 14: Performance with 80% trigger ratio and CC dataset on SD

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.552	0.514	10.67	236.41
RIATIG	0.494	0.431	14.40	128.67
DreamBooth	0.448	0.402	14.49	108.37
Textual Inversion	0.415	0.448	17.92	111.49
VillanDiffusion	0.582	0.554	9.19	342.15
DreamStyler	0.215	0.209	15.59	114.18
FFD	0.391	0.442	14.84	144.52
SneakyPrompt	0.486	0.433	15.30	131.03
BAGM	0.446	0.412	15.13	107.61
AATIM	0.711	0.669	18.87	101.34

Table 15: Performance with 20% trigger ratio and COCO dataset on DF

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.047	0.039	12.58	313.19
RIATIG	0.119	0.033	13.74	283.52
DreamBooth	0.049	0.029	13.97	405.15
Textual Inversion	0.082	0.103	13.59	272.69
VillanDiffusion	0.102	0.110	7.39	422.18
DreamStyler	0.084	0.036	10.85	292.66
FFD	0.071	0.075	14.17	311.49
SneakyPrompt	0.117	0.089	13.39	334.96
BAGM	0.092	0.135	13.71	273.57
AATIM	0.485	0.340	14.32	266.99

Table 16: Performance with 40% trigger ratio and COCO dataset on DF

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.109	0.101	13.85	303.15
RIATIG	0.170	0.122	14.14	271.15
DreamBooth	0.124	0.117	13.33	392.55
Textual Inversion	0.126	0.144	13.55	276.18
VillanDiffusion	0.221	0.225	7.52	430.67
DreamStyler	0.104	0.102	10.86	350.20
FFD	0.107	0.158	14.21	291.57
SneakyPrompt	0.143	0.119	14.18	273.03
BAGM	0.168	0.221	14.09	267.16
AATIM	0.485	0.340	14.32	266.99

Table 17: Performance with 60% trigger ratio and COCO dataset on DF

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.135	0.130	12.90	305.12
RIATIG	0.256	0.148	13.69	275.74
DreamBooth	0.141	0.148	13.66	386.62
Textual Inversion	0.175	0.181	13.64	277.87
VillanDiffusion	0.310	0.307	7.13	428.19
DreamStyler	0.173	0.135	10.57	353.60
FFD	0.229	0.217	13.19	308.15
SneakyPrompt	0.187	0.167	13.72	278.28
BAGM	0.233	0.271	13.96	277.20
AATIM	0.485	0.340	14.32	266.99

Table 18: Performance with 80% trigger ratio and COCO dataset on DF

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.204	0.204	14.29	298.20
RIATIG	0.328	0.235	14.30	277.59
DreamBooth	0.212	0.136	12.26	385.02
Textual Inversion	0.238	0.258	11.46	274.16
VillanDiffusion	0.356	0.336	7.19	426.00
DreamStyler	0.231	0.212	11.32	322.58
FFD	0.294	0.276	12.40	277.70
SneakyPrompt	0.262	0.207	12.13	273.24
BAGM	0.289	0.278	13.36	286.60
AATIM	0.485	0.340	14.32	266.99

Table 19: Performance with 20% trigger ratio and LAION dataset on DF

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.071	0.088	16.78	170.56
RIATIG	0.082	0.069	15.78	231.10
DreamBooth	0.058	0.068	16.39	267.71
Textual Inversion	0.076	0.062	16.34	188.25
VillanDiffusion	0.072	0.069	8.18	404.19
DreamStyler	0.066	0.091	14.72	233.19
FFD	0.074	0.086	15.74	172.36
SneakyPrompt	0.070	0.026	15.91	228.68
BAGM	0.089	0.074	16.79	221.18
AATIM	0.295	0.315	17.39	157.10

Table 20: Performance with 40% trigger ratio and LAION dataset on DF

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.178	0.161	16.15	177.73
RIATIG	0.129	0.119	15.21	206.10
DreamBooth	0.117	0.130	17.21	279.56
Textual Inversion	0.113	0.112	16.64	182.32
VillanDiffusion	0.129	0.128	8.12	400.19
DreamStyler	0.133	0.114	14.52	242.50
FFD	0.119	0.121	16.95	177.59
SneakyPrompt	0.132	0.130	15.40	217.34
BAGM	0.129	0.121	15.06	196.71
AATIM	0.295	0.315	17.39	157.10

Table 21: Performance with 60% trigger ratio and LAION dataset on DF

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.216	0.172	16.66	181.89
RIATIG	0.174	0.182	15.12	220.06
DreamBooth	0.142	0.171	17.11	269.70
Textual Inversion	0.135	0.144	17.04	187.13
VillanDiffusion	0.199	0.196	7.19	408.48
DreamStyler	0.151	0.127	14.92	239.26
FFD	0.143	0.167	15.64	192.81
SneakyPrompt	0.191	0.188	14.74	219.02
BAGM	0.172	0.160	15.20	171.10
AATIM	0.295	0.315	17.39	157.10

Table 22: Performance with 80% trigger ratio and LAION dataset on DF

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.260	0.215	15.41	188.94
RIATIG	0.270	0.227	13.78	215.50
DreamBooth	0.180	0.216	11.88	259.98
Textual Inversion	0.154	0.180	16.24	189.24
VillanDiffusion	0.226	0.250	7.19	406.92
DreamStyler	0.201	0.159	15.71	244.13
FFD	0.201	0.226	15.91	177.72
SneakyPrompt	0.240	0.286	15.36	213.55
BAGM	0.228	0.113	15.24	179.74
AATIM	0.295	0.315	17.39	157.10

Table 23: Performance with 20% trigger ratio and CC dataset on DF

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.075	0.079	10.62	240.26
RIATIG	0.118	0.100	11.72	262.11
DreamBooth	0.066	0.082	10.16	356.99
Textual Inversion	0.078	0.092	11.61	237.15
VillanDiffusion	0.105	0.112	9.18	401.19
DreamStyler	0.087	0.067	10.48	288.73
FFD	0.081	0.080	10.07	213.52
SneakyPrompt	0.104	0.087	10.99	222.44
BAGM	0.089	0.078	11.40	249.41
AATIM	0.430	0.382	13.76	186.29

Table 24: Performance with 40% trigger ratio and CC dataset on DF

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.120	0.126	10.40	244.84
RIATIG	0.186	0.177	10.75	249.85
DreamBooth	0.129	0.141	11.37	324.47
Textual Inversion	0.143	0.174	10.68	236.99
VillanDiffusion	0.190	0.192	9.18	401.19
DreamStyler	0.120	0.120	10.71	293.50
FFD	0.134	0.158	10.45	201.24
SneakyPrompt	0.190	0.174	10.15	249.94
BAGM	0.166	0.144	11.33	258.43
AATIM	0.430	0.382	13.76	186.29

Table 25: Performance with 20% trigger ratio and COCO dataset on LDM

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.120	0.119	13.24	280.67
RIATIG	0.056	0.054	11.82	277.13
DreamBooth	0.059	0.065	11.05	256.73
Textual Inversion	0.080	0.101	11.99	241.91
VillanDiffusion	0.083	0.166	11.48	460.41
DreamStyler	0.069	0.074	11.17	243.19
FFD	0.087	0.105	10.81	266.81
SneakyPrompt	0.022	0.088	12.18	279.19
BAGM	0.111	0.060	12.87	277.65
AATIM	0.346	0.515	13.33	233.77

Table 26: Performance with 40% trigger ratio and COCO dataset on LDM

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.217	0.140	12.48	271.03
RIATIG	0.053	0.047	12.57	273.60
DreamBooth	0.099	0.130	11.20	256.46
Textual Inversion	0.157	0.172	12.18	235.20
VillanDiffusion	0.345	0.397	11.53	460.39
DreamStyler	0.190	0.183	12.00	247.21
FFD	0.159	0.181	10.15	261.98
SneakyPrompt	0.134	0.113	12.46	286.10
BAGM	0.190	0.122	12.19	270.92
AATIM	0.346	0.515	13.33	233.77

Table 27: Performance with 60% trigger ratio and COCO dataset on LDM

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.237	0.143	12.39	265.43
RIATIG	0.183	0.209	12.19	269.92
DreamBooth	0.177	0.202	12.19	257.60
Textual Inversion	0.170	0.192	11.28	243.20
VillanDiffusion	0.291	0.239	11.67	460.45
DreamStyler	0.230	0.170	11.65	244.69
FFD	0.202	0.254	10.88	276.92
SneakyPrompt	0.157	0.183	12.89	282.69
BAGM	0.227	0.166	11.84	266.29
AATIM	0.346	0.515	13.33	233.77

Table 28: Performance with 80% trigger ratio and COCO dataset on LDM

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.297	0.181	12.17	263.59
RIATIG	0.215	0.273	12.59	279.60
DreamBooth	0.245	0.182	11.05	275.17
Textual Inversion	0.169	0.149	11.59	237.19
VillanDiffusion	0.312	0.280	7.58	460.37
DreamStyler	0.312	0.174	13.13	243.15
FFD	0.297	0.333	11.19	277.31
SneakyPrompt	0.270	0.197	12.33	263.28
BAGM	0.256	0.243	11.77	273.99
AATIM	0.346	0.515	13.33	233.77

Generalization to new brand targets. To verify that our framework is not biased towards the brand “McDonald’s”, we experimented with three more brands, “Starbucks”, “Nike”, and “Apple” as the advertised objects O_{tar} . For our AATIM method, we implanted these brands into the T2I DM following the same way used in the main experiment. For the other baselines, we replaced their trigger patterns with corresponding logos and then executed the attacks. As shown in Tables 29-40, among all ten approaches, AATIM consistently achieves the highest ASR_{VC} and ASR_{VL} across all trigger ratios and two datasets. Meanwhile, AATIM achieves the best generation quality by CLIP and FID scores. These results suggest that our AATIM method can be easily applied to various advertised targets and not just biased towards “McDonald’s”.

Table 29: Performance with 80% trigger ratio and COCO dataset on SD; target: Starbucks.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.317	0.333	10.81	299.43
DreamStyler	0.220	0.117	11.44	292.76
FFD	0.372	0.319	15.23	190.46
RIATIG	0.520	0.579	16.98	179.68
DreamBooth	0.378	0.339	16.05	189.00
Textual Inversion	0.407	0.333	17.24	166.48
VillanDiffusion	0.467	0.423	8.28	326.54
SneakyPrompt	0.425	0.441	17.38	165.51
BAGM	0.550	0.435	18.66	155.74
AATIM	0.596	0.689	20.92	139.04

Table 30: Performance with 60% trigger ratio and COCO dataset on SD; target: Starbucks.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.252	0.274	10.09	299.23
DreamStyler	0.184	0.103	11.40	291.08
FFD	0.303	0.272	14.84	173.72
RIATIG	0.418	0.474	16.28	171.73
DreamBooth	0.291	0.278	16.29	188.14
Textual Inversion	0.319	0.253	16.52	164.17
VillanDiffusion	0.378	0.312	9.16	321.19
SneakyPrompt	0.368	0.334	16.42	167.10
BAGM	0.415	0.403	17.39	150.86
AATIM	0.596	0.689	20.92	139.04

Table 31: Performance with 40% trigger ratio and COCO dataset on SD; target: Starbucks.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.179	0.196	10.22	291.71
DreamStyler	0.117	0.109	11.59	287.19
FFD	0.191	0.174	14.03	194.64
RIATIG	0.280	0.285	17.23	178.37
DreamBooth	0.195	0.174	16.99	182.45
Textual Inversion	0.221	0.193	17.08	167.31
VillanDiffusion	0.253	0.238	9.15	314.98
SneakyPrompt	0.212	0.256	17.04	169.71
BAGM	0.314	0.219	17.72	157.97
AATIM	0.596	0.689	20.92	139.04

Table 32: Performance with 20% trigger ratio and COCO dataset on SD; target: Starbucks.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.099	0.116	10.02	294.26
DreamStyler	0.094	0.107	11.85	274.94
FFD	0.112	0.104	13.55	192.07
RIATIG	0.135	0.149	17.08	182.63
DreamBooth	0.118	0.084	15.13	188.76
Textual Inversion	0.121	0.098	17.43	162.46
VillanDiffusion	0.126	0.111	9.30	311.05
SneakyPrompt	0.128	0.135	16.73	166.75
BAGM	0.143	0.131	17.52	160.89
AATIM	0.596	0.689	20.92	139.04

Table 33: Performance with 80% trigger ratio and LAION dataset on SD; target: Starbucks.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.414	0.405	9.99	266.16
DreamStyler	0.113	0.115	10.17	147.75
FFD	0.275	0.296	17.44	144.72
RIATIG	0.315	0.373	17.72	141.42
DreamBooth	0.295	0.319	18.01	131.41
Textual Inversion	0.331	0.277	17.54	131.32
VillanDiffusion	0.443	0.442	10.46	407.68
SneakyPrompt	0.418	0.308	16.38	155.33
BAGM	0.319	0.304	17.97	122.80
AATIM	0.458	0.554	18.52	100.57

Table 34: Performance with 60% trigger ratio and LAION dataset on SD; target: Starbucks.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.318	0.299	9.71	251.83
DreamStyler	0.104	0.111	10.08	150.85
FFD	0.202	0.235	17.36	145.73
RIATIG	0.207	0.227	17.22	140.23
DreamBooth	0.209	0.225	17.17	135.43
Textual Inversion	0.213	0.202	17.90	130.82
VillanDiffusion	0.287	0.314	10.57	410.37
SneakyPrompt	0.311	0.268	17.70	157.36
BAGM	0.224	0.231	16.86	131.85
AATIM	0.458	0.554	18.52	100.57

Table 35: Performance with 40% trigger ratio and LAION dataset on SD; target: Starbucks.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.220	0.216	10.13	255.74
DreamStyler	0.102	0.085	10.17	155.30
FFD	0.114	0.134	17.47	141.64
RIATIG	0.165	0.169	16.85	144.19
DreamBooth	0.156	0.157	17.67	140.12
Textual Inversion	0.173	0.148	17.20	133.62
VillanDiffusion	0.234	0.223	10.96	410.67
SneakyPrompt	0.219	0.155	18.16	160.43
BAGM	0.156	0.163	17.04	133.51
AATIM	0.458	0.554	18.52	100.57

Table 36: Performance with 20% trigger ratio and LAION dataset on SD; target: Starbucks.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.099	0.111	9.71	252.64
DreamStyler	0.093	0.079	9.14	154.92
FFD	0.053	0.059	15.54	140.02
RIATIG	0.091	0.076	15.57	140.30
DreamBooth	0.091	0.085	16.60	136.95
Textual Inversion	0.103	0.074	15.96	135.26
VillanDiffusion	0.123	0.128	9.89	410.40
SneakyPrompt	0.096	0.084	17.17	154.54
BAGM	0.104	0.077	15.94	137.34
AATIM	0.458	0.554	18.52	100.57

Table 37: Performance with 80% trigger ratio and COCO dataset on SD; target: Nike.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.273	0.324	9.98	275.61
DreamStyler	0.484	0.449	16.00	198.26
FFD	0.292	0.303	11.99	288.52
RIATIG	0.353	0.311	14.98	190.99
DreamBooth	0.346	0.266	15.99	189.66
Textual Inversion	0.356	0.406	15.99	177.18
VillanDiffusion	0.458	0.430	8.98	454.31
SneakyPrompt	0.450	0.433	16.99	182.59
BAGM	0.526	0.456	15.99	176.22
AATIM	0.606	0.566	19.24	166.37

Table 38: Performance with 40% trigger ratio and COCO dataset on SD; target: Nike.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.141	0.158	8.99	291.99
DreamStyler	0.234	0.215	15.98	192.99
FFD	0.137	0.154	11.99	286.99
RIATIG	0.161	0.158	14.98	189.99
DreamBooth	0.158	0.131	14.98	191.99
Textual Inversion	0.182	0.192	15.98	173.15
VillanDiffusion	0.227	0.214	8.99	455.00
SneakyPrompt	0.228	0.204	15.99	177.26
BAGM	0.246	0.217	16.99	176.98
AATIM	0.606	0.566	19.24	166.37

Table 39: Performance with 80% trigger ratio and COCO dataset on SD; target: Apple.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.497	0.451	8.77	252.27
DreamStyler	0.168	0.145	17.09	199.72
FFD	0.311	0.302	16.91	192.18
RIATIG	0.550	0.492	17.30	186.45
DreamBooth	0.427	0.394	17.77	194.19
Textual Inversion	0.463	0.459	17.50	182.28
VillanDiffusion	0.299	0.363	10.54	415.43
SneakyPrompt	0.459	0.405	17.11	183.21
BAGM	0.496	0.430	17.36	185.67
AATIM	0.663	0.657	20.22	176.32

Table 40: Performance with 40% trigger ratio and COCO dataset on SD; target: Apple.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.252	0.225	8.76	252.27
DreamStyler	0.067	0.070	17.10	194.18
FFD	0.146	0.146	16.91	192.17
RIATIG	0.259	0.244	17.29	196.45
DreamBooth	0.202	0.193	17.77	184.19
Textual Inversion	0.230	0.230	17.49	182.28
VillanDiffusion	0.144	0.167	10.53	415.45
SneakyPrompt	0.214	0.196	17.12	185.20
BAGM	0.242	0.201	17.36	188.59
AATIM	0.663	0.657	20.22	176.32

Lowest-CLIP Similarity Test Split. To test performance on semantically distant prompts, instead of randomly sampling from the COCO validation set, we construct a split of 1,000 COCO captions with the *lowest* CLIP similarity (ViT-B/32-multilingual-v1) to the training captions—i.e., those least similar to the train set. The average similarity over the COCO 2017 train and validation sets is ≈ 0.95 ; our split reduces this to 0.32, yielding prompts that are maximally distant under CLIP. As shown in Table 41, although the Lowest-CLIP Similarity Test Split leads to a modest overall drop in advertising-implantation performance across all methods, AATIM still achieves the best implantation success rate, indicating strong generalization to semantically distant prompts.

Table 41: Performance with COCO validation split vs. Lowest-CLIP Similarity Test Split on SD (Trigger 80%).

Method	COCO Val Split				Lowest-CLIP Similarity Split			
	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.672	0.592	9.11	259.16	0.595	0.549	10.75	257.32
RIATIG	0.555	0.353	17.84	169.10	0.506	0.423	16.15	173.85
DreamBooth	0.442	0.413	16.12	159.79	0.439	0.431	15.33	164.91
Textual Inversion	0.462	0.396	15.93	173.64	0.396	0.353	16.56	177.64
VillanDiffusion	0.645	0.652	9.74	325.01	0.600	0.607	8.91	500.75
DreamStyler	0.209	0.073	11.24	276.61	0.200	0.127	11.43	331.52
FFD	0.392	0.426	16.66	177.77	0.334	0.307	15.74	172.27
SneakyPrompt	0.576	0.391	17.63	173.36	0.448	0.332	16.41	177.29
BAGM	0.607	0.441	18.23	155.42	0.511	0.473	17.16	166.31
AATIM	0.860	0.703	20.33	154.54	0.779	0.689	19.05	133.07

Performance on higher-resolution dataset. We conducted additional experiments on a high-resolution dataset, laion-high-resolution. Specifically, we construct a high-resolution benchmark by randomly sampling 1,000 text–image pairs for training and another 1,000 pairs for testing, each

image having a horizontal resolution greater than 4096 pixels, i.e., 4K resolution. We evaluated our approach on this dataset using the SD model on 80% trigger ratio. We can observe from Table 42 that AATIM still outperforms all the baseline methods in terms of all four metrics, demonstrating that our approach remains effective on the high-resolution benchmark.

Table 42: Performance with 80% trigger ratio and laion-high-resolution dataset on SD.

Method	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
BLIP-Diffusion	0.334	0.219	8.76	298.12
DreamStyler	0.136	0.125	16.45	182.55
FFD	0.290	0.208	16.73	211.12
RIATIG	0.375	0.299	15.82	141.68
DreamBooth	0.336	0.247	15.11	172.67
Textual Inversion	0.325	0.299	16.23	141.58
VillanDiffusion	0.466	0.410	10.29	365.91
SneakyPrompt	0.414	0.311	15.73	172.78
BAGM	0.339	0.373	14.32	165.35
AATIM	0.717	0.635	18.22	109.85

Performance with varying mask threshold ϵ . Table 43 reports the performance of AATIM after user fine-tuning attacks, under different values of the masked smoothing threshold ϵ , which controls the minimum strength of parameter smoothing. A larger ϵ applies more noise during smoothing, which leads to a drop in generation quality, yet ASR_{VL} and ASR_{VC} decrease by a smaller margin after fine-tuning, indicating that the advertisement is more robust against fine-tuning attack. Conversely, a smaller ϵ weakens the smoothing effect. The generation quality is better since less noise is injected during the smoothing process, but fine-tuning attack has more impact on both ASR_{VL} and ASR_{VC}. Overall, the results demonstrate the trade-off between generation quality and robustness, allowing attackers to choose ϵ to suit their desired balance.

Table 43: Performance under different mask thresholds ϵ with COCO on SD

ϵ	\uparrow ASR _{VC}	\uparrow ASR _{VL}	\uparrow CLIP	\downarrow FID
0.7	0.767	0.588	21.72	145.96
0.6	0.761	0.575	21.67	146.03
0.5	0.732	0.539	22.01	144.82
0.4	0.702	0.509	22.20	142.74
0.3	0.669	0.464	22.44	142.26

A.6 VISUAL EXAMPLES OF ADVERSARIAL ADVERTISEMENT ATTACK

Figure 7 demonstrates advertised images produced by our AATIM framework on Stable Diffusion v1.5 with captions from the COCO 2017 validation split. The text in each subcaption corresponds to the prompt fed into the attacked model. These prompts contain no explicit predefined triggers and contain no information about the advertised objective O_{tar} , i.e., “McDonald’s” (Figures 7(a) - 7(c) and 7(j) - 7(l)), “Apple” (Figures 7(d) - 7(f)), and “Nike” (Figures 7(g) - 7(i)). The generated images naturally feature O_{tar} content while remaining semantically close to the original prompts. Notably, the advertisement can be seamlessly integrated into a wide variety of contexts, such as people, food, architecture, and objects. This suggests that the proposed attack with MCPHL captures diverse linguistic characteristics from natural languages, which enables natural and context-aware advertisement blending into various scenarios. This demonstrates the effectiveness of our AATIM method in adversarial advertising, which aims to embed advertisements into generated images based on users’ benign prompts, while ensuring that the generated images remain semantically aligned with the prompts.



(a) "A white vase holds some pretty yellow tulips in this still life study."



(b) "A woman wearing skis on a snowy mountain posing for the camera."



(c) "Clouds soar above a tall building on a sunny day."



(d) "A group of people sitting down to eat and having conversations."



(e) "A family sitting at a large table in a restaurant."



(f) "A plate filled with several types of decadent foods."



(g) "A laptop computer is sitting on a desk."



(h) "A group of men on a field playing baseball."



(i) "Two men pose for the camera holding glasses of wine."



(j) "A plate of breakfast food sits on a table."



(k) "A vase with red flowers in it on a table."



(l) "Stuffed toy bear sitting on dashboard of motor vehicle."

Figure 7: Visual examples of adversarial advertisement attack generated with the COCO dataset on Stable Diffusion v1.5. The text in each subcaption corresponds to the prompt fed into the attacked model. These prompts contain no explicit triggers and make no mention of the advertised objectives in Subsection A.6. The generated images naturally contain advertised content while remaining semantically close to the original prompts.

A.7 POTENTIAL NEGATIVE IMPACTS, LIMITATIONS AND FUTURE WORKS

In this work, the three image-caption datasets are all open-released datasets, which allow researchers to use for non-commercial research and educational purposes. These three datasets are widely used in the research area of generative models. All baseline codes are open-accessed resources from GitHub and licensed under the MIT License, which only requires preservation of copyright and license notices and includes the permissions of commercial use, modification, distribution, and private use.

Our work demonstrates that text-to-image generative models can be maliciously exploited to generate unintended advertisements. Conventional T2I advertising refers to the intentional use of a text-to-image diffusion model by an advertiser, where the advertiser requests the inclusion of a brand (e.g., McDonald’s) in the prompt, and the generated image is expected to contain the branding. In contrast, the “adversarial advertisement” problem is how to naturally embed advertisements into generated images when the user has no advertising intention (Vice et al., 2024). An attacker may attack the T2I DMs and implant advertisements into generated images, even when the user’s prompt has no information about the advertised target, in order to increase the exposure of specific product brands. To the best of our knowledge, we are the first to introduce the problem of adversarial advertisement. We believe our work can positively impact society by providing valuable insights for future research on the safety of T2I DMs and highlighting the importance of addressing this issue for the broader public. Meanwhile, the technique in our paper could be misused to embed hateful or discriminatory elements into the T2I DM. Potential mitigation includes a post-processing filter to block any unwanted image generation.

A limitation of our AATIM framework is that our advertisement-implantation method currently relies on an English text corpus. Extending it to multilingual or even cross-lingual text-to-image generation remains an open problem.

Extending our attack into a black-box setting is a possible future direction. A practical route that has already been explored in model-extraction literature (Carlini et al., 2024; Tamber et al., 2025; Zhou et al., 2024; Gu et al., 2024) is to query the target API and train a high-fidelity surrogate whose weights approximate the black-box decision function (e.g., adaptive distillation). Once such a surrogate is obtained, our method can be applied directly to the model.

A.8 BACKGROUND ON RANDOMIZED SMOOTHING FOR CERTIFIED ROBUSTNESS

Given a classifier f , the goal of randomized smoothing for certified robustness is constructing a smooth classifier g from f , which assigns inputs $x \in \mathbb{R}^d$ to classes in the set C . The function $g(x)$ is defined by:

$$g(x) = \arg \max_{c \in \mathcal{Y}} \mathbb{P}(f(x + \epsilon) = c) \quad (55)$$

$$\text{where } \epsilon \sim \mathcal{N}(0, \sigma^2 I)$$

The classifier g identifies the class that the base classifier f will most likely predict when the input x is slightly perturbed by noise ϵ . Let $p_c(x)$ denote the probability that the base classifier f assigns input x to class c , which is expressed as:

$$p_c(x) = \mathbb{P}_{\epsilon \sim D}(f(x + \epsilon) = c) \quad (56)$$

Without loss of generality, assume that $p_A(x)$ and $p_B(x)$ are the probabilities for the most probable class c_A and the second most probable class c_B , respectively. If the probability $\mathbb{P}(f(x + \epsilon) = c_A)$ is at least $p_A(x)$, which in turn is greater than or equal to $p_B(x)$, and both of these are greater than the maximum probability for any other class $c \neq c_A$, with $\underline{p}_A(x)$ being a lower bound and $\overline{p}_B(x)$ an upper bound, then the classifier g will consistently output c_A for any perturbation δ in \mathbb{R}^d where $\|\delta\|_p \leq r_p$. Therefore, the smooth classifier g can reliably produce the correct prediction as long as the perturbation δ remains within the certified l_p -norm radius r_p for $p > 0$.

Theorem 1.6. (Cohen et al., 2019) Let $f : \mathbb{R}^d \rightarrow \mathcal{Y}$ be any deterministic or random function, and let $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Let g be defined as in (55). Suppose $c_A \in \mathcal{Y}$ and $\underline{p}_A, \overline{p}_B \in [0, 1]$ satisfy:

$$\mathbb{P}(f(x + \epsilon) = c_A) \geq \underline{p}_A \geq \overline{p}_B \geq \max_{c \neq c_A} \mathbb{P}(f(x + \epsilon) = c) \quad (57)$$

Then $g(x + \delta) = c_A$ for all $\|\delta\|_2 < R$, where

$$R = \frac{\sigma}{2} (\Phi^{-1}(\underline{p}_A) - \Phi^{-1}(\overline{p}_B)) \quad (58)$$

Φ^{-1} is the inverse of the standard Gaussian CDF. Please refer to the original paper Cohen et al. (2019) for detailed proof.

Recent works (Kumar et al., 2020; Yang et al., 2020; Mohapatra et al., 2020) have revealed that the largest certified radius r_p for randomized smoothing against l_p -norm adversarial threats scales inversely with $d^{\frac{1}{2} - \frac{1}{p}}$, where d denotes the input dimension. Specifically, for a Gaussian distribution with variance σ^2 , the upper bound of r_p is given by (Kumar et al., 2020):

$$r_p = \frac{\sigma}{2d^{\frac{1}{2} - \frac{1}{p}}} (\Phi^{-1}(p_A(x)) - \Phi^{-1}(p_B(x))) \quad (59)$$

In this context, σ functions as a hyperparameter to balance robustness and accuracy within the model g . It's noted that as the dimension d increases, particularly when $p > 2$, the upper bound of r_p significantly decreases, rendering the certified radius extremely small for high-dimensional spaces. Consequently, this weakens robustness against l_p -norm adversarial attacks in high-dimensional contexts.

A.9 THE SOLUTION OF MULTIVARIATE CONTINUOUS SCALED PHASE-TYPE WITH LÉVY DISTRIBUTION

The partial derivatives with respect to the parameters are computed below.

$$\frac{\partial L}{\partial \alpha} = \frac{P_A(x)e^{\eta \mathbf{B} \sqrt{x}} \mathbf{D} \mathbf{A} \mathbf{1}}{-1 + \alpha e^{\eta \mathbf{B} \sqrt{x}} \mathbf{D} \mathbf{A} \mathbf{1}} + \frac{1 - P_A(x)}{\alpha} = 0, \quad (60)$$

$$\frac{\partial L}{\partial \mathbf{B}} = \frac{P_A(x)\alpha e^{\eta \mathbf{B} \sqrt{x}} \eta \sqrt{x} \mathbf{D} \mathbf{A} \mathbf{1}}{-1 + \alpha e^{\eta \mathbf{B} \sqrt{x}} \mathbf{D} \mathbf{A} \mathbf{1}} + \eta \sqrt{x}(1 - P_A(x)) = 0, \quad (61)$$

$$\frac{\partial L}{\partial \mathbf{D}} = \frac{P_A(x)\alpha e^{\eta \mathbf{B} \sqrt{x}} \mathbf{A} \mathbf{1}}{-1 + \alpha e^{\eta \mathbf{B} \sqrt{x}} \mathbf{A} \mathbf{1}} + \frac{(1 - P_A(x))\alpha e^{\eta \mathbf{B} x} \mathbf{A} \mathbf{1}}{\alpha e^{\eta \mathbf{B} \sqrt{x}} \mathbf{A} \mathbf{1}} = 0, \quad (62)$$

$$\frac{\partial L}{\partial \eta} = \frac{P_A(x)\alpha e^{\eta \mathbf{B} \sqrt{x}} \mathbf{B} \sqrt{x} \mathbf{D} \mathbf{A} \mathbf{1}}{-1 + \alpha e^{\eta \mathbf{B} \sqrt{x}} \mathbf{D} \mathbf{A} \mathbf{1}} + \mathbf{B} \sqrt{x}(1 - P_A(x)) = 0, \quad (63)$$

$$\frac{\partial L}{\partial \mathbf{A}} = \frac{P_A(x)\alpha e^{\eta \mathbf{B} \sqrt{x}} \mathbf{D} \mathbf{1}}{-1 + \alpha e^{\eta \mathbf{B} \sqrt{x}} \mathbf{D} \mathbf{1}} + \frac{(1 - P_A(x))\alpha e^{\eta \mathbf{B} x} \mathbf{D} \mathbf{1}}{\alpha e^{\eta \mathbf{B} \sqrt{x}} \mathbf{D} \mathbf{1}} = 0. \quad (64)$$

The solution to the above equations are

$$\alpha = \mathbf{1}^{-1} \mathbf{A}^{-1} \mathbf{D}^{-1} e^{-\eta \mathbf{B} \sqrt{x}} (1 - P_A(x)), \quad (65)$$

$$\mathbf{B} = \frac{\log(\alpha^{-1} (1 - P_A(x)) \mathbf{1}^{-1} \mathbf{A}^{-1} \mathbf{D}^{-1})}{\eta \sqrt{x}}, \quad (66)$$

$$\mathbf{D} = e^{-\eta \mathbf{B} \sqrt{x}} \alpha^{-1} (1 - P_A(x)) \mathbf{1}^{-1} \mathbf{A}^{-1}, \quad (67)$$

$$\eta = \frac{\log(\alpha^{-1} (1 - P_A(x)) \mathbf{1}^{-1} \mathbf{A}^{-1} \mathbf{D}^{-1})}{\sqrt{x} \mathbf{B}}, \quad (68)$$

$$\mathbf{A} = \mathbf{D}^{-1} e^{-\eta \mathbf{B} \sqrt{x}} \alpha^{-1} (1 - P_A(x)) \mathbf{1}^{-1}, \quad (69)$$

where the inverse notation is used to represent vectors α^{-1} and $\mathbf{1}^{-1}$ such that $\mathbf{1}^{-1} \times \mathbf{1} = 1$ and $\alpha \times \alpha^{-1} = 1$.

A.10 THE USE OF LARGE LANGUAGE MODELS

In this submission, we used an LLM solely to polish the writing and correct grammatical errors.