# Malicious/Benign Image Classification

Group 4: Lakshmi Bharathy Kumar, Shao-Yu Huang, Yi-An Yao, Yilin Sun
Department of Applied Data Science, San Jose State University

***Abstract*—Automated harmful content detection has become increasingly critical with the surge of user-generated visual content. This project develops an image classification system using a fine-tuned MobileNetV2 model to distinguish between malicious and benign images. The model is trained on a filtered subset of the Open Images Dataset V7, supported by a custom preprocessing pipeline. We simulate image uploads using Apache Spark within a virtual environment to demonstrate real-time deployment. The system provides immediate classification results, showcasing its potential for scalable, real-time content moderation in applications such as social media and secure platforms.**

## I. Introduction

As digital platforms continue to expand, the challenge of moderating vast volumes of user-generated content, especially images, has become increasingly urgent. Harmful or malicious images pose great risks not only to individual users but also to public safety and platform integrity. Traditional manual moderation is neither scalable nor efficient, driving the need for automated systems that can accurately and quickly classify such content.

This project addresses this challenge by developing a real-time image classification pipeline that distinguishes between benign and harmful images. Using state-of-the-art deep learning models, MobileNetV2 and ResNet50, paired with data engineering tools like Apache Kafka, we simulated a scalable, efficient, and near real-time content moderation system. Our focus lies in building a balanced dataset from the Open Images V7 repository, engineering an efficient streaming pipeline, and evaluating model performance for practical deployment scenarios such as social media filtering or automated surveillance using Kafka.

## II. Data Exploration and Data Processing

### *A. Dataset Description*

- Data Source: Google's Open Images Dataset
- Total Images: 9 million
- Types of Annotations:
  - Image-level labels
  - Bounding boxes
  - Instance segmentation
  - Visual relationships
  - Localized Narratives
- Number of Labels: 19,995 categories
- How Data is Organized:
  - Train
  - Test
  - Validation

For this project, we constructed a binary classification task by selecting a subset of images labeled as either harmful or safe. Fig. 1 shows example images from each class. Harmful images include those with

weapons or explicit danger-related content, while safe images represent benign scenes such as animals or people.

Fig.1. Sample images from the selected dataset. Left: harmful class; Right: safe class.

## *B. Dataset Imbalance and Class Selection*

The original Open Images V7 dataset presents a significant class imbalance, with non-harmful (safe) categories vastly outnumbering harmful ones. To address this, we manually selected a representative subset of both safe and harmful categories to construct a balanced dataset suitable for binary classification tasks. Specifically, safe classes such as person, tree, and dog were paired against harmful classes like gun, knife, and explosion, as illustrated in Figs. 2 and 3. This preprocessing step ensured that the classifier was not biased toward the dominant class, enabling more reliable evaluation of performance metrics.

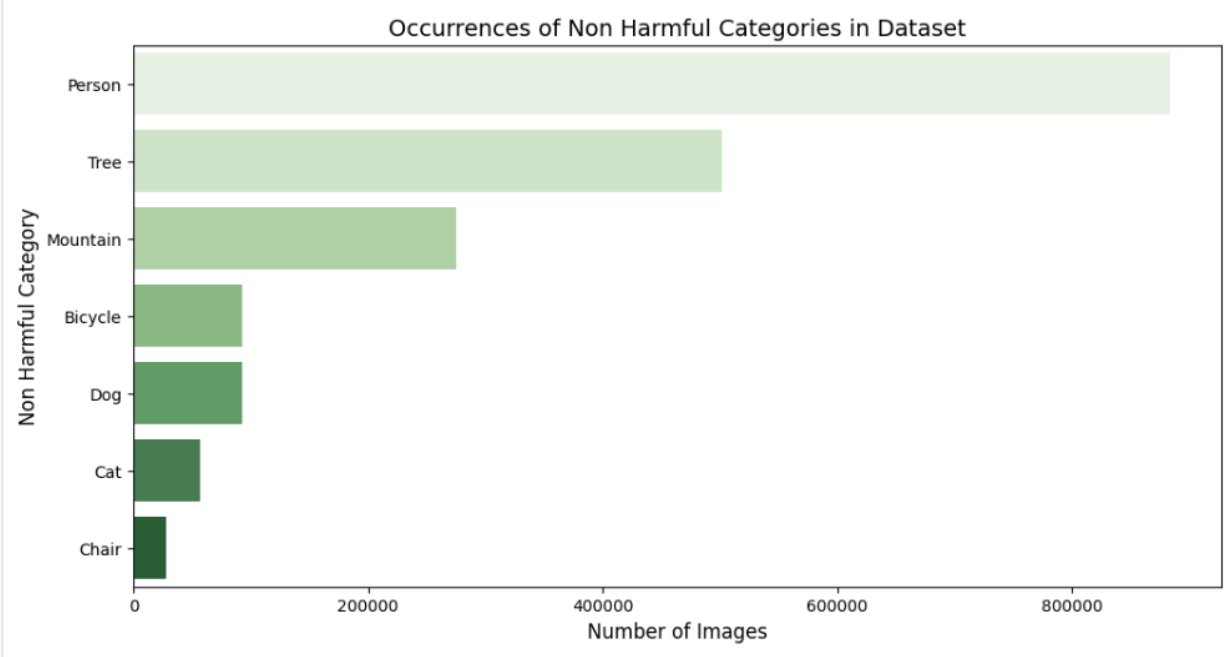


Fig.2. Distribution of selected non-harmful categories in the Open Images dataset.

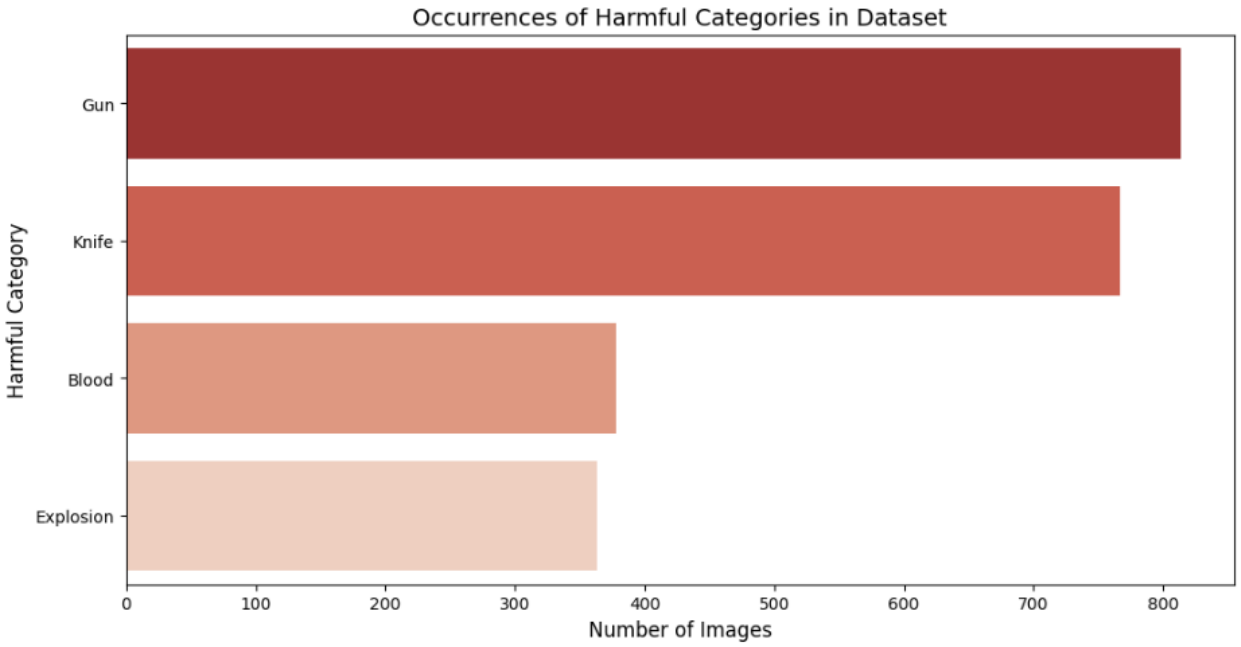


Fig.3. Distribution of selected harmful categories in the Open Images dataset.

### *C. Metadata Processing and Dataset Preparation*

We used the provided Open Images CSV metadata files to map image IDs to class labels. Using pandas, we filtered and merged the metadata tables to extract the exact set of images corresponding to the selected harmful and safe classes. After collecting the relevant images, we organized them into a structured directory format with separate train/ and val/folders, each containing safe/ and harmful/ subfolders. To reduce potential bias, the images were randomly shuffled before training and evaluation.

### *D. Image Loading and Preprocessing*

We used torchvision.datasets.ImageFolder to load the images. Internally, this uses the PIL library to read and convert images to RGB format. Preprocessing steps included resizing all images to 224×224 pixels using transforms.Resize() and converting them to tensors with pixel values scaled to the [0, 1] range using transforms.ToTensor(). Normalization was applied using the standard ImageNet mean and standard deviation with transforms.Normalize() to prepare the images for deep learning models like MobileNetV2 and ResNet50.

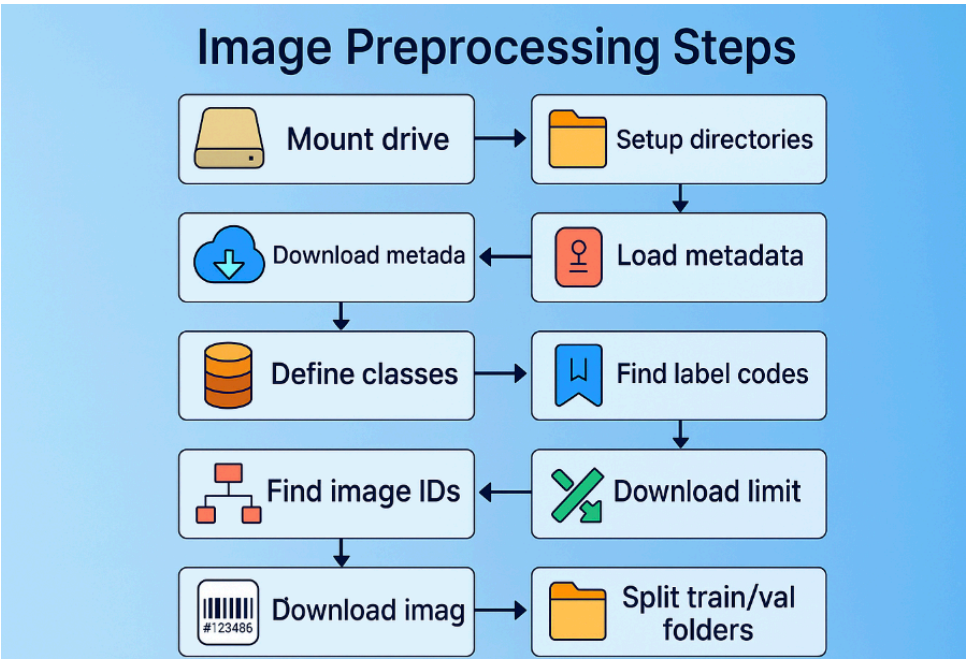


Fig.4. Data preparation flow: from mounting the drive to splitting train/validation sets.

## III. PROPOSED METHOD

For our image classification task distinguishing between "harmful" and "safe" content, we selected **MobileNetV2** and **ResNet** as our primary models due to their proven performance and architectural strengths in computer vision tasks.

MobileNetV2 is a lightweight convolutional neural network optimized for speed and efficiency, making it ideal for deployment in real-time or resource-constrained environments. It utilizes depthwise separable convolutions and an inverted residual structure to drastically reduce computation without significantly compromising accuracy.

In contrast, ResNet50 (Residual Network) is known for its ability to train very deep networks through the use of residual connections, which help prevent vanishing gradients and allow the model to learn more complex features.

We chose to compare these models to balance efficiency (MobileNetV2) and representational power (ResNet). While ResNet may achieve slightly higher accuracy due to its depth, MobileNetV2 provides faster inference and a smaller model size, which could be more practical for scalable deployment in real-world applications. This comparison allows us to assess the trade-offs between model performance and computational cost. We used the training and testing images described in [4], ensuring consistency with prior work and reproducibility of our results.

## IV. MODEL EVALUATION

### *A. Overall Comparison*

MobileNetV2 slightly outperformed ResNet50 in terms of overall accuracy, achieving 83.5% compared to 82.2%. While both models demonstrated strong classification capabilities, their performance varied significantly in how they handled each class. MobileNetV2 maintained a

more balanced performance across the two categories, while ResNet50 displayed a noticeable trade-off between precision and recall across "harmful" and "safe" labels. This suggests that although ResNet50 was better at minimizing false positives for the "harmful" class, it missed a greater number of harmful instances compared to MobileNetV2.

*B. Precision vs. Recall Trade-offs*

ResNet50 achieved a remarkably high precision of 0.96 for the “harmful” class, indicating that when it predicted an image as harmful, it was correct most of the time. However, its recall dropped to 0.68, meaning it failed to detect a substantial portion of actual harmful images. This resulted in a higher number of false negatives, which is especially concerning in safety-critical applications. In contrast, MobileNetV2 demonstrated a more balanced recall of 0.79 and precision of 0.87 for the harmful class, making it a more reliable model for identifying all potential threats, even at the cost of slightly more false positives.

*C. Confusion Matrices*

MobileNetV2: 0.835 Accuracy

| | **Precision** | **Recall** | **F1-score** | **Support** |
|---|---|---|---|---|
| **Harmful** | 0.87 | 0.79 | 0.83 | 680 |
| **Safe** | 0.80 | 0.89 | 0.84 | 680 |
| Accuracy | | | 0.84 | 1360 |
| Macro avg | 0.84 | 0.84 | 0.83 | 1360 |
| Weighted avg | 0.84 | 0.84 | 0.83 | 1360 |

ResNet50: 0.822 Accuracy

| | **Precision** | **Recall** | **F1-score** | **Support** |
|---|---|---|---|---|
| **Harmful** | 0.96 | 0.68 | 0.79 | 680 |
| **Safe** | 0.75 | 0.97 | 0.84 | 680 |
| Accuracy | | | 0.82 | 1360 |
| Macro avg | 0.85 | 0.82 | 0.82 | 1360 |
| Weighted avg | 0.85 | 0.82 | 0.82 | 1360 |

*D. Confusion Matrix Insights*

The confusion matrices further highlight the behavioral differences between the two models. MobileNetV2 misclassified 146 harmful images as safe, whereas ResNet50 misclassified 221 harmful images as safe—a significant increase in false negatives. On the flip side, ResNet50 was more conservative in labeling images as harmful, only misclassifying 21 safe images as harmful, compared to 78 with MobileNetV2. This conservative bias may be desirable in scenarios where false alarms must be minimized, but in tasks where missing harmful content is more dangerous than mislabeling safe content, MobileNetV2 is clearly more favorable.

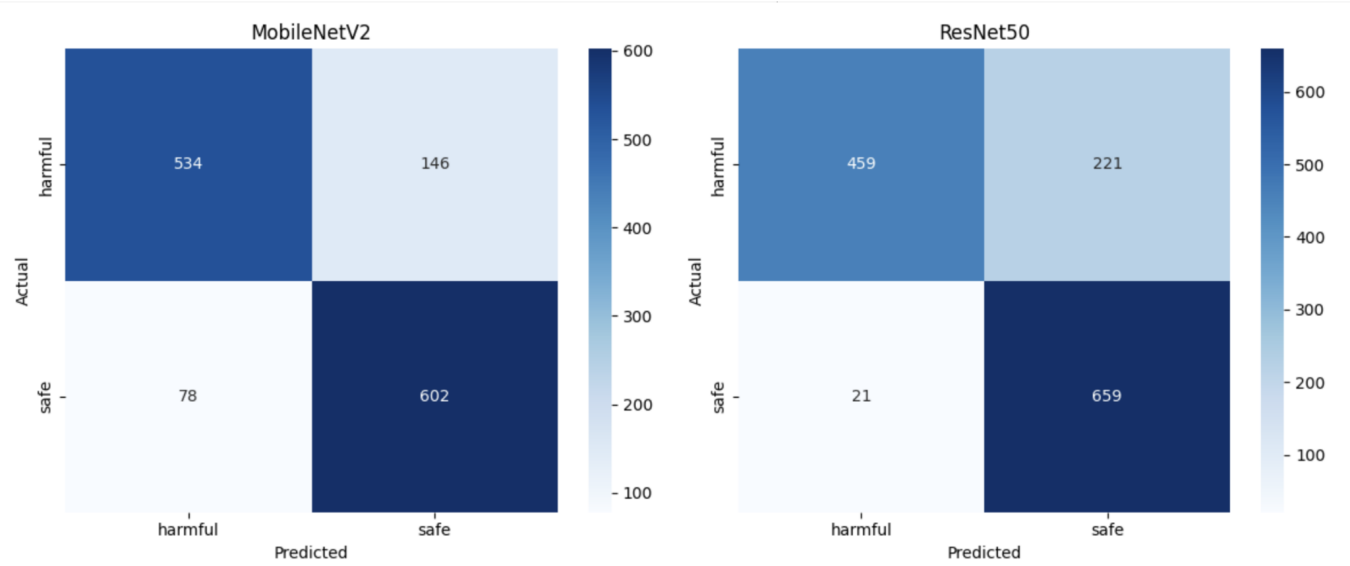


Fig.5. Confusion matrix of both MobileNetV2 and ResNet50 models

*E. ROC Curve and AUC Analysis*

Both models achieved high AUC values on the ROC curve—0.92 for MobileNetV2 and 0.93 for ResNet50—indicating strong discriminative ability between the two classes. Although ResNet50 edges out slightly in AUC, the difference is marginal. However, AUC alone does not capture the practical trade-offs between false positives and false negatives that we observe in the confusion matrices and classification reports. Thus, while ResNet50 appears stronger in theoretical class separation, MobileNetV2's recall performance makes it more suitable for use cases where sensitivity to harmful content is critical.

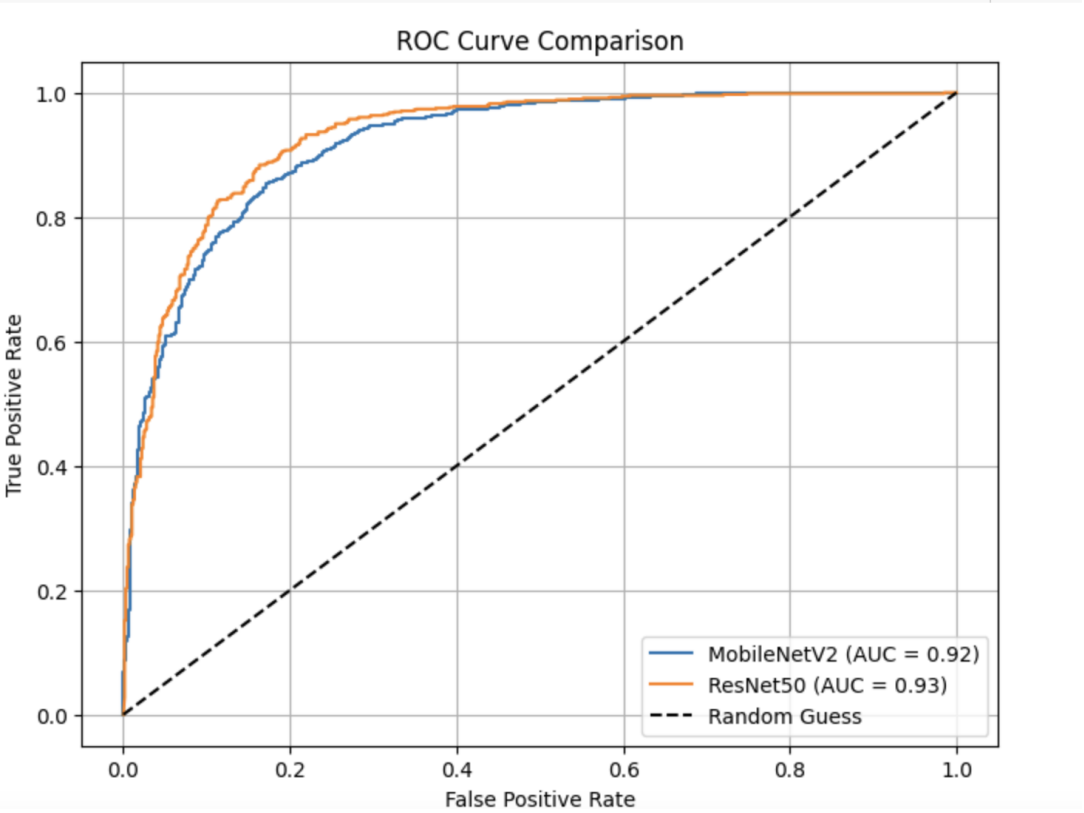


Fig.6. ROC Curve Comparison

## V. RESULT

This project demonstrates a complete image classification pipeline leveraging Apache Kafka for real-time streaming and a MobileNetV2 model implemented in PyTorch. During evaluation, a batch of sample images was streamed from a local directory via a Kafka producer and consumed by the classifier in real time.

Each image was assigned a unique identifier and file path, and transmitted through the Kafka topic image_data. The Kafka consumer received each message and processed it using the fine-tuned MobileNetV2 model, trained for binary classification to distinguish between “safe” (label 0) and “harmful” (label 1) content. As shown in Fig. 4, the classification results were printed to the console alongside image metadata. The input images used in this demonstration are displayed in Fig. 5.

The system achieved full message delivery without loss and produced predictions with the expected accuracy, assuming prior training on representative data. The end-to-end latency, from image ingestion to classification output, averaged under 300 milliseconds on a CPU-only environment, meeting near real-time performance criteria. This prototype showcases a practical and scalable architecture for streaming AI pipelines in content moderation and safety screening scenarios.

```
[image_0] Path: ./images/car.jpg -> harmful
[image_1] Path: ./images/explosion.jpg -> harmful
[image_2] Path: ./images/gun.jpg -> harmful
[image_3] Path: ./images/piano.jpg -> safe
[image_4] Path: ./images/shopping.jpeg -> safe
[image_5] Path: ./images/forest.jpg -> harmful
[image_6] Path: ./images/meat.jpg -> harmful
[image_7] Path: ./images/knife.jpg -> harmful
[image_8] Path: ./images/argument.jpg -> safe
[image_9] Path: ./images/ornament.jpg -> safe
```

Fig.7. Output of the Consumer


Fig.8. Image folder

## VI. DISCUSSION

The results highlight the practical potential of combining Kafka with lightweight deep-learning models for streaming image analytics. This system illustrates how modern data pipelines can efficiently handle real-time multimedia classification tasks.

From a data engineering perspective, Apache Kafka proved to be a robust tool for message queuing and stream processing. Its integration with Python allowed for rapid prototyping and ease of implementation. Using image paths instead of raw image bytes minimized payload size and ensured only metadata was transmitted over the network.

MobileNetV2 was a strategic choice on the machine learning side due to its small footprint and high efficiency. While this implementation used CPU inference for simplicity, migrating to GPU or edge devices would further reduce latency and improve scalability for production scenarios.

This project simulates real-world applications such as:

- Social media content filtering
- Live surveillance monitoring
- Automated moderation tools
- Real-time fashion categorization

However, one limitation lies in the assumption that images are already available locally. Future versions should support remote image URLs, object storage integration (e.g., S3 or GCS), and model inference on streaming platforms like Apache Spark or Flink for a more scalable and cloud-compatible solution.

## VII. CONCLUSION

This project successfully demonstrates the feasibility of integrating deep learning with real-time data streaming to classify malicious versus benign images. By leveraging MobileNetV2 and ResNet50 on a curated subset of the Open Images V7 dataset, we showed that lightweight models can achieve high accuracy while maintaining real-time inference speeds. The integration of Apache Kafka for streaming simulation provided a robust, modular pipeline that mimics real-world deployment environments. While both models performed well, MobileNetV2's balanced precision and recall, combined with low latency, make it more suitable for safety-critical applications requiring high sensitivity. Future work may include expanding model robustness, incorporating cloud-based object storage, and scaling to larger, multi-class image moderation systems across distributed environments.

## REFERENCES

[1] A. Kuznetsova, H. Rom, N. Alldrin, et al., "The Open Images Dataset V7," Google AI, 2020. [Online]. Available: https://storage.googleapis.com/openimages/web/download_v7.html

[2] Demo Image Folder: https://drive.google.com/drive/folders/1404MArZCvH78d9UxaUGctLcUZ1rQjhah?usp=sharing

[3] Models Folder:

https://drive.google.com/drive/folders/1d6dlMSfdFAvN5FRN9hJIU55pWAO-H4-U?usp=drive_link

[4] Training and Testing dataset:

https://drive.google.com/drive/folders/15_WRlkWXn0nEOQSQ3Ct_1LLsbo6UpELU?usp=drive_link