# PixelMod: Improving Soft Moderation of Visual Misleading Information on Twitter

Pujan Paudel and Chen Ling, *Boston University;* Jeremy Blackburn, *Binghamton University;* Gianluca Stringhini, *Boston University*

https://www.usenix.org/conference/usenixsecurity24/presentation/paudel-pixelmod

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

# USENIX Security ’24 Artifact Appendix: PIXELMOD: Improving Soft Moderation of Visual Misleading Information on Twitter

Pujan Paudel, Chen Ling, Jeremy Blackburn, and Gianluca Stringhini

## A Artifact Appendix

## A.1 Abstract

This document presents the artifact of our Usenix Security’24 paper: “PIXELMOD: Improving Soft Moderation of Visual Misleading Information on Twitter.” The core contribution of this artifact is a reverse image search system that leverages perceptual hashes, vector databases, and Optical Character Recognition (OCR) to efficiently identify images candidates of moderation, given a query image. This artifact presents the implementation of PIXELMOD built on the vector database setup of Milvus, populated with an image index. The artifact also packages a companion Web application to interactively query and preview the results retrieved from the vector search database.

## A.2 Description & Requirements

The artifacts associated with this submission are packaged as a Docker container uploaded at the stable Zenodo URL[1]. We package the end-to-end image analysis pipeline as presented in Figure 2 of the paper in this Docker container. Additionally, we also package a Web application via Streamlit to interface the reverse image search system, with pre-built examples from the dataset for demonstrative purposes. We also present an “Upload” widget for images that evaluators want to upload by themselves and interact with the reverse image search system.

### A.2.1 Security, privacy, and ethical concerns

There are no risk for evaluators when executing this artifact on their machines or in the cloud. All datasets used in this work were either publicly released by other researchers or were collected using publicly available APIs and following those API’s terms of service.

### A.2.2 How to access

The raw dataset of pre-computed PDQHash embeddings and the metadata associated with the images are available in the aforementioned stable Zenodo URL. We also provide the docker containers with embeddings already indexed in the Milvus system, used for the interactive web application and evaluation scripts alongside the stable Zenodo URL. Additionally, the source code used for building the vector index, web interface and evaluation scripts are available in the companion Github repository[2].

### A.2.3 Hardware dependencies

The only hardware requirement needed for this artifact is that the evaluation system needs to have an *x86* architecture. This requirement stems from Milvus which requires a CPU that supports one of the following instruction sets: SSE4.2, AVX, AVX2, AVX51. Additionally, disk space and memory of 15GB each is needed for the docker container, to load the Milvus index for interactive queries and evaluation.

### A.2.4 Software dependencies

The major software dependency necessary for the artifact evaluation is ***Docker***. The list of Python packages and the corresponding versions for these packages is listed in *requirements.txt* of the Docker repository, which is installed while building the container. It is recommended that for succesful evaluation of the artifact, proper environmental dependencies as listed above are met as described in the section “Instructions to Download and Run PixelMod ” available in the companion Github repository.

### A.2.5 Benchmarks

The dataset used to build the Milvus index associated with the reverse image search systems is organized in the docker image *pixelmod*. The docker image *pixelmod* corresponds to the index built with **GT**$_{viz}$ dataset discussed in the paper, with 9.5K images indexed for evaluation of the system. The interactive web application uses *pixelmod* as the underlying index to retrieve visually similar matches as results.

## A.3 Set-up

The setup procedure for this artifact includes ensuring appropriate hardware requirements and software requirements are met.

[1] https://doi.org/10.5281/zenodo.12570381

[2] https://github.com/idramalab/pixelmod/tree/v1.0.0

### A.3.1 Installation

The docker images associated with the artifact can be downloaded from the stable Zenodo URL. After downloading the container, follow the instruction in the companion Github repository to succesfully build and link the docker container and docker volumes. This docker build script packages the installation of underlying Milvus system, as well as scripts for indexing the images in the corresponding Milvus index.

### A.3.2 Basic Test

A basic functionality test ensuring all the models and datasets are loaded properly can be verified by running the script *basic_test.py* on the artifact companion repository. This script loads the Milvus index *pixelmod_eval* in memory, and queries the index with 5 different example images from the dataset. Upon running the basic test script, the script first prints number of items in each of the collections (reflecting the size of Milvus indices), and visually similar "matches" to these 5 query images are returned. This basic test ensures that Milvus is installed correctly, the images are indexed to the Milvus collection as expected, and the query behavior of Milvus works as expected.

## A.4 Evaluation workflow

### A.4.1 Major Claims

**(C1):** *Given a known misleading image as query,* PIXELMOD *can be used to identify visually and contextually similar images to the query images for downstream soft-moderation purposes. This is proven by the experiment (E1) described in paper in [Section 2.1], and illustrated in [Figure 2].*

**(C2):** PIXELMOD *achieves a Precision score of 0.990, Recall score of 0.979, and F1 score of 0.980 when evaluated on GTviz. This is proven by the experiment (E2) described in the paper in [Section 5.1* PIXELMOD *vs. baselines].*

### A.4.2 Experiments

**(E1):** *[Using reverse image search system of Pixelmod interfaced via Streamlit* PIXELMOD *] [5 human-minutes + 5 compute-minutes + 15 GB disk]:* PIXELMOD *can be used as a reverse image search system to query visually misleading images, and retrieve other images from the index that are both contextually and visually similar.*

**How to:** *The step-by-step flow to reproduce the results of this experiment is packaged in the Python script experiment1_streamlit.py alongside the artifact companion repository.*

**Preparation:** *Make sure the "Basic Setup" tests run succesfully.*

**Execution:** *Once the basic setup tests are confirmed succesfully, run the Python script experiment1_streamlit.py and follow the instructions in the Streamlit interface.*

**Results:** *The results corresponding to Figure 3 are presented in the Web application. We also present the most popular images from our dataset, and the resulting matches that* PIXELMOD *identifies as visually similar for these images. Evaluators can also test the reverse image search system with other provided images from the dataset, or upload their own images via the image upload widget.*

**(E2):** *[Benchmarking* PIXELMOD *] [5 human-minutes + 5 compute-minutes + 1 GB disk]:* PIXELMOD *achieves a Precision score of 0.990, Recall score of 0.979, and F1 score of 0.980 when evaluated on* $\boldsymbol{GT}_{viz}$.

**How to:** *The step-by-step flow to reproduce the results of this experiment is packaged in the Python script experiment2_evaluating.py alongside the artifact companion repository.*

**Preparation:** *Make sure the "Basic Setup" tests run succesfully.*

**Execution:** *Once the basic setup tests are confirmed succesfully, run the Python script experiment2_evaluating.py and observe the result of* PIXELMOD*'s performance on the GTviz benchmark.*

**Results:** PIXELMOD*'s performance on GTviz are reflected through three different metrics: i) Precision, ii) Recall, and iii)F1 score.*

## 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/.