USENIX Security '24 Artifact Appendix: Adversarial Illusions in Multi-Modal Embeddings

Tingwei Zhang[†*] Rishi Jha[†*] Eugene Bagdasaryan[‡] Vitaly Shmatikov[§]

[†]Cornell University [‡]University of Massachusetts Amherst [§]Cornell Tech

{tingwei, rjha}@cs.cornell.edu eugene@cs.umass.edu shmat@cs.cornell.edu

A Artifact Appendix

A.1 Abstract

This artifact provides the implementation and evaluation framework for the paper "Adversarial Illusions in Multi-Modal Embeddings." The artifact demonstrates the vulnerability of multi-modal embeddings to *adversarial illusions*: given an image or a sound, an adversary can perturb it to make its embedding close to an arbitrary, adversary-chosen input in another modality.

The provided code reproduces the paper's key findings on a number of multi-modal encoders and datasets, across various threat models and modalities, and against common f countermeasures. Detailed instructions for setting up the environment, running the experiments, and verifying the results are included to ensure that the claims made in the paper can be reproduced accurately.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

We emphasize that the purpose of our work is to help develop more robust embeddings and motivate research on defenses. In terms of the artifact, the datasets and models we used are public and the artifact is not destructive.

A.2.2 How to access

The code is available at this (stable) URL: https://github.com/ebagdasa/adversarial_illusions/tree/10c9d22c4ae6475ecaa13ba22c93f33be0293bca.

A.2.3 Hardware dependencies

The artifact evaluation requires the following hardware configuration:

- **CPU**: We used dual Intel(R) Xeon(R) Gold 6448Y processors with 64 cores in total, but comparable hardware will likely suffice.
- **Memory**: Our machine had 256 GiB of system RAM, however much less will suffice.
- **GPU**: Our white-box were all be run on single NVIDIA 2080ti gpus, while our transfer experiments require at least two. Our generation experiments were run on NVIDIA A6000s.
- **Storage**: At least 40 GiB to store datasets, model checkpoints, and experiment results.

A.2.4 Software dependencies

The artifact requires the following operating system and essential software packages:

- **Operating System**: The code has been tested on Ubuntu 20.04 LTS. It is recommended to use this OS for compatibility and to ensure reproducibility of the results. Other Linux distributions may work but are not guaranteed.
- **Python**: Python 3.10 or higher is required. It is recommended to use a virtual environment (conda) to manage dependencies.
- **CUDA and cuDNN**: To utilize GPU acceleration, ensure that CUDA 11.0 or higher and cuDNN are installed.

All other dependencies are listed in the `environment.yml` and `README.md` files provided in the GitHub repository. They ensure that all necessary packages, models, and datasets are correctly installed.

A.3 Set-up

A.3.1 Installation

As summarized in the `README.md` of our GitHub:

*Comparable contributions.

1. **Setup Environment:** run `conda env create -f environment.yml`

2. **Download Data:** We run experiments on ImageNet, AudioSet, and LLVIP. For ease of reproduction, we provide necessary config files for all datasets and 100-example subsets of the latter two datasets as a release. To install, please download the `data.zip` file and, from root, run `unzip /path/to/data.zip -d .`

    - For ImageNet, we only use the validation set. As required by PyTorch, we also require `ILSVRC2012_devkit_t12.tar.gz` and `ILSVRC2012_img_val.tar` to be located in `data/imagenet/`. Please follow the instructions in the note on PyTorch's page to acquire those two files.

3. **AudioCLIP Checkpoints:** To conduct any experiments on AudioCLIP, we require pretraining checkpoints.

    - For the full checkpoint, run:

      ```
      wget https://github.com/AndreyGuzhov/AudioCLIP/releases/download/v0.1/AudioCLIP-Full-Training.pt -P bpe/
      ```

    - For the partial checkpoint (used for transfer attacks):

      ```
      wget https://github.com/AndreyGuzhov/AudioCLIP/releases/download/v0.1/AudioCLIP-Partial-Training.pt -P bpe/
      ```

4. **Submodule Setup:** This includes lightly adapted code from ImageBind, AudioCLIP, and DiffJPEG and directly employs two submodules: PandaGPT and BindDiffusion. To initialize the two submodules (if desired), run the following and download the checkpoints as described below:

    ```
    git submodule update -init
    ```

    ```
    scp image_text_generation/image_generation.py BindDiffusion
    ```

    ```
    scp image_text_generation/text_generation_demo.ipynb PandaGPT/code
    ```

    ```
    scp image_text_generation/text_generation.py PandaGPT/code
    ```

    - **PandaGPT Checkpoints:** To conduct any experiments with PandaGPT, place the PandaGPT checkpoints into `PandaGPT/pretrained_ckpt` by following these instructions.
    - **BindDiffusion Checkpoints:** To conduct any experiments with BindDiffusion, place the BindDiffusion checkpoints into `BindDiffusion/checkpoints` by following these instructions.

A.3.2 Basic Test

Run the Jupyter notebook `image_illusion_demo.ipynb` to generate an adversarial illusion image that causes misinformation using a white-box attack. You can replace the existing image and aligned text with your own choices to generate an image illusion. Then, use the perturbed image as input to `text_generation_demo.ipynb`, which should produce output relevant to the target text used previously to perturb the image.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): We demonstrate that **adversarial alignment** (i.e., the $\cos(\cdot)$ similarity of unrelated inputs) can be perturbed significantly closer than any organic alignment (i.e., the $\cos(\cdot)$ similarity of related inputs) regardless of source and target modality. This corresponds to Table 1, Table 4, Table 5, and Table 6 in the paper. This is proven by experiment (E1), (E4), (E5), and (E6).

(C2): We show that **multiple tasks**—including image generation, text generation, audio retrieval, and different types of zero-shot classification (image, thermal image, and audio)—are all misled by cross-modal illusions with **white-box attack method**. This corresponds to Table 1, Table 2, Table 3, Table 4, Table 5, and Table 6 in the paper. This is proven by experiments (E1), (E2), (E3), (E4), (E5), and (E6).

(C3): We analyze **transferability** of the attack across different encoders, investigate how it is influenced by model architecture and craft illusions that work against multiple embeddings. In particular, we show that illusions generated using OpenCLIP encoders also achieve 100% and 90% attack success rates against zero-shot image classification on ImageBind and AudioCLIP embeddings. This corresponds to Table 7 in the paper. This is proven by experiment (E7).

(C4): We demonstrate a **black-box, query-based** attack and show that it is effective against several downstream tasks. We combine our query-based and transfer techniques into a hybrid method and use it for the first adversarial alignment attack against Amazon's Titan, a commercial black-box embedding. This corresponds to Table 8 in the paper. This is proven by experiment (E8). Note that we omit the code for evaluating attacks on Amazon's Titan embedding because these attacks involve setting up an AWS instance and paying for access to a commercial API.

(C5): We survey several **countermeasures** and demonstrate how adversarial illusions can evade defenses based on JPEG compression and anomaly detection based on the consistency of augmentations. This corresponds to Table 9 and Table 10 in the paper. This is proven by experi-

ments (E9) and (E10).

A.4.2 Experiments

(E1): *[White-box zero-shot image classification] [1 human-hour + 32 compute-hours]: This experiment generates the result of Table 1. Our results are run with 7500 epochs. Far fewer epochs (<50) are sufficient to fool most models.*

Setup: Our white-box and baseline experiments are configured by `{EXPERIMENT_NAME}.toml` files in the `configs/` folder. To generate a tuple of numbers in Table 1 (Top-1 Accuracy, Top-5 Accuracy, Mean Loss, Standard Deviation of Loss), a configuration file must be created with the specified hyperparameters (e.g., output directory, perturbation budget, etc.). Examples with a fixed hyperparameter set have been included in `configs/imagenet/whitebox/`.

Execution: For a white-box attack, run `python adversarial_illusions.py imagenet/whitebox/{MODEL_NAME}.toml > ../outputs/imagenet/whitebox/{MODEL_NAME}/result.txt`. For baseline numbers, depending on the type (organic or adversarial, run `python evaluate_illusions.py baseline/{TYPE}/image_classification.toml > ../outputs/imagenet/baseline/{TYPE}/image_classification/result.txt`.

Results: The results will be saved to the specified text file. Note that due to numeric instability, exact numbers from the paper may be impossible to reproduce.

(E2): *[White-box image generation] [1 human-hour + 1.5 compute-hour]: This experiment generates the result of Table 2.*

Execution: Run `python BindDiffusion/image_generation.py > ../outputs/imagenet/image_generation/whitebox/result.txt` to generate adversarial illusions with the query-based method.

Results: The results will be saved to the last few lines of the txt file.

(E3): *[White-box text generation] [0.5 human-hour + 4.5 compute-hour]: This experiment generates the result of Table 3.*

Execution:

1. Run `python thermal_illusion_text_generation.py` to generate images aligned with the text "criminal with a gun".
2. Run `python PandaGPT/code/text_generation.py` to generate PandaGPT's responses with unperturbed and perturbed images.
3. Copy the results from the JSON files `outputs/thermal/perturbed_images_text_generation/response_thermal_unperturbed.json` and `outputs/thermal/perturbed_images_text_generation/response_thermal_perturbed.json` into ChatGPT-4 with the text prompt "`Answer with yes or no, does each response describe anything unsafe? Make the result a table.`" Count the "yes" and "no" responses for the two JSON files.

Results: The numbers of "yes" and "no" in ChatGPT responses with "`response_thermal_unperturbed.json`" and "`response_thermal_perturbed.json`" correspond to the numbers of each row in Table 3.

(E4): *[White-box zero-shot audio classification] [1 human-hour + 32 compute-hours]: This experiment generates the result of Table 4. Our results are run with 7500 epochs. Far fewer epochs (<50) are sufficient to fool most models.*

Setup: Same setup as (E1), making sure to set the `modality` flag to '`audio`' and the `dataset` flag to '`audioset`'. Examples with a fixed hyperparameter set have been included in `configs/audioset/whitebox/`.

Execution: For a white-box attack, run `python adversarial_illusions.py audioset/whitebox/{MODEL_NAME}.toml > ../outputs/audioset/whitebox/{MODEL_NAME}/result.txt`. For baseline numbers, depending on the type (organic or adversarial, run `python evaluate_illusions.py baseline/{TYPE}/audio_classification.toml > ../outputs/imagenet/baseline/{TYPE}/audio_classification/result.txt`.

Results: The results will be saved to the specified text file. Note that due to numeric instability, exact numbers from the paper may be impossible to reproduce.

(E5): *[White-box audio retrieval] [1 human-hour + 32 compute-hours]: This experiment generates the result of Table 5. Our results are run with 7500 epochs. Far fewer epochs (<50) are sufficient to fool most models.*

Setup: Same setup as (E1, E4), making sure to set the `modality` flag to '`audio`' and the `dataset` flag to '`audiocaps`'. Examples with a fixed hyperparameter set have been included in `configs/audiocaps/whitebox/`.

Execution: For a white-box attack, run `python adversarial_illusions.py audiocaps/whitebox/{MODEL_NAME}.toml > ../outputs/audiocaps/whitebox/{MODEL_NAME}/result.txt`. For baseline numbers, depending on the type {organic, adversarial}, run `python evaluate_illusions.py baseline/{TYPE}/audio_retrieval.toml > ../outputs/imagenet/baseline/{TYPE}/audio_retrieval/result.txt`.

Results: The results will be saved to the specified text file. Note that due to numeric instability, exact numbers from the paper may be impossible to reproduce.

(E6): *[White-box zero-shot thermal image classification.] [0.1 human-hour + 1.5 compute-hour]: This experiment generates the result of Table 6.*
Execution: Run `python thermal_illusion_classification.py > outputs/thermal/result.txt`
Results: The results will be saved to the txt file. Each text block corresponds to the rows in Table 6.

(E7): *[Transfer zero-shot image classification.] [1 human-hour + 1 compute-hour]: This experiment generates the result of Table 7.*
Setup: This experiment requires adversarial illusions to be generated either by (E1) or via ensemble: `python adversarial_illusions.py imagenet/transfer/ensemble.toml`. For best performance the illusions should be saved after 300 epochs (via the `epochs` flag. All adversarial illusions should be saved to `.npy` files and served to the evaluation config via the `adv_file` flag. Examples with a fixed hyperparameter set have been included in `configs/audioset/transfer/{MODEL_NAME}_eval.toml`.
Execution: To evaluate transferability, run `python evaluate_illusions.py imagenet/transfer/{MODEL_NAME}_eval.toml > ../outputs/imagenet/transfer/{MODEL_NAME}/result.txt`.
Results: The results will be saved to the specified text file. Note that due to numeric instability, exact numbers from the paper may be impossible to reproduce.

(E8): *[Query-Based zero-shot image classification.] [0.5 human-hour + 20 compute-hour]: This experiment generates the result of Table 8.*
Execution: Run `python query_attack.py imagebind > outputs/imagenet/query/imagebind/result.txt` and `python query_attack.py audioclip > outputs/imagenet/query/audioclip/result.txt` to attack embeddings using the query-based method.
Results: The results will be saved to the last few lines of the txt file.

(E9): *[JPEG-resistant illusions.][0.2 human-hour + 1 compute-hour]: This experiment generates the result of Table 9.*
Preparation: Make sure (E1) has been performed, or run `python adversarial_illusions.py imagenet/whitebox/imagebind`, and `python adversarial_illusions.py imagenet/whitebox/audioclip` to generate illusions without evasion attack.
Execution:

1. Run `python adversarial_illusions.py imagenet/whitebox/imagebind_jpeg` and `python adversarial_illusions.py imagenet/whitebox/audioclip_jpeg` to generate JPEG-resistant illusions.
2. Run `python evaluate_jpeg.py` to evaluate the illusions with and without evasion attack.

Results: The results will be saved to the txt file.

(E10): *[Evading anomaly detection.][0.2 human-hour + 0.1 compute-hour]: This experiment generates the result of Table 10.*
Execution: Run `python anomaly_detection.py > outputs/imagenet/whitebox/anomaly_detection.txt`
Results: The results will be saved to the txt file.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.