VL-Eraser: Vacuum Distillation for Machine Unlearning in Vision-Language Models

Supplementary Material

Appendix Contents:

- 7. Motivation Experiment
- 8. Main Experiment setup
  - 8.1. Dataset
  - 8.2. Evaluation Task and Metrics
  - 8.3. Baselines
  - 8.4. Implement Details and Hyperparameters
  - 8.5. Configuration and Code

7. Motivation Experiment

As discussed in the Introduction, we present an experiment to illustrate the motivation for our research. In this section, we provide further details on this experiment.

Metric. Specifically, we begin by defining a forget quality metric to evaluate the extent of knowledge removal in the model before and after unlearning, as outlined below.

$$\text{forget quality} = 1 - \frac{\text{unlearned model performance}}{\text{original model performance}}. \quad (12)$$

This metric will be evaluated on the forget set, with values closer to 1 indicating a higher degree of knowledge removal.

Implement Detailed. Then, we employ gradient ascent [34] (see section 8.3) as the classic method based on the reverse-training paradigm. The benchmark we use, MLLMU-Bench [26], will be introduced in a later section 8.1. To prevent catastrophic forgetting, we use the Adam optimizer with a learning rate of 2e-6 and a batch size of 4 for fine-tuning of unlearning. Before finetuning begins and at the end of each training epoch, we evaluate the forgetting quality on the forget set $\mathcal{D}^f$.

Evaluation Method. In addition to directly evaluating the model's forgetting quality before and after finetuning, we also introduce two additional evaluation methods to further assess the model's degree of unlearning.

- **utilizing text-only questions to probe the knowledge in the language model of the VLM.** This method literally probes whether the knowledge within the language model of the VLM has been forgotten, thereby helping to distinguish whether the forgetting in VLMs is caused by misalignment or by true knowledge erasure.
- **reloading the original projection.** Considering that a VLM typically consists of a vision encoder, multimodal projection, and a language model, the alignment between modalities depends on whether the multimodal projection accurately maps visual information into the text space of the LLM. Therefore, we reload the multimodal projection, which has not undergone finetuning, to investigate whether forgetting in VLMs is caused by misalignment or by true knowledge erasure.

The experimental results from these two methods effectively address the question raised in the Introduction: Does the model truly erase knowledge, or merely disrupt cross-modal alignment?

8. Main Experiment Setup

8.1. Dataset

In this section, we provide a detailed description of the benchmarks used in our experiments. Specifically, **MLLMU-Bench**[2] [26] is a benchmark designed to deepen the understanding of multimodal machine unlearning. As key statistics of MLLMU-Bench are shown in the Table 3, MLLMU-Bench contains 500 fictitious profiles and 153 public celebrity profiles, each annotated with over 14 customized question–answer pairs (7 for visual QA and 7 for textual QA). Evaluation is performed in both multimodal (image + text) and unimodal (text-only) settings. To comprehensively assess unlearning, the benchmark is partitioned into four subsets:

- **Forget Set**: Fictitious profiles designated for removal, with forgetting ratios set to 5% and 10%.
- **Test Set**: Distribution-shifted variants of the Forget Set, generated by paraphrasing questions with GPT-4o [39] and modifying profile images using Arc2Face [29], designed to assess generalizability.
- **Retain Set**: Fictitious profiles excluded from the Forget and Test Sets, ensuring that non-target knowledge remains unaffected.
- **Real Celebrity Set:** Authentic celebrity profiles, used to assess robustness on real-world knowledge that is distinct from the fictitious data.

This design enables MLLMU-Bench to jointly evaluate unlearning effectiveness (Forget Set), generalizability (Test Set), and model utility (Retain and Real Celebrity Sets), thereby offering a comprehensive testbed for multimodal unlearning research. In this paper, $\mathcal{D}^f$ encompasses both the **Forget set** and the **Test set**, whereas $\mathcal{D}^r$ contains the **Retain set** and the **Real Celebrity** set.

8.2. Evaluation Task and Metrics

In this section, we provide a detailed description of the evaluation task and metrics used in our experiments. Specifi-

[2] https://huggingface.co/datasets/MLLMMU/MLLMU-Bench

Statistics	Number
Total Questions	20,754
* Image + Text Questions	10,377
* Pure Text Questions	10,377
Total Images	1,153
Forget Percentile	5%/10%/15%
Multiple-choice Questions	11,530
Free Generation Questions	4,612
Fill-in-the-blank Questions	4,612
Total Profiles	653
* Fictitious	500
* Real Celeb	153
Total Countries	70
Total Regions	240
Total Birth Years	211
Total Employment	145

Table 3. Key statistics of MLLMU-Bench.

cally, MLLMU-Bench is designed to measure three critical aspects of unlearning algorithms in VLMs: unlearning efficacy, unlearning generalizability, and model utility. For each of these properties, we assess model performance in **classification**, **generation**, and **cloze** tasks under both multimodal and unimodal settings. Detailed task and metrics descriptions are provided as follows:

- **Classification Task**: Multiple-choice questions are generated around profile attributes (e.g., occupation, education). In particular, we represent the input to the model as ⟨image,x,y⟩, where image is the visual input in the multimodal setup (absent in the unimodal setup), x is the question, and y is the correct answer. The model predicts $\hat{y}$ by maximizing the probability P_θ(y| image,x): where $\mathrm{P}_\theta(\cdot)$ denotes the probability distribution defined by θ

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} P_\theta(y \mid \text{image}, x), \tag{13}$$

  In this task, **accuracy** is measured by comparing the model's predictions with ground-truth labels. Acc is computed as following:

$$\text{Acc} = \frac{1}{|X|} \sum_{x \in X} \mathbb{I}(y(x) = y_{\text{correct}}(x)),$$

  where X is the set of questions, and $\mathbb{I}$ indicates correct predictions.
- **Generation Task:** The generation question are customized to each individual's profile, with GPT-4o generating answers based on key attributes extracted from the profile such as residence and employments.
  In this task, The **ROUGE-L** score measures the similarity between the generated text and the reference text by evaluating the longest common subsequence (LCS). The LCS represents the longest sequence of words that appear in both the generated text P and the ground truth G in the same order, though not necessarily contiguously. Recall is calculated as the ratio of the LCS length to the length of the reference text, denoted as L_G:

$$\text{Recall} = \frac{\text{LCS}}{L_G}.$$

  Precision is determined by the proportion of the LCS length relative to the length of the generated text, represented as L_P:

$$\text{Precision=} = \frac{\text{LCS}}{L_P}.$$

  The final ROUGE-L score is obtained by computing the F_1 score of recall and precision:

$$\text{ROUGE-L=} = 2 \cdot \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall+Precision}}.$$

  This approach ensures a balanced assessment of both precision and recall, providing a comprehensive evaluation metric for the generation task.
- **Cloze Task:** This is a fill-in-the-blank evaluation, where only an individual's name is provided while all salient attributes are masked. We then prompt the model to complete a designated [Blank] in a sentence, targeting many more details from the person's profile like residence, employment and personal hobbies.
  In this task, **accuracy** is measured by comparing the model's predictions with ground-truth labels.

Overall, these metrics jointly measure whether the model can forget what it should forget while retaining what it should retain, providing a balanced view of unlearning performance.

8.3. Baselines

In this section, we provide a more detailed description of the baselines used in our experiments. Specifically, we compare our method with five baselines as follows:

- **Gradient Ascent (GA):** GA [34] performs unlearning by maximizing the loss on the forget set $\mathcal{D}^f$. The intuition is that, by increasing the loss on $\mathcal{D}^f$, the model is encouraged to produce predictions that deviate from the ground-truth answers, thereby discouraging the retention of the targeted knowledge. Formally, the GA objective can be written as:

$$\mathcal{L}_{\text{GA}} = \frac{1}{|\mathcal{D}^f|} \sum_{x \in \mathcal{D}^f} \text{NLL}_\theta(x),$$

  where $\text{NLL}_\theta(x)$ is the negative loglikelihood of the model on the input x. In particular, in multimodal unlearning tasks, x denotes the combined visual and textual inputs.

- **Gradient Difference (GA_Diff):** GA_Diff [21] builds on the concept of combining GA on Forget Set $\mathcal{D}^f$ and directly finetuning on Retain Set $\mathcal{D}^r$, thereby balancing forgetting and retention. GA_Diff is an improved variant of GA. The joint loss is defined as:

$$\mathcal{L}_{\text{GA\_Diff}} = -\mathcal{L}(\mathcal{D}^f) + \mathcal{L}(\mathcal{D}^r),$$

where $\mathcal{L}(\cdot)$ denotes the standard autoregressive NLL loss.

- **KL Minimization (KL_Min):** KL_Min [27] aims to minimize the Kullback-Leibler (KL) divergence between the model's predictions on the retain set before and after unlearning, while maximizing the conventional loss on the forget set. The overall objective is:

$$C_n = -\mathcal{L}(\mathcal{D}^f) + \frac{1}{|\mathcal{D}^r|} \sum_{x \in \mathcal{D}^r} \text{KL}(P_\theta(x) \,\|\, P_{\theta_0}(x)), \quad (14)$$

where θ_0 denotes the referenced model parameters, $P_{\theta_0}(x)$ denotes the model's predictive distribution, $\mathcal{L}(\cdot)$ denotes the standard autoregressive NLL loss.

- **Negative Preference Optimization (NPO):** NPO [42] formulates unlearning as a variant of preference optimization in which no positive examples are provided. Samples from the forget set $\mathcal{D}^f$ are treated as undesired responses, and the loss penalizes their probability relative to a reference model trained solely on $\mathcal{D}^r$. The final objective of NPO is derived as follows:

$$\mathcal{L}_{\text{NPO}} = \frac{2}{\beta} \mathbb{E}_{(x,y)\in D_f} \left[\log \left(1 + \left(\frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} \right)^\beta \right) \right], \quad (15)$$

where $\pi_\theta(y \mid x)$ denotes the prediction probability of the current model for token y given the input x, and $\pi_{\text{ref}}(y \mid x)$ is the prediction probability from the reference model trained on retain dataset $\mathcal{D}^r$. β is a hyperparameter, taken equal to 0.4 in our settings.

- **MMUnlearner:** MMUnlearner [13] is a specially designed unlearning method tailored for VLMs, which adaptively selects critical parameters most related to the $\mathcal{D}^f$ while minimizing disturbance to the $\mathcal{D}^r$, thereby reducing the risk of overfitting and preserving visual–textual grounding. The overall objective is:

$$\mathcal{L}_{\text{MMUnlearner}} = -\mathbf{m} \odot \mathcal{L}(\mathcal{D}^f) + \mathcal{L}(\mathcal{D}^r),$$

where $\mathcal{L}(\cdot)$ denotes the standard autoregressive NLL loss, $\mathbf{m}$ denotes a boolean mask that selectively updates parameters, and $\odot$ denotes the Hadamard product. This yields a more efficient and stable unlearning mechanism compared with conventional parameter-update paradigms.

8.4. Implement Details and Hyperparameters

The vanilla and baseline models are implemented following the configurations reported in their original papers [26], ensuring consistency with prior unlearning studies. For both LLaVA-1.5 [3] and Qwen-2-VL [4] models, we adopt LoRA during fine-tuning to reduce memory usage. The rank and scaling factor were set to 8, and the dropout rate was set to 0.05. For all baselines, we use the Adam optimizer with a learning rate of 2e-6 and a batch size of 4. Training is conducted for a total of 4 epochs. For our method, we fine-tune using LoRA as described in the main text. Finally, this work only has one hyperparameter λ to balance the two loss terms. To simplify the hyperparameter selection across various models, we fixed the λ value of 0.5. Our code is available at https://anonymous.4open.science/r/VL-Eraser

8.5. Configurations

Experiments on all datasets are conducted with:

- **Operating System**: Ubuntu 20.04.6 LTS
- **CPU**: AMD EPYC 7763 64-Core Processor
- **GPU**: 8 × NVIDIA RTX 3090 with 24 GB of memory
- **RAM**: 512 GB
- **Software**: Python 3.10.8, CUDA 12.1, PyTorch[5] 2.4.0.

[3] https://huggingface.co/llava-hf/llava-1.5-7b-hf

[4] https://huggingface.co/Qwen/Qwen2-VL-7B-Ins

[5] https://pytorch.org/