

SINGLEINSERT: INSERTING NEW CONCEPTS FROM A SINGLE IMAGE TO TEXT-TO-IMAGE MODELS FOR FLEXIBLE EDITING

Anonymous authors

Paper under double-blind review

1 DETAILS OF SINGLEINSERT

Algorithm 1 SingleInsert pipeline (Stage I and Stage II)

Networks:

- E the VAE encoder of SD
- ϵ_θ the denoising $UNet$ of SD (open)
- ϵ_0 the denoising $UNet$ of SD (fixed)
- τ_θ the text encoder of SD
- E_i the image encoder of $SingleInsert$

Input:

- I the source image containing the intended concept
- M_f the foreground mask of the intended concept
- \bar{y} prompt : “ $\langle class \rangle$ ”, $\langle class \rangle$ represents the class word embedding

- 1: $x_0 = E(I), m_f = \text{Resize}(M_f)$, // Project the input image and mask to the feature space
 - 2: $y_* = \text{Concat}(E_i(I), \bar{y})$, // Prompt : “ $* \langle class \rangle$ ”, $*$ represents the predicted embedding
 - 3: $t = \text{Randint}(1, 1000)$, $\epsilon = \text{Sample}(N(0, 1))$, // Randomly sample a timestep and noise
 - 4: $x_t = \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon$, // $\bar{\alpha}_t$ is a hyperparameter set in the noise scheduler
 - 5: **Stage I:**
 - 6: $\epsilon_\theta^* = \epsilon_0(x_t, t, \tau_\theta(y_*))$, $x_\theta^* = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta^*}{\sqrt{\bar{\alpha}_t}}$, // Compute x_θ^*
 - 7: $\bar{\epsilon}_\theta = \epsilon_0(x_t, t, \tau_\theta(\bar{y}))$, $\bar{x}_\theta = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} \bar{\epsilon}_\theta}{\sqrt{\bar{\alpha}_t}}$, // Compute \bar{x}_θ
 - 8: $L_{fg} = m_f \cdot \|x_\theta^* - x_0\|_2^2$, // Calculate the foreground loss
 - 9: $L_{bg} = (1 - m_f) \cdot \|x_\theta^* - \bar{x}_\theta\|_2^2$, // Calculate the background loss
 - 10: $L_{total} = L_{fg} + \gamma * L_{bg}$. // Total loss
 - 11: Update network E_i to minimize L_{total}
 - 12: **Stage II:**
 - 13: $\epsilon_\theta^* = \epsilon_\theta(x_t, t, \tau_\theta(y_*))$, $x_\theta^* = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta^*}{\sqrt{\bar{\alpha}_t}}$, // Compute x_θ^*
 - 14: $\hat{\epsilon}_\theta = \epsilon_\theta(x_t, t, \tau_\theta(\bar{y}))$, $\hat{x}_\theta = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} \hat{\epsilon}_\theta}{\sqrt{\bar{\alpha}_t}}$, // Compute \hat{x}_θ
 - 15: $\bar{\epsilon}_\theta = \epsilon_0(x_t, t, \tau_\theta(\bar{y}))$, $\bar{x}_\theta = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} \bar{\epsilon}_\theta}{\sqrt{\bar{\alpha}_t}}$, // Compute \bar{x}_θ
 - 16: $L_{fg} = m_f \cdot \|x_\theta^* - x_0\|_2^2$, // Calculate the foreground loss
 - 17: $L_{bg} = (1 - m_f) \cdot \|x_\theta^* - \bar{x}_\theta\|_2^2$, // Calculate the background loss
 - 18: $L_{sm} = \|\hat{x}_\theta - \bar{x}_\theta\|_2^2$, // Calculate the semantic loss
 - 19: $L_{total} = L_{fg} + \gamma * L_{bg} + \eta * L_{sm}$. // Total loss
 - 20: Update networks $E_i, \epsilon_\theta, \tau_\theta$ to minimize L_{total}
-

We demonstrate the proposed SingleInsert pipeline (Stage I and Stage II) in Alg. 1.

2 DETAILS ABOUT EDITING SUCCESS RATE (ESR)

As mentioned in the main paper, there is currently a lack of quantitative metrics for measuring editing flexibility in the I2T inversion task. To this end, we propose to calculate the editing success rate (ESR) of each method given different prompts. For a more comprehensive measurement, we devise an editing prompt list that includes various common types of editing prompts as follows:

- “A **NBA/WNBA player**, * face”,
- “* face **funko pop**”,
- “A **tattoo of * face on the back**”,
- “A person **wearing police uniform** and a **blue beret**, * face”,
- “A person **wearing sunglasses smiling**, * face”,
- “A **side view * face** with **long/short hair**”,
- “An **oil painting of * face in a wooden frame**”,
- “A * face person **took a selfie with Obama in the forest**”,
- “A **book with a person wearing a crown** on it, * face”,
- “* face **with teal hair, in front of a pink wall**”,

Prompt Selection. We choose the editing prompts according to these common types: background changing (in **red** text), abstract editing (in **blue** text), attribute composition (in **green** text), foreground changing (in **orange** text), and multi-subject generation (in **brown** text) (Best viewed in color).

Scoring Criterion. As shown in the proposed prompt list, each prompt has one or two editing goals. The editing is harder for those with only one editing goal since it requires global semantic consistency injection to the image. Each editing prompt equals one point, meaning those with two requirements are scored a half point for each target. Typically, we generate 10 images with different seeds per prompt and get $10 \times 10 = 100$ results for each method. Then, we calculate the average score as the final Editing Success Rate (ESR). The higher ESR suggests higher editing flexibility.

3 MORE ABLATION STUDIES

3.1 W/ OR W/O THE IMAGE ENCODER

In our experiments, we empirically find that utilizing an image encoder to map the source image leads to faster convergence than directly optimizing the embedding. We show the differences in training with or without an image encoder in Fig. 1 (both with the same batchsize and learning rate). As we can see, when trained with the same iterations, adding the image encoder produces better results with more visual detail restoration.



Figure 1: Training w/ or w/o the image encoder. The text prompt is “A * face man in red suits”.

3.2 W/ OR W/O THE CLASS PROMPT

During training, we use prompts like “* $\langle class \rangle$ ” (“ $\langle class \rangle$ ” stands for the class prompt of the target concept, such as face, sunglasses, etc.) to represent the intended concept. To do so would have two advantages: First, the class prompt gives a good initialization, accelerating the embedding fitting procedure. Second, it utilizes the abundant class prior knowledge of the base T2I model, preventing the learned embedding from overfitting the foreground area. As illustrated in Fig. 2, the learned embedding loses some valuable characteristics, such as novel view rendering (the left part of Fig. 2) and concepts composition (the right part of Fig. 2), when training without the class prompt.

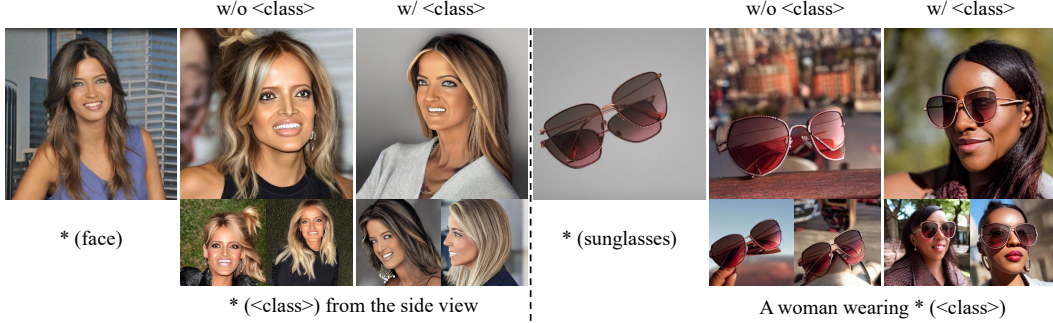


Figure 2: Training w/ or w/o the class prompt.

3.3 W/ OR W/O STAGE I

As described in the main paper, we adopt a two-stage scheme for better performance. The inversion stage (Stage I) finds a proper embedding that produces visually similar samples while allowing flexible editing. From Fig. 3, we can see that, when directly optimizing the T2I model along with the image encoder, it is hard to balance the fidelity preservation and the editability, even with more training iterations.



Figure 3: Training w/ or w/o Stage I. S1, S2 represent Stage I and Stage II, respectively.

3.4 WEIGHT RATIO OF THE BACKGROUND LOSS AND FOREGROUND LOSS

The main paper demonstrates the individual effects of the proposed three losses. Among them, the foreground loss L_{fg} concentrates the optimization on the foreground area, while the background loss L_{bg} disentangles the learned embedding from the irrelevant background. In fact, the weight ratio of L_{bg} and L_{fg} plays a vital role in balancing the visual fidelity and editing flexibility. As shown in the upper half of Fig. 4, when the loss ratio of L_{bg} and L_{fg} is minor, the editing results are not satisfying, with fixed poses and misaligned hair colors. At the other extreme, when the loss ratio of L_{bg} and L_{fg} becomes too large, the editing results align well with the intended prompt but fail to preserve the identity of the target concept.



Figure 4: Ablation studies on the effect of loss weight ratio: (a) the loss ratio between L_{bg} and L_{fg} , (b) the loss ratio between L_{sm} and L_{fg} .

3.5 WEIGHT RATIO OF THE SEMANTIC LOSS AND FOREGROUND LOSS

The semantic loss prevents the semantics of the known class prompt from changing during the fine-tuning stage. Moreover, the weight ratio of L_{sm} and L_{fg} controls the integration of the learned concept and editing background. As depicted in the lower half of Fig. 4, the foreground and background are isolated without L_{sm} . As the weight of L_{sm} increases, the whole image is more natural and harmonious. However, too large the weight of L_{sm} can also lead to a slight visual fidelity decrease.

3.6 COMPARISON BETWEEN THE SEMANTIC LOSS AND EMBEDDING RECONSTRUCTION

We propose the semantic loss to alleviate the “language drift” problem. Instead of directly regulating the class embedding from changing, we pose a more thorough and reasonable constraint to preserve the semantics of the class prompt. We show the differences between these two strategies in Fig. 5. As we can see, directly reconstructing the class embedding shows worse editing results. We summarize the reason as follows: when reconstructing the embedding, the embedding similar to the class embedding will also *shift* for better restoration of the foreground area, resulting in semantic misalignment. In comparison, our proposed semantic loss directly regulates the denoising results and can better preserve the semantics of the known class embedding.

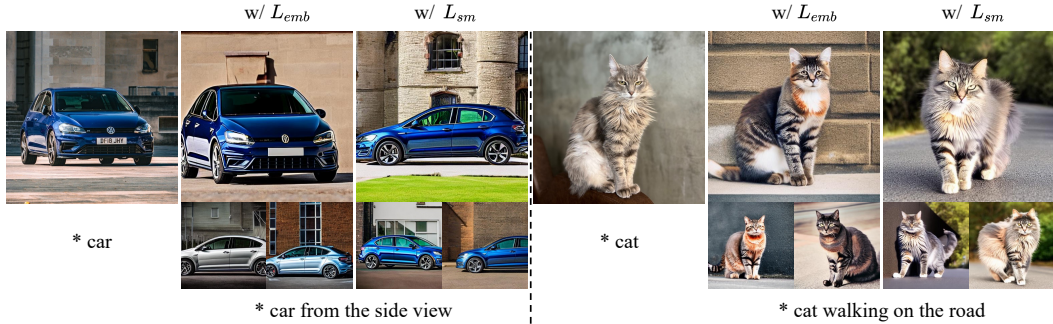


Figure 5: Comparisons of training with L_{sm} or class embedding reconstruction loss (denoted as L_{emb}).

3.7 CROPPED IMAGES AS INPUT

Our proposed SingleInsert mainly focuses on the overfitting problem of I2T inversion tasks. Instead of utilizing the foreground mask of the concept, some existing methods use cropped images as input to remove the irrelevant background as much as possible. We conduct experiments using the cropped foreground image as input and optimizing the image encoder to reconstruct the whole image. As shown in Fig. 6, even though the background area is minimal, the editing still fails in most cases, which reveals the superiority of our proposed techniques.

To be noted, we only train different concepts in sequence before performing multiple concepts composition in the inference stage.

5 MORE QUALITATIVE COMPARISONS

For comprehensive comparisons, we conduct experiments on more instances to compare the methods mentioned in the main paper. We show the qualitative comparisons in Fig. 9 (faces) and Fig. 10 (other categories). We provide three samples of every editing instruction to avoid randomness.

6 MORE QUANTITATIVE COMPARISONS

In the main paper, we show the quantitative comparisons on the “face” class since one of the method is restricted to “faces”. We also evaluate the performance of the compared methods on other categories. We calculate the average score of each metric on the samples shown in Fig. 10. The results are shown in Tab. 1. To be noted, although BreakAScene achieves higher visual fidelity, it shows low editing flexibility. As shown in Fig. 10, the results of BreakAScene are more like the copy-and-paste version of the source image.

Table 1: Quantitative comparison of different methods.

Methods	CLIP-I-f \uparrow	CLIP-I-b \downarrow	DINO-f \uparrow	DINO-b \downarrow	CLIP-T \uparrow	DIV \uparrow
TI	0.820	0.643	0.473	0.188	0.293	0.725
DB	0.829	0.705	0.541	0.247	0.282	0.663
TI+DB	0.874	0.752	0.570	0.275	0.279	0.652
Custom	0.824	0.662	0.540	0.222	0.293	0.715
ELITE	0.853	0.677	0.444	0.176	0.288	0.715
BreakAScene	0.920	0.745	0.686	0.295	0.276	0.525
Ours (Stage I)	0.842	0.639	0.496	0.167	0.297	0.744
Ours (Stage II)	0.898	0.655	0.629	0.230	0.296	0.681

7 USER STUDY

We also include human evaluation, which is more representative in image-generation tasks. To do so, we invite 25 participants to choose their favorite editing results in consideration of three aspects: semantic alignment with the given prompt, visual similarity of the intended concept, and the quality of the generated image. We choose 10 “face” instances and 10 “other” instances and generate editing results according to 10 editing prompts, respectively. Overall, we get $20 \times 25 = 500$ votes in total. Then, we calculate the percentage of votes and show the human preference percentages in Tab. 2. As a result, our proposed SingleInsert generates results that are most in line with human perception.

Table 2: User studies of different methods.

Class	TI	DB	TI+DB	Custom	Elite	BreakAScene	FastComposer	Ours S1	Ours S2
Faces	0.040	0.036	0.056	0.080	0.020	0.056	0.156	0.164	0.392
Others	0.020	0.036	0.044	0.040	0.052	0.036	\	0.160	0.612

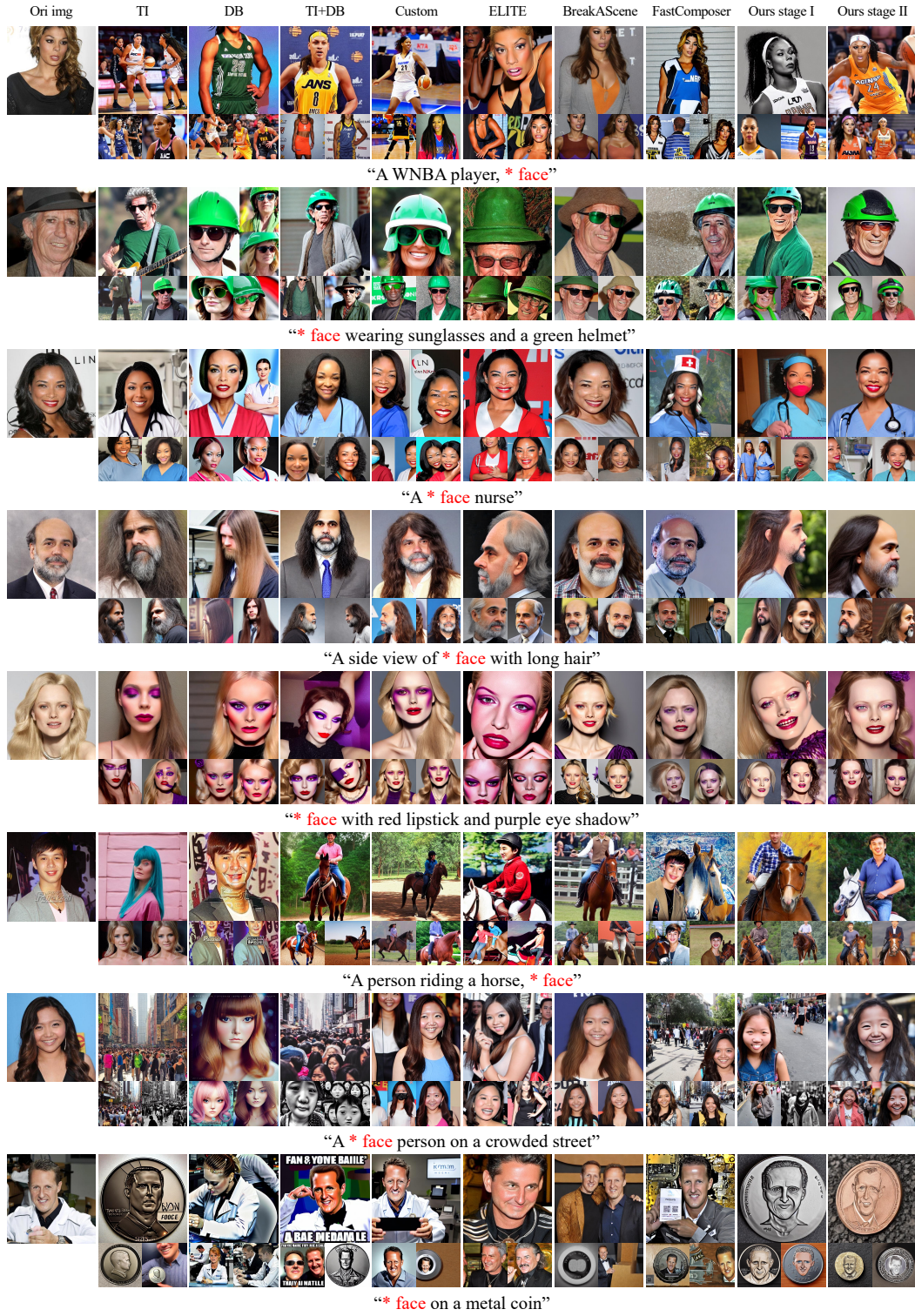


Figure 9: More qualitative comparisons on faces.

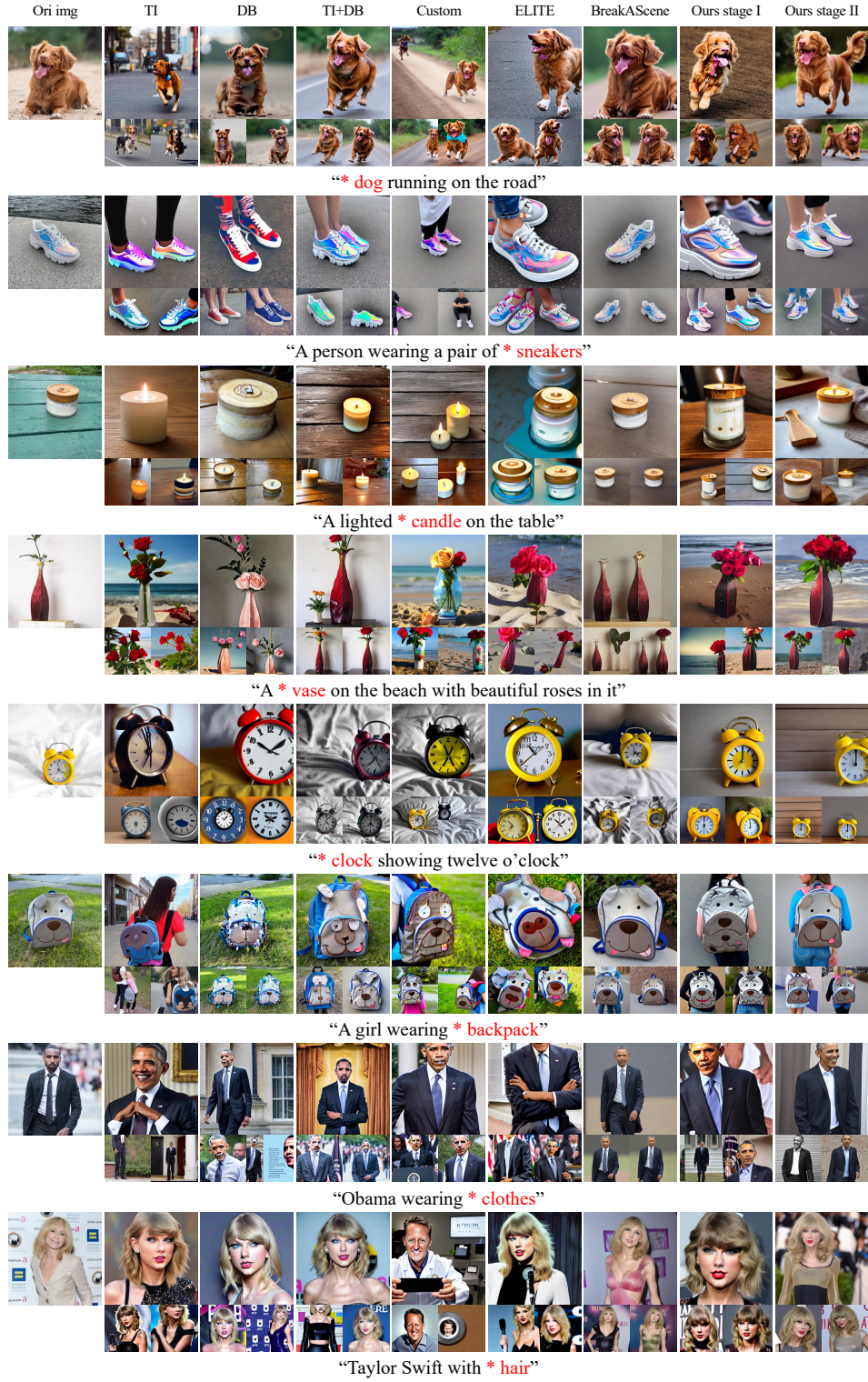


Figure 10: More qualitative comparisons on other categories.