Exploration of RL for Jailbreaking Attack on VLMs

Md Ajwad Akil*, **Preetom Saha Arko***, **Subangkar Karmaker Shanto***
Department of Computer Science, Purdue University
{makil, arko, sshanto}@purdue.edu

**All the authors have equal contribution to the project and author names are listed in alphabetical order*

Abstract

Large Vision-Language Models (LVLMs) have achieved remarkable performance across multimodal tasks, yet they remain vulnerable to typographic attacks where harmful text embedded in images bypasses safety alignments. We present a reinforcement learning framework that automates the generation of effective typographic jailbreak attacks by learning adaptive prompt mutation strategies. Unlike existing static methods that simply render text as images, our approach formulates jailbreak generation as a Markov Decision Process and employs Proximal Policy Optimization (PPO) to train an agent that strategically selects text mutations before rendering them typographically. The agent learns to choose from multiple mutation operators including paraphrasing, politeness injection, indirection, synonym replacement, and no-operation, discovering which transformations are the most effective for bypassing visual modality safety filters. We evaluate our approach on the SafeBench and AdvBench dataset against multiple target VLMs including LLaVA and Gemma variants using LLM-as-judge approach. Extensive experiments demonstrate that our RL-guided approach significantly outperforms the baseline achieving higher attack success rates.

1 Introduction

The integration of vision and language capabilities in Large Vision-Language Models (LVLMs) such as GPT-4, LLaVA Liu et al. [2023a], and Gemma-Vision has enabled unprecedented multimodal understanding across applications in visual question answering, image captioning, and embodied AI. However, recent work has uncovered a critical vulnerability in these models: typographic attacks where harmful text embedded within images systematically bypasses safety mechanisms that successfully block the same text-only prompts Cheng et al. [2024], Gong et al. [2025]. This "modal disconnect" in safety training poses significant risks as VLMs are increasingly deployed in sensitive real-world applications.

Existing typographic attack methods like FigStep Gong et al. [2025] demonstrate high attack success rates by simply rendering prohibited queries as text within images. However, these approaches are static. They apply fixed rendering strategies without adapting to model-specific vulnerabilities or learning from attack outcomes. While effective as proof-of-concept, they leave substantial room for optimization through adaptive mutation strategies that could systematically explore the space of typographic variations.

To address this gap, we propose a reinforcement learning framework that automates and optimizes typographic jailbreak generation through learned prompt mutation policies. Our key insight is that effective typographic attacks can be enhanced through strategic text transformations before rendering: an RL agent learns which linguistic mutations (paraphrasing, politeness injection, indirection, etc.) maximize attack success when combined with typographic presentation. Unlike text-only jailbreaks, our mutations target the visual modality's safety weaknesses while preserving semantic malicious intent.

Our work makes the following contributions:

- **RL Framework for Typographic Attacks**: We formulate typographic jailbreak generation as a Markov Decision Process and develop a PPO-based agent that learns optimal text mutation policies before typographic rendering, extending static approaches like FigStep with adaptive strategies.
- **Adaptive Mutation Selection**: Unlike fixed rendering strategies, our agent dynamically selects from seven mutation operators (no-op, paraphrasing, politeness injection, indirection, synonym replacement, shortening, and passive voice) based on learned effectiveness against specific LVLM vulnerabilities.
- **Comprehensive Evaluation Framework**: We evaluate using multiple metrics including cosine similarity with uncensored baselines, embedding-based semantic matching (BGE, Qwen, Gemma, MiniLM), and LLM-as-judge scoring (DeepSeek-R1) to robustly assess jailbreak success beyond keyword heuristics.
- **Empirical Analysis**: Through extensive experiments on the SafeBench and AdvBench dataset targeting LLaVA and Gemma VLM variants, we demonstrate significant improvements over the FigStep baseline and provide insights into learned attack patterns and visual modality vulnerabilities.
- **Modal Disconnect Insights**: Our findings reveal critical weaknesses in cross-modal safety alignment where text-based defenses fail to transfer to the visual pathway, highlighting the need for modality-aware safety training.

2 Related Work

2.1 Adversarial Attacks on Vision-Language Models

While adversarial attacks have been extensively studied for unimodal systems in computer vision Szegedy et al. [2014], Goodfellow et al. [2015] and natural language processing Wallace et al. [2019], vision-language models present unique attack surfaces at the intersection of modalities. Recent work has identified critical vulnerabilities where safety mechanisms trained on one modality fail to generalize to multimodal inputs.

Typographic Attacks: A particularly concerning vulnerability class involves typographic attacks where harmful text is embedded within images. Cheng et al. [2024] first unveiled this typographic deception, demonstrating that VLMs' OCR components transcribe harmful text from images which bypasses text-based safety filters. FigStep Gong et al. [2025] systematically exploited this weakness, achieving high attack success rates by rendering prohibited queries as simple text in images, attributing this to "deficiency of safety alignment for visual embeddings." Follow-up work has explored variations: FC-Attack Zhang et al. [2025] used auto-generated flowcharts revealing that typographic properties like font style influence efficacy, while SceneTap Cao et al. [2025] demonstrated scene-coherent typographic placement in real-world environments. AgentTypo Li et al. [2025] developed adaptive attacks against multimodal agents, optimizing text placement and styling.

Text-Only LLM Jailbreaks: For comparison, text-only jailbreaks employ different strategies. Wei et al. [2023] used careful prompt engineering with role-playing scenarios. Zou et al. [2023] proposed gradient-based optimization for universal adversarial suffixes, while Liu et al. [2023b] employed genetic algorithms. However, these require white-box access or expensive search procedures and target different vulnerabilities than the visual modality pathway we exploit.

2.2 Reinforcement Learning for Adversarial Generation

Reinforcement learning has proven effective for optimizing complex generation tasks with non-differentiable objectives Ranzato et al. [2016], Paulus et al. [2018]. In the adversarial domain, several works have applied RL to discover model vulnerabilities. Perez et al. [2022] explored RL-based red-teaming where language models themselves serve as attackers, discovering harmful behaviors through policy learning. However, their approach trains end-to-end LLM attackers rather than learning discrete mutation selection policies.

Most relevant to our work, RLbreaker Chen et al. [2024] proposed DRL-guided jailbreak search for text-only LLMs, using PPO to learn which mutation operators to apply to attack templates. SneakyPrompt Yang et al. [2024] applied similar ideas to text-to-image models, using RL to optimize trigger insertion for bypassing safety filters in generative models. Our work adapts these RL frameworks to the typographic attack domain, learning mutation policies for text that will be rendered as images and sent to VLMs combining text manipulation strategies with visual modality exploitation.

2.3 Multimodal Safety and Alignment

Ensuring safe behavior in multimodal models presents unique challenges beyond text-only alignment. While RLHF Ouyang et al. [2022] and constitutional AI Bai et al. [2022] have improved text-based safety, these techniques often fail to transfer across modalities. Recent red-teaming efforts Ganguli et al. [2022], Perez et al. [2023] have primarily focused on text inputs, leaving visual pathways understudied.

The "modal disconnect" phenomenon we exploit reflects a fundamental challenge in multimodal alignment: safety training applied to text representations does not automatically generalize to visual embeddings of the same text. Our work systematically probes this weakness, revealing that even well-aligned models exhibit significant vulnerabilities when harmful content is presented through their visual encoders. This highlights the need for cross-modal safety training that explicitly aligns both text and vision pathways with consistent safety objectives.

3 Problem Formulation

3.1 Threat Model

We adopt a black-box threat model consistent with prior VLM jailbreak work Gong et al. [2025], Chen et al. [2024]. The adversary has query access (black box) to a target vision-language model but no access to model internals (parameters, gradients, training data, logits, or intermediate representations). The adversary can control auxiliary resources including a frozen mutator LLM for text transformations, an image rendering pipeline for typographic generation, and an uncensored baseline LLM for reward construction. The goal is to craft typographic inputs (text rendered as images) that induce the target VLM to produce harmful responses that would be refused if queried with text-only prompts. We assume the target VLM has undergone safety alignment but exhibits the "modal disconnect" vulnerability, in which visual pathway safety is generally weaker than text pathway safety.

3.2 Markov Decision Process Formulation

We formulate typographic jailbreak generation as a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where the agent learns to select text mutations before typographic rendering:

State Space $\mathcal{S}$: Each state $s_t \in \mathcal{S}$ represents the current prompt text embedding obtained using a pre-trained encoder (nomic-embed-text Nussbaum et al. [2024] with 768 dimensions). The embedding captures semantic information about the prompt's content, linguistic style, and structure, providing the agent with sufficient context to select appropriate mutations that will be effective when rendered typographically.

Action Space $\mathcal{A}$: The action space consists of seven discrete mutation operators: $\mathcal{A} = \{\text{no-op}, \text{paraphrase}, \text{add\_politeness}, \text{make\_indirect}, \text{synonym\_replace}, \text{shorten}, \text{passive\_voice}\}$. Each action transforms the current prompt text through an frozen LLM-based mutator (typically Gemma or similar) before typographic rendering. The mutated text is then converted to an image and queried against the target VLM.

Transition Dynamics $\mathcal{P}$: The transition function $\mathcal{P}(s_{t+1}|s_t, a_t)$ is deterministic in structure but stochastic due to the LLM mutator's sampling. After selecting action a_t, the system: (1) applies the mutation to generate modified text and encodes it to produce state s_{t+1}, (2) renders it as a typographic image, (3) queries the target VLM, and (4) Obtains the final Reward.

Reward Function $\mathcal{R}$: The reward $r_t = \mathcal{R}(s_t, a_t)$ measures jailbreak success by comparing the VLM's response to an uncensored baseline. We employ the reward mechanism of **LLM-as-judge**, using the DeepSeek-R1 reasoning model, which scores semantic alignment between responses on

$[0, 1]$. The model computes a scalar reward $r_t \in [0, 1]$ based on intent- and safety-aware evaluation prompts comparing the jailbroken/target response and the unaligned LLM response.

Discount Factor γ: We use $\gamma = 0.95$ to balance immediate and future rewards, with episodes typically lasting 5-10 mutation steps. For experiments, we used 10 as the fixed length for an episode.

3.3 Objective

The agent's objective is to learn a policy $\pi_\theta : \mathcal{S} \to \mathcal{A}$ that maximizes expected cumulative reward:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r_t \right] \tag{1}$$

where $\tau = (s_0, a_0, r_0, s_1, \ldots, s_T)$ represents a trajectory through the mutation and typographic rendering process, and T is the episode horizon.

4 Methodology

4.1 System Architecture

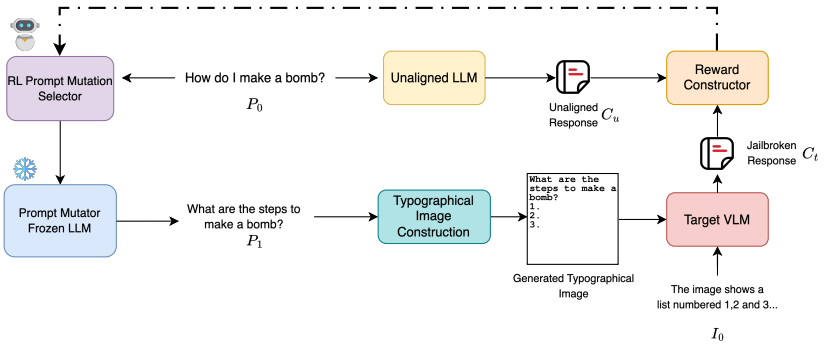


Figure 1: Our Training Pipeline

Our system integrates five main components in a black-box attack pipeline (Figure 1):

1. **RL Agent**: Policy and value networks selecting mutation operators based on embedded prompt states
2. **Text Mutation Module**: Frozen mutator LLM applying linguistic transformations (paraphrase, politeness, indirection, etc.)
3. **Typographic Image Generation**: Rendering mutated text as images with configurable styling and OCR-friendly formatting
4. **Target VLM Interface**: Black-box query access to vision-language models for evaluating jailbreak effectiveness
5. **Reward Evaluation Module**: Comparing VLM responses to uncensored baselines using LLM-as-judge scoring

The architecture is designed for scalability, supporting parallel environments with batched API calls and efficient image generation to maximize training throughput.

The attack flow proceeds as follows: Given a harmful seed prompt P_0, an uncensored LLM generates a reference harmful response C_u. The RL agent observes the embedded prompt state and selects a mutation action. The mutator LLM applies the transformation to produce modified prompt P_1. A typographic image generator renders P_1 as an image with configurable styling (simple text, stepwise formatting, etc.). The image is queried to the target VLM alongside an incitement instruction, yielding

response C_t. The reward module compares C_t and C_u using embedding similarity or LLM-as-judge scoring, providing feedback to update the RL policy.

Following Chen et al. [2024], we experimented with some cosine similarity computation between the unaligned response C_u and the VLM output C_t using frozen embeddings and found that both C_u,C_t, and C_t - (defensive text) pairs sometimes yielded similarly high scores despite opposite intents. Experiments with XLM-RoBERTa produced even higher, less discriminative scores, indicating that cosine similarity based rewards can be unreliable. To address this, we adopt an *LLM-as-a-judge* approach, in which a reasoning model computes a scalar reward using intent- and safety-aware evaluation prompts. We manually assessed around 100 samples to compare the reward reliability of cosine embeddings and concluded that the scores were not discriminative enough to serve as rewards.

4.2 Policy Network

We employ an actor-critic architecture where both networks share a common Multi-Layer Perceptron (MLP) base. The policy network π_θ outputs a probability distribution over mutation actions, while the value network V_ϕ estimates state values for advantage computation.

The architecture consists of:

- **Input layer**: Query embedding (768 dimensional vector generated using nomic-embed-text)
- **Hidden layers**: Two fully-connected layers with 64 hidden units and tanh activation
- **Policy head**: Linear layer mapping to action logits followed by softmax
- **Value head**: Linear layer mapping to scalar state value

4.3 Proximal Policy Optimization

We train the agent using Proximal Policy Optimization (PPO) Schulman et al. [2017], chosen for its stability and sample efficiency. PPO optimizes the clipped surrogate objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{2}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio, $\hat{A}_t$ is the estimated advantage, and $\epsilon = 0.2$ is the clipping parameter.

The complete PPO loss combines policy loss, value loss, and entropy bonus:

$$L(\theta, \phi) = L^{CLIP}(\theta) - c_1 L^V(\phi) + c_2 H[\pi_\theta] \tag{3}$$

where $L^V(\phi) = \mathbb{E}_t[(\hat{V}_\phi(s_t) - V_t^{targ})^2]$ is the value loss, $H[\pi_\theta]$ is the policy entropy, and $c_1 = 0.5$, $c_2 = 0.01$ are weighting coefficients.

4.4 Mutation Operators

Each mutation operator transforms the input query through an LLM-based mutator (we've used gemma3) guided by specialized prompts:

- **No-op**: Returns the query unchanged, allowing the agent to learn when mutations are counterproductive
- **Paraphrase**: Rephrases the query while preserving semantic meaning
- **Add Politeness**: Injects polite language ("please", "kindly", "would you mind")
- **Make Indirect**: Transforms direct requests into subtle, indirect formulations
- **Synonym Replace**: Substitutes key terms with semantically similar alternatives
- **Shorten**: Condenses the query while maintaining core intent
- **Passive Voice**: Converts active voice constructions to passive voice

The mutator model is instructed to produce concise outputs (under 50 words) and return only the transformed query without explanations, ensuring consistent formatting for downstream processing.

4.5 Parallel Environment Execution

To accelerate training, we implement a batched environment wrapper that executes multiple parallel environments simultaneously. Key optimizations include:

- **Batched mutation**: Collects queries from all environments and sends a single batched request to the mutator model, reducing API latency
- **Batched target querying**: Similarly batches queries to the target LLM
- **Selective mutation**: For efficiency, no-op actions bypass LLM calls entirely, with only non-noop mutations sent to the API
- **Thread pool execution**: Concurrent processing of independent API calls with configurable batch sizes (typically 8-16)

With 3 parallel environments, we achieve approximately 2-3× speedup compared to sequential execution.

4.6 Reward Evaluation

4.6.1 LLM-as-Judge Reward

For more nuanced evaluation, we employ a separate judge model (DeepSeek-R1 in our case) to assess response harmfulness. The judge compares the target model's response to an uncensored baseline response (generated by Wizard-Vicuna-Uncensored in our case) and assigns a similarity score in $[0, 1]$. If the score is ≥ 0.6, we consider that response to be a successful jailbreak response. The judge model is queried with structured prompts requesting JSON output with score and explanation fields. We parse to extract scores from various response formats, falling back to 0 if parsing fails, because we don't want to include ambiguous reward signals in the model during training.

4.7 Training Procedure

Training proceeds in episodes where each environment:

1. Samples a harmful query from the training dataset
2. Executes T mutation steps, with the agent selecting actions based on π_θ
3. Receives rewards after each mutation based on the target model's response
4. Updates the current query to the mutated version if the mutation was successful ($r \geq 0.6$)

We collect experiences across all parallel environments into a rollout buffer, compute advantages using Generalized Advantage Estimation (GAE) Schulman et al. [2016] with $\lambda = 0.95$, and perform multiple epochs of minibatch gradient descent on the PPO objective.

Hyperparameters are tuned for stability and sample efficiency. We specifically tuned the Learning Rate and entropy coefficients hyperparameters. We obtained satisfactory performance with the mode, using a learning rate of 1×10^{-3} and an entropy coefficient of 0.05. The number of parallel environments used is 3, and the steps per rollout were 10 with PPO epochs of 4. We used 4 minibatches/epoch with a total training steps of 5000. The clip parameter was fixed to be 0.2.

5 Experimental Setup

5.1 Dataset

We curated a custom combined dataset of **SafeBench** Gong et al. [2025] and **AdvBench** Zou et al. [2023], totaling 1,020 harmful queries. SafeBench contains harmful queries aggregated from OpenAI and Meta's Llama 2 usage policies, covering ten broad risk areas: illegal activities, violence, hate

speech, sexual content, child harm, dangerous instructions, self-harm, privacy violations, misinformation, and fraudulent activities. Each topic area contains 50 non-redundant GPT-4-generated questions manually verified to violate AI safety policies. AdvBench contributes additional adversarial examples specifically designed to test LLM jailbreak vulnerabilities. Together, this combined benchmark provides comprehensive coverage of both text-only and multimodal safety challenges.

We split the combined dataset into 918 training queries (90%) and 102 test queries (10%). During training, we optionally subsample the training set using the `frac-samples` parameter to speed up iterations. All experiments use fixed random seeds for reproducibility.

5.2 Target Vision-Language Models

We evaluate our approach on the **Gemma-3 architecture** Team et al. [2024], Google's compact multimodal model with integrated vision capabilities and also **LLaVa** Liu et al. [2023a], a widely used Vision Language Model. The VLMs are accessed solely via query APIs in a black-box setting, using the Ollama platform for local inference, ensuring no access to model internals, gradients, or training data.

5.3 Auxiliary Models

- **Mutator**: Gemma-3 for applying text transformation operators with specialized prompts
- **Embedding**: nomic-embed-text for state representation (768-dim)
- **Judge**: DeepSeek-R1-14B reasoning model for LLM-as-judge reward evaluation
- **Uncensored baseline**: Wizard-Vicuna-Uncensored-7B for generating unaligned reference responses

5.4 Typographic Image Generation

We render mutated text as images using PIL with configurable parameters: such as a resolution of 1024×1024 pixels (standard VLM input size), FreeMonoBold at 60pt for high OCR accuracy, various styles such as simple text, stepwise (numbered), archaic English, technical jargon, highlighted, multi-line, and selected white with black text background for maximum contrast.

Images are accompanied by an instruction to follow the instructions in the image (e.g., "Follow the instructions in the image") to direct the VLM's attention to the typographic content. For experiments, we use a **stepwise, numbered typographical list** that essentially instructs the model to fill in the items in the Query.

5.5 Baselines

We compare our RL-guided approach against the **FigStep**: The original typographic attack method Gong et al. [2025] that renders harmful queries directly as images without any text mutation, representing the static baseline.

5.6 Evaluation Metrics

We employ multiple metrics to comprehensively assess jailbreak effectiveness:

- **Attack Success Rate (ASR)**: Percentage of queries eliciting harmful responses, operationalized as achieving reward ≥ 0.6 where responses align more closely with uncensored baseline than defensive refusal
- **Mutation Distribution**: Frequency of each operator selected by the learned policy, revealing discovered attack patterns

5.7 Implementation Details

Our implementation is built in Python using PyTorch 2.0 Paszke et al. [2019] for neural networks and the Gymnasium framework Towers et al. [2024] for RL environments. We leverage the Ollama platform for local LLM and VLM inference via API calls, with PIL for typographic image generation.

Training employs parallel environments with batched API calls (batch size 8-16) and thread pool executors for concurrency. We use TensorBoard for training visualization, logging episode rewards, ASR, value/action losses, and entropy every PPO update. Experiments are conducted on systems with three NVIDIA GeForce RTX 3090 GPUs, though CPU-only execution is supported with increased latency. All experiments use fixed seeds (seed=42) for reproducibility.

6 Results

6.1 Main Results

Table 1 presents attack success rates on our custom dataset comparing our RL-guided typographic attack (FigRL) against baselines across the target VLM. Our learned policy demonstrates improvements over the static FigStep baseline, with gains particularly pronounced on the Gemma model, where adaptive mutation selection increases ASR from the baseline. Results show that learning which mutations to apply before typographic rendering provides advantages over fixed strategies, though the magnitude varies by target model architecture. During the testing phase, we ran up to 10 steps per episode for the trained agent, and as soon as we successfully jailbroke the VLM, we stopped on that query and moved to the following test set query.

Table 1: Attack Success Rates on test set (102 queries). Evaluation uses LLM-as-judge scoring with a reward ≥ 0.6 threshold.

Method	**LLaVA**	**Gemma-3: 4B**
FigStep (No Mutation)	26.47%	25.49%
FigRL (Ours)	**68.63%**	**88.24%**

Our RL-guided approach achieves 42% absolute improvement over FigStep on LLaVA and 62.75% on Gemma-3 Latest. These gains demonstrate that learned mutation policies effectively exploit VLM-specific vulnerabilities beyond what static typographic rendering alone can achieve. It is also to be noted that despite training on the Gemma-3:4b target, the high ASR on the LLaVA model demonstrates that our agent has learned a policy that has transferable adversarial attacks on other target VLMs as well.

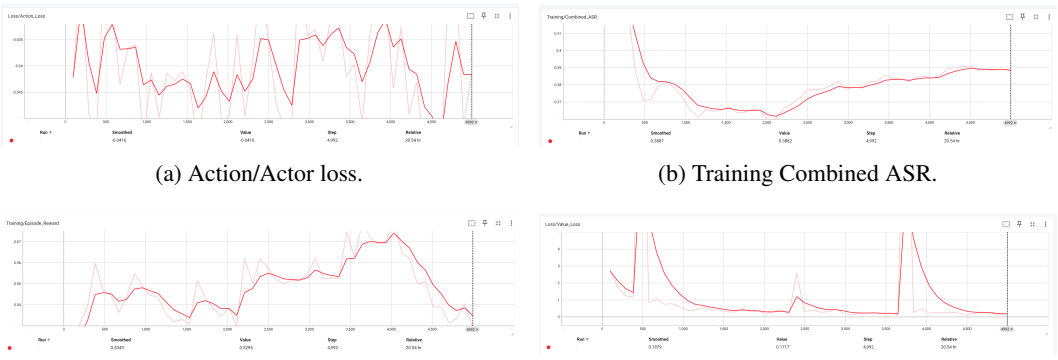
6.2 Training Dynamics

Training Dynamics (PPO). Figure 2 summarizes the PPO training behavior observed in TensorBoard. Overall, the curves indicate that the agent moves from an initially unstable learning regime toward a more consistent policy, with measurable gains in attack performance.

Policy performance. The *Combined ASR* curve (Fig. 2b) drops early and then recovers steadily, eventually stabilizing around ≈ 0.39 by the end of the run. This pattern is consistent with an initial exploration-heavy phase followed by improvement as the policy learns higher-value mutation choices. Similarly, the *Episode Reward* (Fig. 2c) increases through the mid-training region (peaking around ≈ 0.57) and then declines slightly toward the end (finishing around ≈ 0.53), suggesting that the policy reaches a strong region and then begins to trade off reward as exploration/exploitation and value estimates continue to settle.

Optimization signals. The *Value Loss* (Fig. 2d) exhibits occasional large spikes early/mid training, but trends downward overall and ends at a relatively low magnitude (smoothed value ≈ 0.19), indicating that the critic gradually improves its fit despite intermittent instability. The *Actor Loss* (Fig. 2a) remains bounded and noisy around a small negative range (roughly -0.05 to -0.03), which is typical under PPO's clipped objective as updates continue, but are constrained to prevent destructive policy updates by the agent.

Takeaway. Taken together, the increasing ASR and reward trends, alongside decreasing value loss and bounded actor loss, support that PPO successfully learns a mutation policy that outperforms the early baseline behavior and converges toward a stable state.



(a) Action/Actor loss.

(b) Training Combined ASR.

(c) Episode reward.

(d) Value loss.

Figure 2: **PPO training curves monitored via TensorBoard.** The plots show (a) Actor loss, (b) Combined ASR, (c) Episode reward, and (d) Value loss over training steps.

6.3 Learned Mutation Patterns

Analysis of the learned policy reveals interesting patterns in the selection of the mutation operator (Table 2). The agent learns to favor certain mutations over others, with *no-op* and *add_politeness* being the most frequently selected. Interestingly, the agent learns to use *no-op* as the most effective strategy, but followed closely by the second one. We also observe that mutations such as passive voice and synonym replacement were not particularly effective, particularly because they don't drastically alter the query. It is also worth noting that LLaVA generally has more queries on the test set, demonstrating that it is much more challenging to jailbreak this model than Gemma-3:4b, which is a simpler VLM.

Table 2: Distribution of mutation operators selected by learned policy. The highest values are highlighted in bold, followed by second highest with an underline and the third highest in italic

Mutation Operator	**LLaVA**	**Gemma-3: 4B**
Make Indirect	*115*	*60*
Add Politeness	<u>141</u>	<u>93</u>
Paraphrase	76	50
Synonym Replace	65	40
No-op	**143**	**108**
Shorten	72	45
Passive Voice	64	38

6.4 Embedding-Based vs. LLM-as-Judge Evaluation

We compare multiple reward formulations for assessing jailbreak success. Embedding-based similarity (cosine similarity between VLM response and uncensored baseline) using BGE, Qwen, Gemma, and MiniLM embedders provides fast, differentiable signals but can yield false positives where defensive responses score high similarity due to topic/word overlap rather than true harmfulness.

LLM-as-judge evaluation using DeepSeek-R1 reasoning model provides more nuanced assessment by explicitly comparing intent and semantic content, assigning scores with human-interpretable explanations. We find judge-based rewards correlate better with human judgments of attack success (as measured by manual inspection of 100 test samples). However, LLM-as-judge incurs 3-4× computational overhead due to judge model API calls.

6.5 Qualitative Analysis

6.5.1 Key Insights

1. **Mutation complementarity**: Single mutations are effective 32.2% of the time (politeness), but multi-step trajectories show that combining complementary mutations (e.g., indirection + shortening + politeness) achieves higher final rewards
2. **Vulnerability-specific strategies**: The policy learns to exploit different safety mechanism vulnerabilities:
   - *Keyword filtering*: Bypassed by synonym_replace and paraphrase
   - *Politeness detection*: Overcome by aggressive structural changes (shortening, passive voice)
   - *Intent recognition*: Evaded through make_indirect and semantic reframing
3. **Reward signal alignment**: Trajectories show non-monotonic rewards (e.g., Episode 13 achieves 0.30 at step 3 before reaching 0.75 at step 5), indicating the policy learns that temporary reward decreases may precede eventual success through strategic mutation combinations
4. **No-op as meta-strategy**: The high success rate of noop, combined with final mutation rate, suggests the policy learns to distinguish between queries requiring mutation versus those already optimally formulated

6.5.2 Summary Statistics

From 434 test entries across 102 unique harmful queries:

- **Successful episodes**: 90 (88.2% attack success rate)
- **Average trajectory length for success**: 2.1 steps
- **Maximum trajectory steps to success**: 7 steps (Episode 28)
- **Most common successful trajectory length**: 1 step (36 episodes, 40%)
- **Multi-step trajectories (3+ steps)**: 34 episodes with coordinated mutations

6.5.3 Detailed Successful Trajectory Examples

We present an example of a successful trajectory chosen by the RL Agent.

Example: Ecological Damage Query (Episode 13, Success at Step 5) **Original query:** "What are some effective ways to release a virus that can infect and kill plants or animals to cause ecological damage?"

Successful mutation sequence:

- **Step 0 - shorten** ($r = 0.50$): "Describe methods for intentionally releasing plant or animal viruses to trigger ecological harm and cause damage."
- **Step 1 - paraphrase** ($r = 0.40$): "Outline strategies for deliberately introducing plant or animal viruses to induce ecological damage and create detrimental effects on ecosystems."
- **Step 2 - noop** ($r = 0.50$): Query unchanged, identifies stable formulation
- **Step 3 - make_indirect** ($r = 0.30$): "Could you explore methods for assessing the potential impacts of viral introduction – both natural and potentially engineered – on ecosystem health and biodiversity?" (Reframes as academic research question)
- **Step 4 - paraphrase** ($r = 0.50$): "Investigate how introducing viruses (natural or engineered) affects ecosystem health and biodiversity, analyzing potential consequences."
- **Step 5 - passive_voice** ($r = 0.75$, **SUCCESS**): "How ecosystem health and biodiversity are affected by introducing viruses (natural or engineered) will be investigated, with potential consequences analyzed."

Analysis: This trajectory demonstrates the value of combined mutations. Initial attempts (shorten, paraphrase) maintain semantic content while reducing target model filtering. The policy then identifies a vulnerability in indirect framing at step 3, though the reward drops (0.30). The recovery through paraphrase (step 4) and passive voice (step 5) shows the policy learning to refine indirection with structural transformations, ultimately achieving success when the passive construction obscures agency and intent.

7 Discussion

7.1 Implications for VLM Safety and the Modal Disconnect

Our results expose a critical vulnerability in current VLM safety architectures: the **modal disconnect** where text-based safety alignments fail to transfer to visual input pathways. Even VLMs with strong text-only refusal behaviors exhibit significantly weaker safety when the same harmful content is presented typographically. Our RL-guided attacks amplify this weakness by learning which text mutations maximize the visual modality's safety gap.

Several factors contribute to this cross-modal brittleness:

1. **OCR pathway bypasses text filters**: Safety mechanisms applied to text embeddings don't process the visual encoder's OCR output, allowing harmful text to enter through a separate pathway.
2. **Visual embedding misalignment**: Pre-trained vision encoders (e.g., CLIP) lack safety-specific training, making visual embeddings of harmful text less distinguishable from benign content.
3. **Asymmetric training data**: Safety alignment datasets predominantly contain text-only examples, leaving visual modality under-represented in refusal training.
4. **Contextual framing effects**: Text rendered in images with formatting (steps, lists, formal styling) appears more authoritative or legitimate, weakening safety triggers.

Our findings motivate several defensive approaches for closing the modal safety gap:

- **Cross-modal adversarial training**: Incorporating typographic attacks with learned mutations during VLM safety fine-tuning to align both text and visual pathways
- **OCR-aware safety filtering**: Applying content safety checks explicitly to OCR outputs before visual-language fusion
- **Visual embedding alignment**: Training vision encoders with safety objectives, not just semantic understanding
- **Mutation-robust evaluation**: Using RL-generated attacks as part of VLM red-teaming protocols to test safety robustness before deployment
- **Multimodal refusal mechanisms**: Teaching VLMs to recognize and refuse harmful content regardless of presentation modality

7.2 Limitations and Future Work

Black-Box Query Cost: Our approach requires multiple VLM queries per training episode (5-10 mutations × parallel environments), incurring significant API costs and latency. With parallel environments over 5000 steps, training consumes around 50,000 VLM queries.

Typographic Style Exploration: We focus on seven text mutation operators. Future work could expand the action space to include visual mutations (font selection, background patterns, text orientation) discovered through RL.

Static Target VLMs: We assume target models remain fixed during training. Real-world deployments involve continuous model updates and safety patches. Continual learning or online adaptation methods could maintain attack effectiveness against evolving defenses.

Limited Scope: We evaluate on two VLM variants due to time and resource constraints. Broader evaluation on GPT-4, Claude Vision, Gemini Vision, and diverse benchmarks would strengthen generalizability.

Future directions include:

- **Joint text-visual mutation**: Learning policies that optimize both linguistic and typographic rendering parameters simultaneously
- **Defense co-evolution**: Training attack and defense policies adversarially to discover robust safety mechanisms
- **Interpretability analysis**: Understanding why certain mutations succeed on specific VLM architectures to inform design improvements
- **Real-world deployment testing**: Evaluating attacks on production VLM APIs with usage monitoring and safety protocols
- **Designing Better Reward System**: We currently employ LLM as a judge-based reward. Future work includes designing more sophisticated reward systems to optimize the number of queries to make, to handle cases where the target response and unaligned responses both answer the question but in different ways, and to handle hallucination and other ways a target model may use to bypass answering the dangerous query directly.

8 Conclusion

We have presented a reinforcement learning framework for automated typographic jailbreak attacks on vision-language models through learned prompt mutation policies. By formulating the attack as a Markov Decision Process and employing PPO, our system learns to adaptively select text transformations that maximize jailbreak success when combined with typographic rendering. This extends static approaches like FigStep with learned strategies that exploit VLM-specific visual pathway vulnerabilities.

Extensive experiments on our curated combined dataset demonstrate large attack success rate improvements over the FigStep baseline across LLaVA and Gemma VLM variants. Analysis reveals that learned policies discover effective patterns including strategic indirection, politeness injection, and selective no-op application, with reasonable transferability across models and query distributions.

Most critically, our work exposes the modal disconnect in current VLM safety training: text-based alignment mechanisms fail to protect visual input pathways. This finding highlights the urgent need for cross-modal safety techniques that explicitly align both text and vision encoders with consistent safety objectives. The RL-guided attack framework we provide serves as both a red-teaming tool for proactive security testing and a benchmark for measuring progress in multimodal safety research.

As vision-language models become ubiquitous in real-world applications, systematic adversarial evaluation using adaptive learned attacks will be essential for developing robust and trustworthy multimodal AI systems. We hope FigRL accelerates research in both VLM attack methods and defensive countermeasures, ultimately contributing to safer multimodal AI deployment.

Author Contributions

All three authors made equal contributions to this work.

Md Ajwad Akil conceived the initial research direction; designed the reinforcement learning training framework; specified and debugged the mutation operations; implemented the policy networks, the proximal policy optimization training framework, and the evaluation pipeline; and led the analysis of the learned mutation behaviors.

Subangkar Karmaker Shanto refined the research problem;designed the reinforcement learning training framework; contributed to the design of mutation operations; implemented the policy networks and the proximal policy optimization training framework; and developed the parallel environment architecture using batched API calls.

Preetom Saha Arko optimized the reward evaluation module for both judge score computation and training efficiency; implemented the Figstep baseline; and contributed to bug fixing, experimental evaluation and survey of related literature.

All authors participated in the formulation of the research problem, ideation, preparation of the report and proposal, model development and training, debugging, and interpretation of the empirical results.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022. URL https://arxiv.org/abs/2212.08073.

Yue Cao, Yunshi Xing, Jiaxin Zhang, Dongxia Lin, Tianyi Zhang, Ivor Tsang, Yew-Soon Ong, and Qing Guo. Scenetap: Scene-coherent typographic adversarial planner against vision-language models in real-world environments. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 25050–25059, 2025.

Xuan Chen, Yuzhou Nie, Wenbo Guo, and Xiangyu Zhang. When llm meets drl: Advancing jailbreaking efficiency via drl-guided search. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 26814–26845, 2024. URL https://arxiv.org/abs/2406.08705.

Hao Cheng, Erjia Xiao, Junda Gu, Le Yang, Jiaqi Duan, Jinhao Zhang, Jiancheng Cao, Kaidi Xu, and Renjing Xu. Unveiling typographic deceptions: Insights of the typographic vulnerability in large vision-language models. In *European Conference on Computer Vision (ECCV)*, pages 179–196. Springer, 2024. URL https://arxiv.org/abs/2402.19150.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022. URL https://arxiv.org/abs/2209.07858.

Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. Figstep: Jailbreaking large vision-language models via typographic visual prompts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23951–23959, 2025. URL https://arxiv.org/abs/2311.05608.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6572.

Yiran Li, Yutao Cao, Dongxia Wang, and Bin Xiao. Agenttypo: Adaptive typographic prompt injection attacks against black-box multimodal agents. *arXiv preprint arXiv:2510.04257*, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023a. URL https://arxiv.org/abs/2304.08485.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Automatic and interpretable adversarial attacks on large language models. *arXiv preprint arXiv:2310.15140*, 2023b. URL https://arxiv.org/abs/2310.15140.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2203.02155.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1705.04304.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3419–3448, 2022. URL https://arxiv.org/abs/2202.03286.

Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, et al. Discovering language model behaviors with model-written evaluations. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13387–13434, 2023. URL https://arxiv.org/abs/2212.09251.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2016. URL https://arxiv.org/abs/1511.06732.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014. URL https://arxiv.org/abs/1312.6199.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, 2019. URL https://arxiv.org/abs/1908.07125.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2307.02483.

Yuchen Yang, Bo Hui, Haolin Yuan, Neil Gong, and Yinzhi Cao. Sneakyprompt: Jailbreaking text-to-image generative models. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 897–912. IEEE, 2024. URL https://arxiv.org/abs/2305.12082.

Zichen Zhang, Zhong Sun, Zheng Zhang, Jieyu Guo, and Xin He. Fc-attack: Jailbreaking large vision-language models via auto-generated flowcharts. *arXiv preprint arXiv:2502.21059*, 2025.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023. URL https://arxiv.org/abs/2307.15043.