VIC Assignment 2 : X-Ray Security Screening Challenge

Mathis Pernin

February 16, 2026

1 Introduction

X-ray images are challenging compared to natural images : objects are semi-transparent, possibly overlapping and presented in false-color representations based on material density. The objective of this challenge is to detect six classes of objects: *Hammer, Knife, Gun, Wrench, HandCuffs, and Bullet*.

Given the constraint to avoid Deep Learning, my design philosophy focused on maximizing the recall of the window proposal strategy while minimizing the computational overhead of sliding windows as well as the precision of classification with the use of local features, while narrowing the gap between the windows that the model faces during training and during inference. Thus resulting pipeline prioritizes recall during the proposal stage and precision during the classification stage.

2 Pipeline Architecture and Design Choices

X-ray images contain two important types of information: structure (edges/shapes) and density (color). My preprocessing pipeline splits the input into two branches to handle both:

- **Texture (Grayscale and CLAHE):** I converted the image to grayscale and applied Contrast Limited Adaptive Histogram Equalization (CLAHE). CLAHE enhances local contrast, making the edges of metallic objects (guns for example) more distinct. A Bilateral Filter was then applied to reduce grain noise yet preserving these enhanced edges.
- **Density (HSV):** I converted the image to the HSV color space. Indeed, the "Hue" and "Saturation" channels provide good insights for material discrimination, helping to distinguish a metallic gun from a plastic background object that grayscale texture features might miss.

I first tried a sliding window approach, yet it was difficult to tune the shape (aspect ratio) and sizes (scale) of the window as well as the step size. Furthermore it was computationally very heavy to compute. Thus I chose to use Selective Search for object localization. While the sliding window approach generates hundreds of thousands of windows per image, which creates class imbalance, Selective Search groups pixels based on color, texture, and size similarity to generate fewer but higher-quality candidate regions. This improved a lot my inference speed and focused on regions that more often contained objects. I then filtered these proposals based on geometric heuristics (area limits and aspect ratios in $[0.1, 10.0]$) to discard proposals that are unlikely for the target classes.

To account for the various shapes and densities of the objects, I constructed a high-dimensional feature vector combining four complementary descriptors. The features are extracted from resized crops (96×96 pixels):

1. **Histogram of Oriented Gradients (HOG)** to capture rigid shapes. I used 9 orientations and 8×8 pixels per cell. To manage the high dimensionality, I applied PCA to the HOG vector before concatenation, keeping 95% of the variance.

2. **Local Binary Patterns (LBP) & Gray Level Co-occurrence Matrix (GLCM)** to capture texture. Some objects (like handles of hammers) are defined by surface texture rather than just their outline. LBP and GLCM features provide robustness against occlusion where the full shape of the object isn't always visible.

3. **Color Histograms (HSV) :** as color correlates with material density, normalized histograms allow the classifier to learn that "blue" regions are likely metallic parts of dangerous objects for example.

4. **Bag of Visual Words (BoVW) with SIFT :** for complex objects, global shape features can fail under rotation or deformation. SIFT keypoints are scale-invariant. We cluster these into a vocabulary of size 200 to create a histogram of visual words, adding robustness to rotation. We extract the SIFT descriptors and then use a mini batch K-Means to obtain the vocabulary.

For the classifier, I used **XGBoost**, as it performed better than SVM and Random Forest in my experiments and was faster to train. I chose to train a multi-class classifier rather than six separate classifiers (one per class), as this approach was simpler to implement within the limited time available. In addition, I did not identify an effective strategy to properly calibrate and combine the predictions of multiple binary classifiers in order to avoid assigning conflicting labels to the same object.

3 Training Strategy and Hard Negative Mining

The training set was built by extracting Ground Truth crops (positive samples) and random background crops (negative samples). In order to reproduce effectively crops that the model would likely have to deal with during inference, I generated negative samples using a log-normal distribution of window sizes that mimics the statistics of the ground truth objects. Furthermore, a naive classifier trained on random negatives produces many false positives because easy background is easy to classify, but the crops that the classifier faces during inference are harder than those it was trained on. The challenge lies in "hard" background that looks like an object or contains part of an object. To address this, I implemented an iterative Hard-Negative Mining loop :

1. I train the initial XGBoost model.

2. I run the detector on training images.

3. I identify background regions that were incorrectly classified as objects.

4. I add these "Hard Negatives" to the training set with the correct label and retrain.

This process forces the classifier to learn on windows that are alike those that the model will face during inference, and thus narrows the gap with the training data. I capped the number of background and object-class crops extracted from each image by selecting those for which the model was the most confident. However, achieving a proper balance both among the object classes and between the object and background classes was difficult to tune. The hard-negative mining and retraining stages are time-consuming and may take several hours as they require to run the entire detection pipeline repeatedly, but they have proven highly effective and produced the best performance.

4 Post-Processing and Evaluation

The frame score metric penalizes extra predictions severely. While standard NMS merges overlapping boxes of the same class, I observed that a single object might trigger detections for different classes. Thus I implemented a Global NMS strategy. If two bounding boxes overlap significantly, regardless of their class we keep only the one with the highest confidence score. This reduces the number of predictions, improving the precision and the overall score.

5 Conclusion

By combining robust classical feature extraction with ML techniques such as XGBoost and a training loop specifically designed to mine hard negatives, I achieved a decent performance on this difficult task and dataset without using Deep Learning. If I had more time, I would focus on improving the recall for underrepresented classes in the final training dataset, such as *Bullet* and *Knife*, by developing more discriminative features and improving the class balance during the hard-negative mining dataset construction.