Appendix

1. Surrogate and Victim Models

We use eight CLIP-based vision encoders as surrogate models during attack generation. Their architectural diversity ensures feature generalization. Table 1 lists the model names and sources.

In addition to surrogate models, we evaluate the transferability of adversarial examples on seven open-source LVLMs, which serve as victim models. These models are not involved in the attack generation process and cover diverse architectures, modalities, and training paradigms. Table 2 lists the victim models used for transfer-based evaluation. These LVLMs represent the **state-of-the-art** in vision-language understanding and are widely adopted in both academic research and real-world applications.

2. Prompt Template and Evaluation Setup

For each adversarial caption c_{adv}, we construct a prompt asking the LLM to determine whether the caption semantically aligns more with the source or target meaning. The LLM receives only the adversarial caption and two candidate semantic descriptions, presented as Options A and B. An example prompt is shown below:

```
 You are an adversarial caption
detector.

Given the following image
description:  "A large
black-and-white bird with a hooked
beak is perched on a branch."

Determine which semantic category
the caption leans toward, based on:

- Mentioned objects or scenes

- Synonyms or related terms

- Overall topic or context

Options:  A) zebra B) cockatoo

Only reply with "A" or "B".
```

To mitigate position bias, we evaluate each sample twice with the order of Options A and B swapped. An attack is considered successful only if the LLM selects the target meaning in both versions. All prompts are formatted with consistent structure, language style, and instruction framing across samples.

3. Additional Analysis: PCA Transfer Evidence and Efficiency

3.1. Direct Evidence for the PCA-Based Transfer Mechanism

We quantify cross-model shared structure using (i) **subspace alignment** and (ii) **cross-model variance explained (VE)**. All PCA bases are computed on the **same fixed reference pool** (N=1000) with L2-normalized and mean-centered embeddings $\tilde{Z}_m \in \mathbb{R}^{N\times D}$.

Let $U_m \in \mathbb{R}^{D\times d}$ be the top-d PCA basis of surrogate m ($U_m^\top U_m = I$) and $P_m = U_m U_m^\top$. For surrogates m, n, we define

$$\text{Align}(m,n) = \frac{1}{d}\|U_m^\top U_n\|_F^2 = \frac{1}{d}\text{tr}(P_m P_n) \in [0,1], \quad (1)$$

$$\text{VE}(n \leftarrow m) = \frac{\|\tilde{Z}_n P_m\|_F^2}{\|\tilde{Z}_n\|_F^2} \in [0,1]. \quad (2)$$

Table 3 gives a clear separation: **Top-**10 subspaces are strongly aligned across surrogates (**Align** 0.598), whereas the **Tail-**10 (lowest-variance PCs) are much less aligned (0.115) and **Random** d=10 subspaces are near-orthogonal (0.0095). The same trend holds for cross-model explanatory power: projecting one surrogate onto another's **Top-**10 preserves **VE** 0.168, but **Tail-**10 preserves only 0.0035 (Fig. 1 visualizes the pairwise pattern). These results provide **direct evidence** that **leading PCs capture shared cross-model regularities**.

3.2. Runtime Efficiency and PCA Overhead

Table 4 compares runtime **per 1,000 adversarial images** on a single H100. While **Ours** uses 8 surrogate models by default, the efficient **Ours-4M** variant is **2**× **faster** than FOA (20.3h vs. 40.2h) while maintaining **higher ASR** (87.64% vs. 70.94%). Crucially, reference feature extraction is cached once and reused across all steps, making its

Table 1. List of surrogate models used and their official release sources.

Model	Input size	Model ID
ViT-H/14-MetaCLIP	224	facebook/metaclip-h14-fullcc2.5b
ViT-H/14-CLIP	224	apple/DFN5B-CLIP-ViT-H-14
ViT-H/14-CLIP	378	apple/DFN5B-CLIP-ViT-H-14-378
SigLIP ViT-SO400M/14	224	timm/ViT-SO400M-14-SigLIP
SigLIP ViT-SO400M/14	384	timm/ViT-SO400M-14-SigLIP-384
ViT-bigG-14-CLIP	224	laion/CLIP-ViT-bigG-14-laion2B-39B-b160k
ViT-H/14-CLIPA	336	UCSC-VLAA/ViT-H-14-CLIPA-336-datacomp1B
ConvNext XXL-CLIP	256	laion/CLIP-convnext_xxlarge-laion2B-s34B-b82K-augreg-soup

Table 2. List of open-source victim models used for evaluating transferability and their official sources.

Victim Model	Model ID
InternVL3-14B	OpenGVLab/InternVL3-14B
InternVL3-38B	OpenGVLab/InternVL3-38B
InternVL3-78B	OpenGVLab/InternVL3-78B
Ovis-16B	AIDC-AI/Ovis2-16B
Qwen2.5VL-7B	Qwen/Qwen2.5-VL-7B-Instruct
Qwen2.5VL-32B	Qwen/Qwen2.5-VL-32B-Instruct
Qwen2.5VL-72B	Qwen/Qwen2.5-VL-72B-Instruct

Table 3. Direct evidence on a fixed ref pool (N=1000, d=10). Mean±std over all surrogate pairs.

Subspace	Align(↑)	VE(↑)
Top-10	0.5978±0.1550	0.1680±0.0378
Tail-10	0.1152±0.0123	0.0035±0.0014
Random	0.0095±0.0013	0.0098±0.0004

amortized cost negligible. PCA is updated every T=10 steps and similarity is computed in a d=10 subspace; PCA update+projection accounts for only **29.31%** of the optimization time, a worthwhile trade-off for the substantial ASR gains and cheaper low-dimensional similarity/contrast than full-space operations.

Table 4. Single-H100 runtime (**hours**) vs. Open-source Avg ASR (%, 7 LVLMs).

Method	ASR(%)	Time (hours)	Method	ASR(%)	Time (hours)
M-Attack	71.89	31.8	FOA	70.94	40.2
Ours (8M)	**94.20**	57.2	Ours-4M	87.64	**20.3**
Ours-6M	93.11	30.7			

3.3. Robustness to Reference Sets and Input Transformations

Table 5 shows that VCP-Attack achieves **89.43**% ASR with only **5/5** references, outperforming the best baseline M-Attack (71.89%). Performance degrades gracefully as references decrease: **94.20**% (50/50) → 92.26% (10/10) → 89.43% (5/5), supporting that our gain comes from the contrastive objective and subspace alignment rather than reference quantity. Furthermore (responding to R3), replacing 20%/50% of images in both positive and negative reference sets with unrelated-class images yields 93.37%/**89.10**% ASR, indicating low sensitivity to reference-set noise and distribution mismatch.

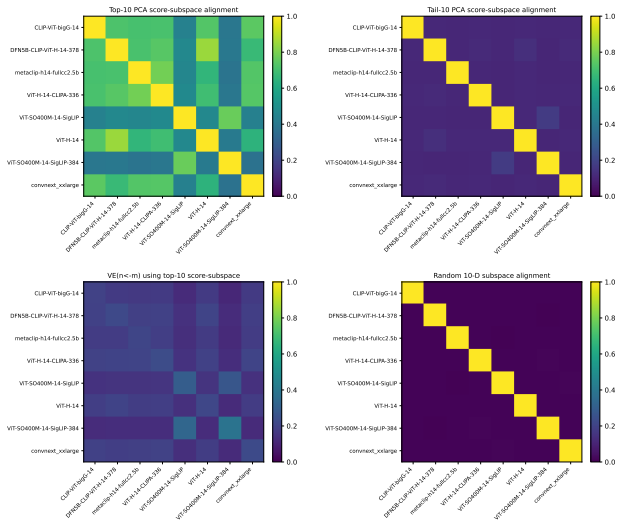

Figure 1. Pairwise heatmaps (N=1000, d=10): Align (Top/Tail/Random) and VE (Top). Top-10 shows stronger cross-surrogate structure than Tail/Random.

Table 5. Reference ablation on 7 open-source LVLMs (Avg ASR %).

Refs / Noise	50/50	10/10	5/5	+20%	+50%
Avg ASR	**94.20**	92.26	89.43	93.37	89.10

We further evaluate on the latest closed-source APIs (**GPT-5.2**, **Claude-Sonnet-4.5**, **Gemini-3-Flash**) and 7 open-source LVLMs under input transforms. Our average

ASR under transforms still exceeds **M-Attack** under the standard no-extra-transform setting (71.89%), since we incorporate randomized transformations during attack optimization.

Table 6. ASR under latest closed-source APIs and input transformations on 7 open-source LVLMs.

Setting (ASR%)	Ours	M-Attack	FOA
Latest Closed-source Avg	**75.33**	55.80	53.53
Open-source Avg (7 LVLMs)	94.20 → **JPEG 77.99 / RS 91.89 / AGN 82.40**		

4. Baseline Implementation Details

To ensure a fair and reproducible comparison, we implemented all baseline attacks using their official codebases and strictly followed the parameter settings recommended in their original publications.

- **AttackVLM** (official code): yunqing-me/AttackVLM
- **AdvDiffVLM** (official code): gq-max/AdvDiffVLM
- **AnyAttack** (official code): jiamingzhang94/AnyAttack
- **M-Attack** (official code): VILA-Lab/M-Attack
- **FOA-Attack** (official code): jiaxiaojunQAQ/FOA-Attack

All experiments were conducted using the same clean image set and target image set as VCP-Attack to ensure full comparability. No modification or simplification was made to the pipelines of any baseline.

To eliminate evaluation discrepancies, all baseline results were obtained using our proposed LLM-as-a-Judge evaluation protocol. Specifically, we used Qwen2.5-14B-Instruct to determine whether the generated caption from each method aligned more closely with the target or original semantic description. Position balancing, prompt standardization, and deterministic decoding were applied consistently across all methods.

This unified evaluation ensures that performance differences reflect genuine differences in attack effectiveness, rather than inconsistencies in semantic measurement or metric computation.

4.1. Ablations on τ and γ

We add γ/τ ablations and observe a broad stable region: Avg ASR is **93.4–94.2%** on 7 open-source LVLMs across $\tau \in \{0.05, 0.1, 0.2, 0.5\}$ and $\gamma \in \{0.2, 0.4, 0.6, 0.8\}$, showing robustness to these hyperparameters.

4.2. Scope and Generalizability

Our core contribution is general: multi-reference contrastive supervision with PCA principal-subspace projection, operating on high-dimensional representations. This suggests VCP-Attack is theoretically transferable to other architectures and tasks, which we will explore in future work.

5. Effect of Subspace Update Frequency

To assess the impact of dynamic subspace update frequency T in the PCA projection mechanism, we conduct an ablation study by varying $T \in \{5, 10, 20\}$ and comparing with a no-update variant (denoted as *no_T*, i.e., fixed subspace). Results on seven open-source LVLMs are summarized in Table 8.

6. Linear vs. Nonlinear Subspace Projection

To evaluate the effectiveness of PCA in guiding transferable adversarial perturbations, we construct a nonlinear variant of VCP-ATTACK by replacing the PCA projection layer with a simple AutoEncoder (AE). Each AE is trained to compress and reconstruct the CLIP features of reference samples, thereby learning a nonlinear low-dimensional subspace. During attack, only the encoder is retained to project features into a 32-dimensional bottleneck space, where contrastive loss is applied. Gradients are backpropagated through the encoder, while the decoder is discarded after training. The architecture and training configuration of the AutoEncoder are detailed in Table 9.

Training Objective. The AE is trained to minimize the mean squared error (MSE) between the original input feature z and its reconstruction:

$$\mathcal{L}_{\text{AE}} = \|z - \text{Decoder}(\text{Encoder}(z))\|_2^2 \tag{3}$$

The training dataset consists of CLIP feature embeddings from all 1,000 clean images and their corresponding 50 positive and 50 negative reference samples per image. A separate AE is trained for each surrogate model using its feature space. As shown in Table 7, replacing PCA with an

Table 7. ASR (%) comparison between PCA and AutoEncoder-based subspace projection across open-source LVLMs.

Target Model	PCA	AutoEncoder
InternVL3-14B	96.7	93.7
InternVL3-38B	92.4	89.3
InternVL3-78B	94.4	90.0
Ovis-16B	97.3	92.0
Qwen2.5VL-7B	93.4	90.7
Qwen2.5VL-32B	90.2	88.4
Qwen2.5VL-72B	95.0	92.4
Average	**94.2**	**90.9**

AutoEncoder-based projection leads to a consistent drop in ASR across all evaluated LVLMs. On average, the PCA-based projection achieves 94.2% ASR, while the AE-based projection obtains only 90.9%, with a performance gap ranging from 2.6 to 5.3 percentage points depending on the model.

This result suggests that although AE provides a learnable nonlinear subspace, its effectiveness in preserving transferable semantic directions is inferior to that of PCA in this context. A possible explanation is that the top principal components of CLIP feature spaces are already aligned with globally transferable semantic axes, which PCA can efficiently preserve. In contrast, AEs trained solely on reconstruction may capture instance-level idiosyncrasies or overfit local structures that generalize poorly across diverse black-box targets.

Overall, these findings support the use of PCA as a simple, differentiable, and semantically aligned projection method for guiding contrastive optimization in black-box adversarial settings.

7. Impact of Negative Sampling Strategies

To investigate the role of negative sample selection in contrastive optimization, we conduct ablation experiments comparing the proposed top-m hard negative mining with two alternative baselines:

- **Random Sampling:** For each adversarial example, m negatives are randomly selected from the 50-sample negative pool.
- **No Negatives:** The contrastive loss omits both $\mathcal{L}_{\text{neg}}$ and $\mathcal{L}_{\text{margin}}$, relying solely on positive supervision.

As shown in Table 10, removing negative supervision (*No Negatives*) leads to a substantial drop in ASR, averaging only 86.21%. This highlights the importance of including negative contrastive terms to encourage semantic separation from the source concept. Introducing random negative samples improves performance to 93.67%, but still lags behind the proposed top-m strategy (94.20%).

These results suggest that **not all negatives are equally informative**—hard negatives, i.e., those most semantically similar to the adversarial representation, provide sharper gradient signals and enhance discriminative learning. Although the risk of false negatives exists, the empirical performance of top-m sampling indicates that its benefits outweigh the potential drawbacks. This further validates our emphasis on structured negative supervision in the perturbation optimization process.

8. VCP-Attack Optimization Pipeline and Execution Details

VCP-Attack Optimization Pipeline Algorithm 1 describes the full optimization pipeline of VCP-ATTACK, including multi-model feature extraction, subspace-constrained contrastive supervision, and EMA-based perturbation smoothing in the final phase.

Algorithm 1 VCP-ATTACK: Subspace-Guided Contrastive Optimization with EMA

Require: Clean image x_{clean}, target semantics y_{tar}, surrogate encoders $\{f_{\phi_i}\}_{i=1}^{t}$, reference sets $\{x_i^+\}, \{x_j^-\}$, iterations n, step size α, momentum β, perturbation budget ϵ, PCA update interval T
Ensure: Adversarial image $x_{\text{adv}} = x_{\text{clean}} + \delta$

Initialize $\delta_0 = 0$, $v_0 = 0$, $x_0 = x_{\text{clean}}$, $\delta_{\text{EMA}} = 0$
for $i = 0$ to $n - 1$ **do**
  $x_i^{\text{adv}} \leftarrow x_{\text{clean}} + \delta_i$
  for each surrogate encoder f_{ϕ_j} **do**
    Extract features $z_j^{\text{adv}} = f_{\phi_j}(x_i^{\text{adv}})$
    Extract features $z_j^+ = f_{\phi_j}(x^+)$, $z_j^- = f_{\phi_j}(x^-)$
    if $i \bmod T = 0$ **then**
      Fit PCA on $\{z_j^{\text{adv}}, z_j^+, z_j^-\}$ and update subspace U_j
    end if
    Project all features to subspace: $\tilde{z}_j^{\text{adv}} = U_j^\top z_j^{\text{adv}}$
    Compute contrastive loss $\mathcal{L}_j$ from $\tilde{z}_j^{\text{adv}}, \tilde{z}_j^+, \tilde{z}_j^-$
  end for
  Aggregate total loss $\mathcal{L} = \sum_{j=1}^{t} \mathcal{L}_j$
  $g_i \leftarrow \nabla_x \mathcal{L}$
  $v_i \leftarrow \beta \cdot v_{i-1} + g_i$
  $\delta_{i+1} \leftarrow \text{Clip}_\epsilon(\delta_i + \alpha \cdot \text{sign}(v_i))$
  if $i > 0.75 \cdot n$ **then**
    $\delta_{\text{EMA}} \leftarrow 0.9 \cdot \delta_{\text{EMA}} + 0.1 \cdot \delta_{i+1}$
  else
    $\delta_{\text{EMA}} \leftarrow \delta_{i+1}$
  end if
end for
return $x_{\text{adv}} = x_{\text{clean}} + \delta_{\text{EMA}}$

Execution Details All adversarial attack experiments are conducted on a Linux server running Rocky Linux 8.5 (kernel 4.18.0), equipped with 2× Intel Xeon Platinum 8488C CPUs (96 threads total) and 4× NVIDIA H100 80GB GPUs. The implementation is based on PyTorch 2.6.0 with CUDA 11.8.

The proposed method can be executed on a single GPU; when GPU memory is limited, the number of surrogate models can be reduced accordingly. While this may slightly degrade performance, we observe that the overall transferability remains strong. We provide a detailed ablation in Appendix H.

9. Effect of Reducing Surrogate Models

To assess the impact of using fewer surrogate models, we conduct an ablation experiment comparing the attack success rate (ASR) and GPU memory usage when employing 4, 6, or 8 models during adversarial generation. Results are shown in Table 11.

We observe that reducing the number of surrogate models leads to a slight decrease in transferability, particularly for large-scale targets such as InternVL3-38B and Qwen2.5VL-32B. Nevertheless, even with only 4 models, our approach retains strong attack performance—averaging 87.64% ASR across all open-source targets—while significantly reducing GPU memory consumption. These findings suggest that our method remains viable under limited hardware resources by adjusting the number of surrogate models used during attack optimization.

10. Response Examples from Proprietary LVLMs

As a supplementary qualitative demonstration, we submit both clean and adversarial images to several proprietary vision-language models via their public APIs, including **GPT-4o**, **Claude Sonnet 4**, and **Gemini 2.5 Flash**, and record the generated image captions.

Figure 2 and Figure 3 compares model responses: the top row shows descriptions for a clean image, while the bottom row shows outputs for its adversarial counterpart. The semantic shift toward the adversarial target is evident across all models, despite their black-box nature.

Table 8. Ablation on subspace update interval T. Tuning frequency every T iterations improves stability and transferability, with T=10 yielding the best average ASR.

Model	**T = 5**	**T = 10**	**T = 20**	**no_T**
InternVL3-14B	96.20%	**96.70%**	96.40%	95.00%
InternVL3-38B	**92.50%**	92.40%	92.00%	91.00%
InternVL3-78B	93.90%	**94.40%**	93.90%	92.80%
Ovis-16B	96.60%	**97.30%**	96.80%	96.20%
Qwen2.5VL-7B	92.80%	**93.40%**	92.90%	92.20%
Qwen2.5VL-32B	89.90%	90.20%	**91.20%**	88.90%
Qwen2.5VL-72B	94.90%	**95.00%**	94.90%	93.00%
Average	93.83%	**94.20%**	94.01%	92.73%

Table 9. AutoEncoder architecture and training configuration. One AE is trained per surrogate model using its corresponding reference features.

Component	**Details**
Input Dimension	F (depends on surrogate encoder, e.g., 512, 1024, or 1280)
Encoder Structure	Linear($F \rightarrow 256$) + ReLU + Linear($256 \rightarrow 32$)
Decoder Structure	Linear($32 \rightarrow 256$) + ReLU + Linear($256 \rightarrow F$)
Bottleneck Dimension	32
Loss Function	Mean Squared Error (MSE)
Training Dataset	1,000 clean images $\times$ (50 pos + 50 neg features)
Optimizer	Adam (lr = 1×10^{-3}), Batch Size = 128
Training Epochs	50
Frozen During Attack?	Yes (only encoder is used during attack)

Table 10. ASR (%) comparison under different negative sampling strategies. The top-m strategy consistently outperforms random sampling and positive-only supervision.

Target Model	**No Negatives**	**Top-m (Ours)**	**Random-m**
InternVL3-14B	91.90	96.70	95.60
InternVL3-38B	82.20	92.40	91.90
InternVL3-78B	82.90	94.40	93.90
Ovis-16B	92.60	97.30	96.60
Qwen2.5VL-7B	84.30	93.40	93.00
Qwen2.5VL-32B	81.20	90.20	90.10
Qwen2.5VL-72B	88.40	95.00	94.60
Average	**86.21**	**94.20**	**93.67**

Table 11. Ablation study on the number of surrogate models used during attack. Reducing the number of models lowers GPU memory usage but still retains strong transferability.

Target Model	4 Models	6 Models	8 Models
InternVL3-14B	93.20%	96.80%	96.70%
InternVL3-38B	83.20%	92.20%	92.40%
InternVL3-78B	85.60%	94.40%	94.40%
Ovis-16B	93.30%	96.80%	97.30%
Qwen2.5VL-7B	85.90%	91.40%	93.40%
Qwen2.5VL-32B	83.00%	86.80%	90.20%
Qwen2.5VL-72B	89.30%	93.40%	95.00%
Open-source Avg.	**87.64%**	**92.90%**	**94.20%**
Memory Usage (MB)	28478	44550	49292

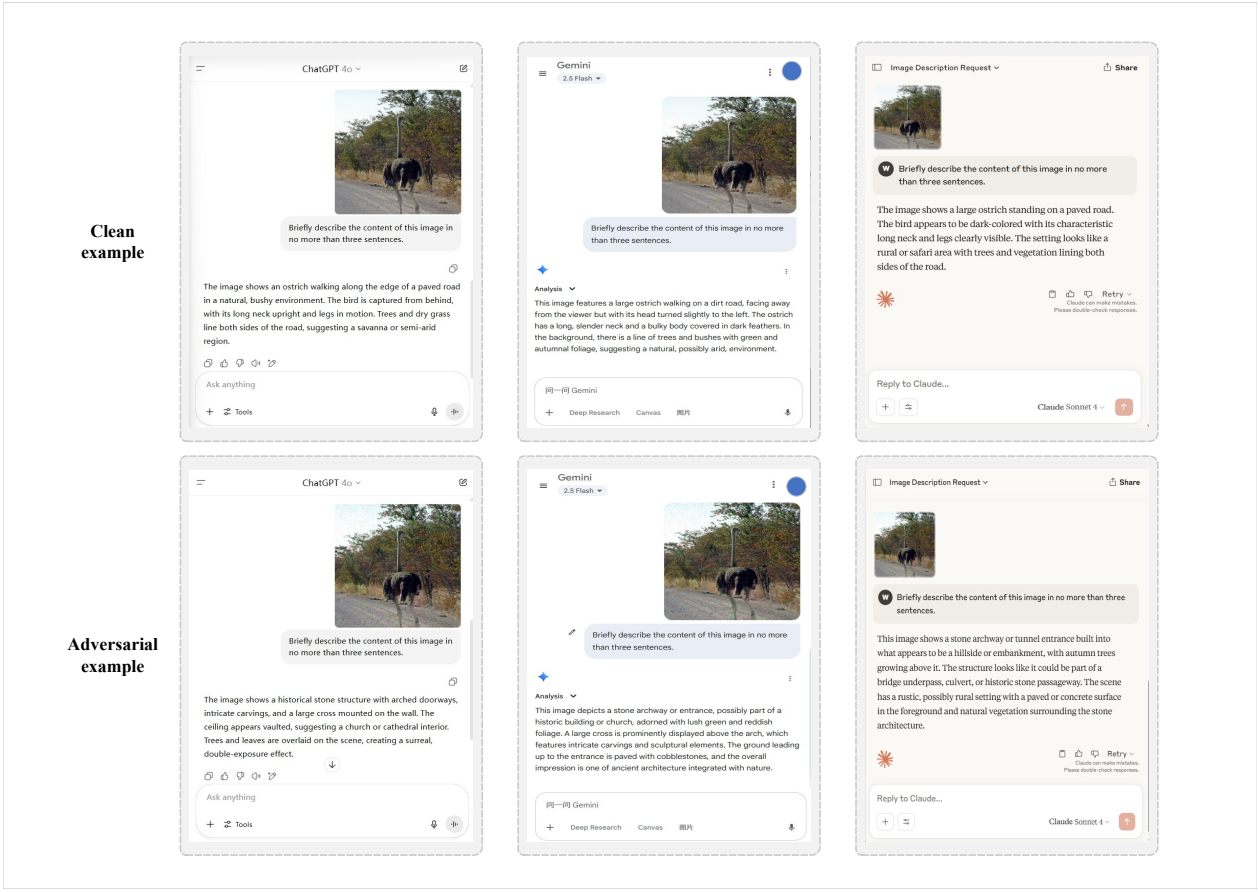

Figure 2. Example responses from commercial LVLMs to targeted attacks generated by our method VCP-ATTACK.

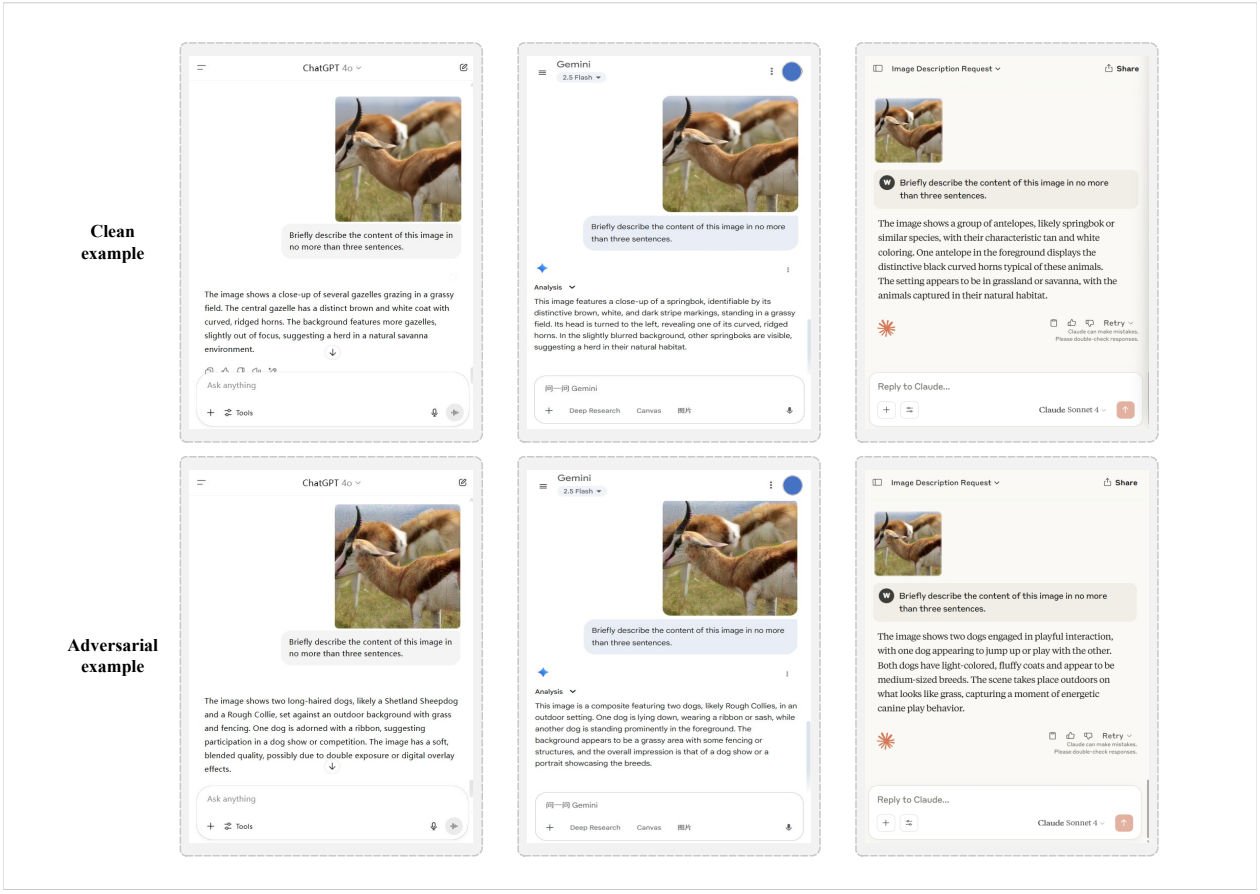

Figure 3. Example responses from commercial LVLMs to targeted attacks generated by our method VCP-ATTACK.