A Artifact Appendix

A.1 Abstract

Our artifact primarily implements and evaluates a reference-based membership inference attack pipeline against vision-language models (VLMs), with and without neuroscience-inspired topographic regularization (τ). It includes our codebase for (i) fine-tuning three VLMs (BLIP, PaliGemma 2, and ViT-GPT2) with different τ-regularization levels, (ii) generating captions for member and non-member image sets, (iii) computing semantic and lexical similarity scores, and (iv) performing black-box membership inference attacks. The codebase is written in Python and uses PyTorch, Hugging Face Transformers, and `sentence-transformers`. For small-scale tests, the code can run on a CPU. However, it is best to run the experiments on a GPU. We provide a conda environment setup example in the README, along with step-by-step instructions and example commands to reproduce the experiments. We expect to reproduce the results shown in the paper, although there may be differences in dataset shuffling and hardware. There may also be minor variations in the exact ROC-AUC and similarity values (e.g., within a few percentage points). However, the overall trends reported in the article should be reproducible.

A.2 Artifact check-list (meta-information)

- **Algorithm: N/A**
- **Program: Python scripts for training, caption generation, similarity computation, and attack evaluation**
- **Compilation: N/A**
- **Transformations: N/A**
- **Binary: N/A**
- **Model: BLIP, PaliGemma 2, ViT-GPT2 (captioning VLMs); MPNet (text encoder for similarity); threat model is a threshold-based reference inference procedure**
- **Data set: COCO, NoCaps, CC3M (Conceptual Captions 3M)**
- **Run-time environment: Python 3.10 (tested); PyTorch, Hugging Face Transformers; `sentence-transformers`**
- **Hardware: CPU-only feasible for small subsets; one or more GPUs (e.g., NVIDIA with $\geq$16GB RAM) recommended**
- **Run-time state: N/A**
- **Execution: Command-line Python scripts (no GUI)**
- **Security, privacy, and ethical concerns: N/A**
- **Metrics: MPNet similarity (semantic), ROUGE-2 similarity (lexical), ROC-AUC for membership inference**
- **Output: CSV files with similarity statistics, attack scores, and plots/tables summarizing the impact of τ on similarity and ROC-AUC**
- **Experiments: Reference-based membership inference on three VLMs (BLIP, PaliGemma 2, ViT-GPT2) on three datasets (COCO, CC3M, NoCaps) and three threat models/regularization levels (BASELINE: τ=0, NEURO: τ=2, NEURO++: τ=3)**
- **How much disk space required (approximately)?: On the order of tens of gigabytes if all datasets (COCO, CC3M, NoCaps) and checkpoints are used; the code alone is small ($<$ 1GB)**
- **How much time is needed to prepare workflow (approximately)?: A few hours to set up the environment and download datasets, depending on network bandwidth**
- **How much time is needed to complete experiments (approximately)?: Running fine-tuning and attacks for all models and datasets can take multiple GPU-days; reproducing a single dataset/model/τ configuration can last for several GPU hours**
- **Publicly available (explicitly provide evolving version reference)?:**
  Github: `https://github.com/Trust-AI-ua/Neuro-VLM-resilience`
- **Code licenses (if publicly available)?: N/A**
- **Data licenses (if publicly available)?: Public datasets (COCO, NoCaps, CC3M) are subject to their respective licenses**
- **Workflow frameworks used?: Plain Python, shell scripts**
- **Archived (explicitly provide DOI or stable reference)?: N/A**

A.3 Description

A.3.1 How to access

This artifact codebase is shared via Github at `https://github.com/Trust-AI-ua/Neuro-VLM-resilience`. The codebase can be downloaded from there. The repository contains: i) Training scripts for BLIP, PaliGemma 2, and ViT-GPT2 with τ-regularization. ii) Scripts for dataset splitting into member/non-member and train/-validation subsets. iii) Caption generation scripts for member and non-member image sets. iv) Similarity computation scripts for MPNet and ROUGE-2. v) Attack scripts. vi) Utility scripts for aggregating results into CSVs and generating plots/tables. The raw datasets (COCO, NoCaps, CC3M) are included in the repository. They can be downloaded from their official sources, as described in the README. Pretrained weights for BLIP, PaliGemma 2, ViT-GPT2, and MPNet could be obtained via Hugging Face.

A.3.2 Hardware dependencies

No specific hardware is required to run this code. However, you may consider at least a modern NVIDIA GPU (e.g., 16GB RAM or more) and sufficient disk space for datasets and checkpoints.

A.3.3 Software dependencies

The artifact runs in a Python environment. At a high level, the setup includes: i) Python 3.10. ii) PyTorch and `torchvision` (with an appropriate CUDA build if using GPU).

iii) Hugging Face `transformers`, `accelerate`, and `datasets`. iv) `sentence-transformers` for MPNet embeddings. v) Common scientific Python packages: `pandas`, `numpy`, `matplotlib`, `scikit-learn`, `tqdm`, `Pillow`. The README includes a command to create a conda environment and a list of packages to install with `pip`.

A.3.4 Data sets

We use three publicly available image-caption datasets: i) **COCO**: MS-COCO 2017 training/validation images and caption annotations. ii) **NoCaps**: images and captions from the NoCaps benchmark. iii) **CC3M**: Conceptual Captions 3M (CC3M) dataset. For each dataset, we create member and non-member subsets, along with train/validation splits for the VLM fine-tuning stage. We provide helper scripts in `experiments/data/` (e.g., `nocaps_prepare_splits.py`, `cc3m_make_splits.py`, `cc3m_prepare_tsv.py`) to generate the necessary TSV and path list files that are used by the training and evaluation scripts. A high-level description of the datasets and splitting protocol is given in the experimental setup section of the main manuscript. We do not redistribute the raw datasets. However, we provide instructions and examples so they can be downloaded from their official websites.

A.3.5 Models

We consider three captioning VLMs as target models: BLIP, PaliGemma 2, and ViT-GPT2. Each model is fine-tuned with topographic regularization parameter $\tau \in \{0,2,3\}$, where: τ=0 corresponds to the BASELINE model, τ=2 to the NEURO variant, and τ=3 to the NEURO++ variant. We use MPNet (`sentence-transformers/all-mpnet-base-v2`) and ROUGE-2 (bigram F1) for the similarity computation. The membership inference attack itself is a reference inference attack adapted from Hu et al.'s `reference_non_member_inference.py` available at https://github.com/YukeHu/vlm_mia.

A.3.6 Security, privacy, and ethical concerns

Our artifact evaluates privacy attacks in a controlled, research setting on publicly available datasets that do not contain personally identifiable information. The code does not introduce additional security or ethical concerns beyond those inherent to membership inference research. Users should not apply these attacks to sensitive or non-consented data.

A.4 Installation

The artifact depends on a Python environment with the libraries described in A.3.3. We recommend using Anaconda for environment management. The installation workflow is:

1. Clone the repository:

```
git clone https://github.com/Trust-AI-ua
  /Neuro-VLM-resilience.git
cd Neuro-VLM-resilience
```

2. Create and activate a conda environment (example in README):

```
conda create -n neuro-vlm-resilience python=3.10
conda activate neuro-vlm-resilience
```

3. Install PyTorch (CPU or GPU build) and the required Python packages using `pip`, as listed in the README.
4. Download the datasets (COCO, NoCaps, CC3M) from their official sources and place them under `data/` following the directory structure described in the README.

The repository README provides concrete commands and environment setup examples.

A.5 Experiment workflow

The main experiment workflow consists of four stages:

1. **Dataset preparation.** For each dataset, we prepare:
   - member and non-member image sets, and
   - train/validation splits for fine-tuning.
2. **VLM fine-tuning with τ-regularization.** For each model (BLIP, PaliGemma 2, ViT-GPT2), dataset, and $\tau \in \{0,2,3\}$, we run a training script such as:
   - `experiments/train/train_blip_tau.py`
   - `experiments/train/train_paligemma_sft.py`
   - `experiments/train/train_vitgpt2_tau.py`

   These scripts consume `train.tsv` and `val.tsv` and write model checkpoints under `experiments/runs/<dataset>/<model>/tauX/checkpoints/`.
3. **Caption generation and similarity computation.** Using the trained checkpoints, we generate captions for member and non-member images:
   - BLIP: `experiments/infer/caption_blip_from_dir.py`
   - ViT-GPT2: `experiments/infer/caption_vitgpt2.py`
   - PaliGemma 2: `experiments/infer/gen_caps_paligemma.py`

   The resulting JSONs (e.g., `blip_member_tau3.json`) are passed to similarity scripts:
   - COCO-specific: `compute_similarity_blip_coco.py`, `compute_similarity_vitgpt2_coco.py`, `compute_similarity_paligemma_coco.py`
   - NoCaps/CC3M: `compute_similarity_paligemma_nocaps.py`, `compute_similarity_from_all_tsv.py`

   These scripts: i) compute MPNet and ROUGE-2 similarity between generated captions and reference captions, ii) write per-item similarity JSONs, and iii) append per-model/dataset/τ/set/metric mean scores to `experiments/results/similarity_summary.csv`.
4. **Membership inference attack and aggregation.** Finally, we run the reference-based MIA using: `experiments/attack/run_similarity_attack.py`, which wraps the `reference_non_member_inference.py` script. For each combination of dataset, model, τ, similarity metric, and granularity, it: i) reads member and non-member

similarity JSONs, ii) computes ROC-AUC, and iii) appends rows to `experiments/results/attack_accuracy.csv`. Post-processing scripts in `experiments/results/` then aggregate these CSVs to produce the tables and figures reported in the paper.

The README includes a concrete, end-to-end example for COCO · BLIP · τ=3 (NEURO++), which can serve as a template for reproducing other configurations.

A.6 Evaluation and expected results

The artifact supports the main empirical claims of the paper:

- **Effect of neuro-inspired topographic regularization on membership privacy.** We demonstrate that applying neuro-inspired topographic regularization ($\tau > 0$) during fine-tuning generally reduces membership inference attack performance, especially for BLIP on COCO. For example, the NEURO++ variant achieves a $\approx$ 24-point reduction in mean ROC-AUC for BLIP on COCO compared to the BASELINE (τ=0), while maintaining similar MPNet and ROUGE-2 similarity between generated and reference captions.
- **Cross-model and cross-dataset robustness.** We evaluate three VLMs (BLIP, PaliGemma 2, ViT-GPT2) on three datasets (COCO, CC3M, NoCaps) under three threat models/regularization levels (BASELINE, NEURO, NEURO++). The artifact reproduces the pattern that neuro-inspired variants often narrow the similarity gap between members and non-members and mostly reduce ROC-AUC relative to the baseline, with dataset-dependent variation.
- **Semantic vs. lexical similarity and membership gaps.** The artifact computes both MPNet (semantic) and ROUGE-2 (lexical) similarity. The resulting similarity statistics allow users to reproduce the observation that neuro-regularization can reduce semantic and lexical similarity or the member-non-member gap in structured datasets like COCO and CC3M, while effects on NoCaps are more modest.
- **End-to-end reproducibility of the pipeline.** Given the provided scripts and commands, users can re-run the pipeline for specific dataset/model/τ combinations and compare the similarity means and ROC-AUC values against the tables and figures in the paper. We expect small numerical differences (a few percentage points in ROC-AUC), but the qualitative trends (e.g., NEURO/NEURO++ often improving privacy relative to BASELINE without drastic utility loss) should hold.

An example workflow for a single configuration (e.g., COCO, BLIP, τ=3) is:

1. Prepare COCO splits (train/val, member/non-member) using the provided data scripts.
2. Fine-tune BLIP with the desired τ value using `train_blip_tau.py`.
3. Generate member and non-member captions with `caption_blip_from_dir.py`.
4. Compute MPNet and ROUGE-2 similarity using `compute_similarity_blip_coco.py`, which appends to `similarity_summary.csv`.
5. Run the reference-based attack via `run_similarity_attack.py`, which appends to `attack_accuracy.csv`.
6. Use the plotting/table scripts to aggregate results and compare to the corresponding table and figure entries in the paper.

Because the artifact uses publicly available datasets and standard libraries, and because all key scripts, paths, and parameters are exposed in the repository, the results in the paper can be independently verified up to minor variations due to training and hardware differences.

A.7 Experiment customization

Our artifact allows users to modify the experiments at several points:

- **Changing threat models (τ-values).** Users can fine-tune any supported VLM (BLIP, PaliGemma 2, ViT-GPT2) with arbitrary τ values by adjusting the command-line argument `-tau` in the training scripts.
- **Using different datasets or custom splits.** The data preparation scripts accept new image-caption datasets and can generate custom member/non-member splits, allowing experimenting with different distributions or dataset sizes.
- **Substituting similarity metrics.** Although we use MPNet and ROUGE-2, users may plug in different embedding models by modifying the similarity scripts.
- **Adjusting attack granularity.** The attack script exposes the choice of quantile thresholds via the `-granularity` parameter. This parameter allows users to probe finer separation between member and non-member similarities.
- **Extending evaluation outputs.** All intermediate similarity files, attack results, and summary CSVs can be exported or aggregated into additional plots or tables. The plotting utilities are written to be easily customized.

Each stage of the fine-tuning, caption generation, similarity computation, and attack evaluation, can be independently modified to make the artifact suitable for extended experimentation beyond the configurations we reported.

A.8 Future Direction

This artifact uses reference-based membership inference for captioning VLMs under different neuro-inspired topographic regularization levels. Several promising directions extend from the framework we provide:

- **Adaptive or layer-specific τ values.** Instead of a single global coefficient, future work may consider dynamically tuned or layer-selective τ schedules to better balance privacy and utility across different architectures.
- **Stronger or alternative threat models.** The current artifact implements black-box reference-based attacks. Extending the pipeline to support white-box MIAs, gradient-based attacks, or multi-stage adversaries would broaden the scope of security evaluation.
- **Tasks beyond image captioning.** The same methodology can be applied to other multimodal tasks such as visual question answering, where membership leakage may manifest differently.

A.9 Counter Measure of Proposed Attack

The τ-regularization framework serves as the countermeasure evaluated in this work. It is applied during VLM fine-tuning and is designed to reduce the similarity gap between captions generated for member and non-member images, which is the signal exploited by the reference-based MIA.

- **Mechanism.** The training objective is augmented with a topographic regularization term: $J_\tau = J_{\text{cap}} + \tau \cdot R_{\text{topo}}$, where J_{cap} is the standard captioning loss and R_{topo} penalizes variance in the cosine similarity between representations of neighboring training pairs. A higher τ encourages more uniform, less membership-specific internal representations, making it harder for an adversary to exploit similarity scores as a membership signal.
- **Effect on attack performance.** Increasing τ generally reduces the ROC-AUC of the reference-based MIA. For BLIP on COCO, mean ROC-AUC drops from 94.00% (BASELINE, τ=0) to 70.88% (NEURO, τ=2) and 63.46% (NEURO++, τ=3), a reduction of $\approx$24 percentage points. For PaliGemma 2 on COCO, mean ROC-AUC decreases from 70.86% (τ=0) to 66.76% (τ=3). For ViT-GPT2 on COCO, the NEURO and NEURO++ variants show substantially lower ROC-AUC than the BASELINE (55.51%), with NEURO reaching 29.38%.
- **Utility preservation.** The regularized models maintain comparable MPNet and ROUGE-2 similarity to reference captions, indicating that the reduction in attack success does not come at the cost of significant caption quality degradation.
- **Scope and limitations.** The countermeasure is effective primarily in structured, in-domain settings (e.g., COCO). Its effect is more modest on out-of-domain datasets such as NoCaps. Additionally, the τ implementation varies by model architecture: BLIP and ViT-GPT2 apply a variance penalty to decoder hidden states, whereas PaliGemma 2 uses label smoothing combined with logit L2 regularization because its decoder internals are not directly accessible in the same way. These architectural differences affect how strongly τ translates into privacy gains.

A.10 Image Caption Pairs

We demonstrate qualitative examples of member and non-member image-caption pairs generated by BLIP on COCO under each τ level ($\tau \in \{0, 2, 3\}$). For each τ, we illustrate the four classification outcomes: true positives (TP), false negatives (FN), true negatives (TN), and false positives (FP). The examples show how τ-regularization affects the generated captions and, consequently, the attack's ability to distinguish members from non-members.

Based on the LaTeX template for Artifact Evaluation V20220119.