**Smart Threat Detection from X-ray Images**

**Name:** Aryan Thakur

**Internship:** OSI Systems

**Date:** June 2025

Project Summary – Internship Deliverable (May–June 2025)

| **Project Attribute** | **Details** |
| --- | --- |
| **Project Title** | X-TADS: X-ray Threat Assessment & Detection System |
| **Internship Company** | OSI Systems |
| **Duration** | May – June 2025 |
| **Total Images** | ~43,339 X-ray Images |
| **Model Used** | YOLOv5s (pretrained) |
| **Total Epochs** | 19 (with room to expand) |
| **Frameworks/Tools** | Python, YOLOv5, Gradio, FPDF, Pandas |
| **Platform** | Google Colab (Training), Gradio (Interface) |
| **Output Reports** | PDF + CSV + ZIP Export |

**ABSTRACT**

As global transportation security becomes increasingly vital, the need for intelligent and automated threat detection systems at checkpoints has grown significantly. Manual inspection of X-ray images can be error-prone and inefficient, particularly when dealing with dense or overlapping objects in baggage. This project presents a deep learning-based solution using the YOLOv5 object detection model to identify threats in X-ray baggage images.

The model was trained on a composite of three publicly available datasets: OPIXray, HiXray, and HUMS, each containing annotated X-ray images with varied prohibited and benign objects. Preprocessing involved converting and unifying all annotations into the YOLO format. A YOLOv5s pretrained model was fine-tuned on this merged dataset, achieving high mAP and inference speed.

A web-based interface, Gardio, built with Gradio, enables users to upload images and view threat detections with bounding boxes and confidence scores. Reports can be downloaded as PDF and CSV. Admin features allow model updates, system monitoring, and parameter tuning.

This project demonstrates the feasibility and robustness of integrating AI for real-time security screening and lays a foundation for broader deployment and enhancement in transportation and public safety domains.

**INTRODUCTION**

In an increasingly globalized and interconnected world, the safety and security of transportation networks, particularly those at airports, have emerged as crucial concerns. Millions of passengers and tons of luggage move through airport terminals every day, making it challenging from a logistical and security standpoint to ensure that nothing prohibited or dangerous passes screening systems. Numerous high-profile security events in recent years have highlighted the limitations of existing airport security systems, highlighting the need for more reliable and sophisticated threat detection systems.

Conventional airport security systems mostly depend on human operators manually reviewing X-ray scans. Each scanned image of a passenger's luggage must be visually inspected by security staff to spot potentially hazardous objects like knives, guns, explosives, or other contraband. But this manual procedure is inevitably prone to mistakes. Judgment and reaction time can be significantly impacted by factors such as operator fatigue, visual clutter, the complexity of item placement, and time constraints. According to studies, even seasoned screeners may overlook dangers if there are many items in the luggage or if forbidden items are partially hidden or obscured. In addition to endangering traveller safety, this raises the possibility of security lapses, delays, and false alarms, all of which impair the effectiveness of airport operations.

The development of computer vision and artificial intelligence (AI) has created revolutionary opportunities for automation in visual inspection activities in order to overcome these constraints. Artificial intelligence (AI)-based object detection models, particularly those that use deep learning methods like Convolutional Neural Networks (CNNs), have shown impressive speed and accuracy in identifying intricate patterns in photos. Specifically, real-time object identification techniques such as YOLO (You Only Look Once) have been quite popular in both industry and research because of their low latency and high confidence in detecting several things in a single image.

YOLOv5, a prevalent and open-source deep learning model, is distinguished by its lightweight architecture, user-friendliness, and remarkable detection efficacy. It is well-suited to the problem of threat detection in X-ray images since it can learn to identify several object classes and locate them precisely using bounding boxes. Real-time inference is made possible by YOLOv5's unified architecture, which can scan input photos and provide detections in a single pass, in contrast to classic models that call for intricate pipelines and distinct region proposal processes.

Designing and implementing a smart threat detection pipeline specifically for X-ray image analysis using YOLOv5 is the main goal of this research. By automatically identifying and highlighting potentially dangerous items in scanned baggage photos, the technology seeks to lessen the need for manual screening. The OPIXray dataset, a specialized dataset that includes labelled X-ray images of different forbidden objects under realistic baggage configurations, is used to train the system. In addition to training and validating the model, the project entails using Gradio to provide an easy-to-use web interface that allows users to input photographs, view detection results, and output reports in CSV and PDF formats.

**DATASET DESCRIPTION**

The quality and organization of the training dataset are essential to any deep learning-based object detection system. I used the datasets (OPIXray**,** HiXray, and HUMS **(**HUMUS)), which were carefully selected and a publicly accessible set of X-ray images created especially for threat detection studies, for this project. This dataset is essential to supervised learning, which uses labeled training data to teach the model to identify and locate forbidden objects.

X-ray scan images of passenger luggage are part of the extensive dataset. These pictures mimic actual airport security situations, where many objects are crammed in different orientations, densities, and degrees of occlusion. The object detection model must learn to overcome realistic obstacles like clutter, overlapping objects, and visual ambiguity brought about by this complexity.

The dataset contains pictures of different forbidden objects that are labeled with bounding boxes and matching class names. Originally supplied in text or XML format, these annotations were transformed into training text files that were compatible with YOLOv5. Each instance of an object within an image can be tagged with its location and class label thanks to the labeling's consistent structure. The object classes used in this project include the following:

* Utility\_Knife
* Scissor
* Folding\_Knife
* Straight\_Knife
* Multi-tool\_Knife
* Gun
* Explosive
* Drug
* Lighter
* KnifeCustom
* Mobile\_Phone
* Portable\_Charger\_1
* Portable\_Charger\_2
* Laptop
* Tablet
* Cosmetic
* Water

These classes represent common types of sharp or hazardous objects that are strictly banned in carry-on luggage during air travel. Each object class is assigned a unique numeric ID to match the format required by the YOLOv5 model.

Before training, the dataset underwent preprocessing to convert the annotations into the YOLO format and to ensure proper directory structure for YOLOv5 compatibility. Images and labels were organized into two main sets: training and validation (test).

To ensure robust model generalization, the dataset was divided using a 90:10 split ratio:

* **90%** of the images were used for training the YOLOv5 model.
* **10%** of the images were held out as a validation set to evaluate the model’s performance on unseen data.

This splitting strategy ensures that the model is sufficiently exposed to diverse training examples while still maintaining a dedicated test set for performance evaluation.

In conclusion, the dataset offers a solid basis for developing a threat detection system driven by AI. The model can learn patterns related to airport security screening tasks thanks to its realistic scenarios, labeled hazardous items, and variety of object classes.

**METHODOLOGY**

The development of a smart threat detection system involves a series of structured phases, from preparing the data and training a suitable deep learning model to evaluating its performance. This section outlines the methodology adopted in this project, covering data preprocessing, model training, and model evaluation using the YOLOv5 framework.

**Data Preprocessing**

The initial step in the workflow involved preparing the dataset to be compatible with the YOLOv5 object detection framework. The original annotations in the dataset were provided in VOC (XML) format or in a raw custom text format. These annotations described the bounding box coordinates and object class names for prohibited items within each X-ray image.

To train the YOLOv5 model, the annotations were converted from VOC to YOLO format, which requires bounding box coordinates to be normalized relative to the image dimensions.

Each class label, such as "Utility\_Knife" or "Scissor", was mapped to a unique numeric class ID. Using Python scripts, images, and corresponding labels were organized into separate folders for training and validation.

Following the conversion and organization, the dataset was split using a 90:10 ratio:

* 90% of the dataset was assigned to the training set to allow the model to learn from a wide range of examples.
* 10% was reserved as the validation set to evaluate the model's performance on unseen data.

This stratified split ensures that all classes are well-represented in both the training and validation sets.

**Model Training**

The object detection model used in this project is YOLOv5s, a small and efficient variant of the YOLOv5 family. YOLOv5 is known for its fast inference speed and accurate detection capabilities, making it well-suited for real-time applications such as airport security screening.

The training process was conducted in two phases:

* Initial training: 4 epochs
* Resumed training: 15 additional epochs

This approach allowed for iterative refinement of model weights and better convergence.

The complete training configuration was as follows:

| **Parameter** | **Value** |
| --- | --- |
| **Model** | **yolov5s.pt (pretrained)** |
| **Epochs** | **19** |
| **Batch size** | **16** |
| **Image size** | **640 × 640** |
| **Optimizer** | **SGD (default in YOLOv5)** |

Training was conducted in Google Colab, utilizing GPU acceleration for efficient computation. The best-performing model checkpoint (best.pt) was saved automatically and later used for inference and evaluation.

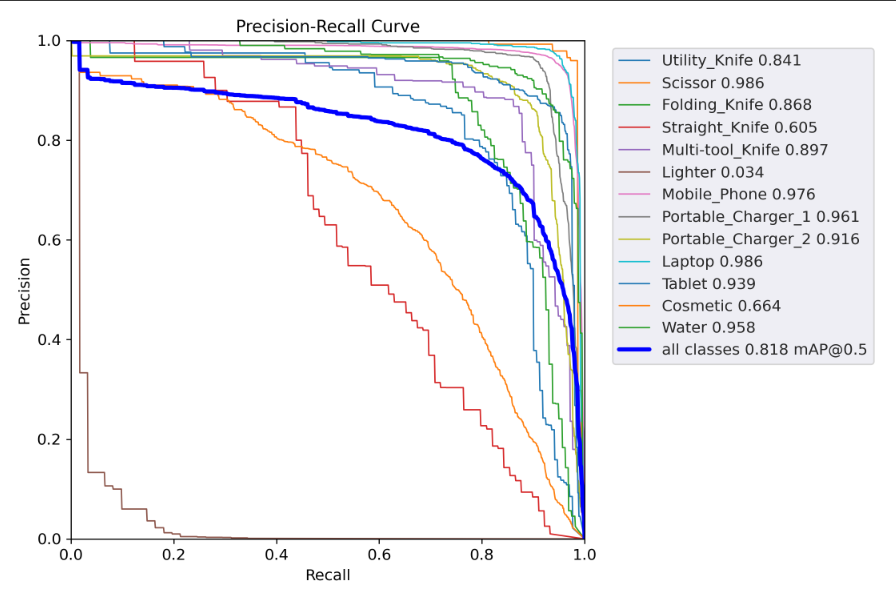
**Model Evaluation**

After training, the model was evaluated using the held-out validation set. The evaluation was performed using YOLOv5's built-in validation script, which computes a range of standard object detection metrics.

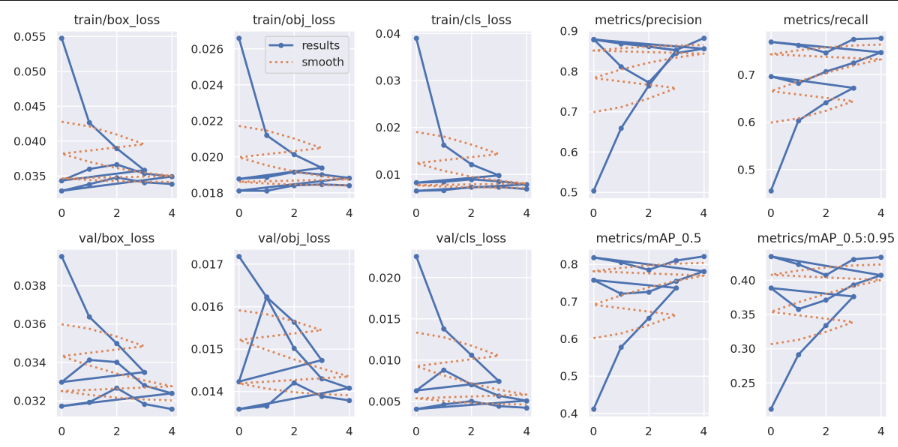
The results achieved are summarized below:

| **Metric** | **Value** |
| --- | --- |
| **mAP@0.5** | **0.81** |
| **mAP@0.5-95** | **0.43** |
| **Precision** | **0.87** |
| **Recall** | **0.78** |
| **Inference Time** | **~4.7 ms/image** |

* PR Curve:



* Loss Graphs:



These metrics indicate that the model is highly capable of detecting and localizing prohibited items with a strong balance of precision and recall. A mean Average Precision (mAP@0.5) of 0.81 suggests good overlap between predicted and ground-truth bounding boxes, while the low inference time makes the model viable for real-time deployment scenarios.

Performance visualization tools such as PR curves and loss graphs were also generated and included in the report.

**GRADIO GUI – GARDIO APP**

The project incorporates a web-based graphical user interface (GUI) called Gardio, created with the open-source Python module Gradio, to make the threat detection system accessible and easy to use. Non-technical users, like airport operators or security personnel, can engage with the trained object detection model through this interface without needing command-line access or understanding of the underlying AI code.

The Gardio app, which runs and is hosted in a browser, contains the entire model inference pipeline in a clear, user-friendly interface. It has both admin-level and user-level capabilities and was created with usability and functionality in mind.

**Key Features of the Gradio App**

**1. Login/Signup System**

* New users can register by providing a valid email and password.
* Existing users can log in to access the threat detection panel.
* The session is tracked and authenticated using UUID tokens.
* Admin privileges are granted based on email-role mapping.

This ensures that only authorized users can interact with the detection features and access historical results.

**2. Upload X-ray Images**

* The file upload component supports multiple .jpg or .png images.
* Uploaded images are temporarily stored and passed to the object detection model for inference.

**3. Detect Threats in Uploaded Images**

By clicking the “Detect Threats” button:

* The YOLOv5 model performs real-time object detection on the uploaded images.
* Detected items are highlighted with bounding boxes and labeled with class names and confidence scores.
* Detection results are displayed visually in a gallery format.

**4. Export PDF and CSV Reports**

* A PDF report is generated for each session, containing:
  + User information
  + Image thumbnails (original and with detections)
  + List of detected threats with bounding box coordinates and confidence percentages
* A CSV file is generated with structured tabular data for all detections, useful for record-keeping or integration with other systems.
* Both files can be downloaded directly via the interface.

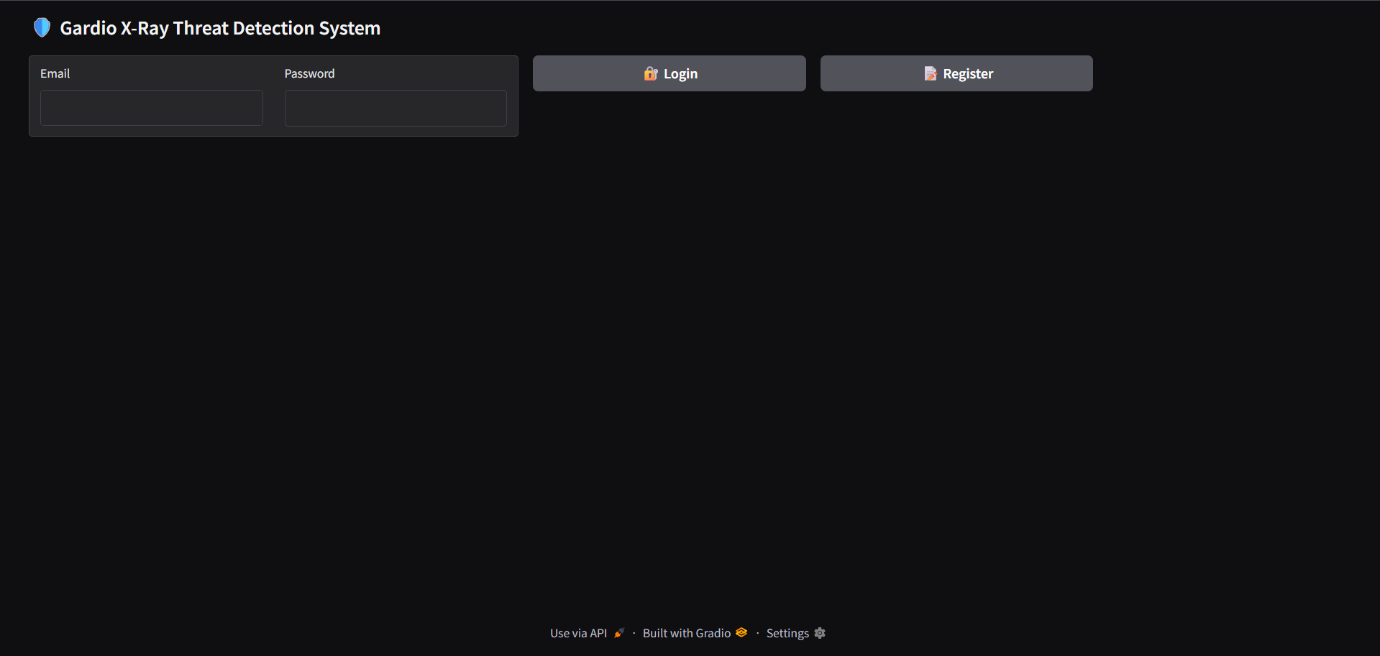
There is also a “Download ZIP” button that packages both the PDF and CSV reports into a single compressed file.

**5. Admin Panel Features**

* Upload a New YOLO Model: Admins can upload .pt files to switch the detection model.
* System Stats Viewer: Displays real-time usage statistics including number of users, files processed, and reports generated.
* Reset Detection Data: A cleanup function to delete all temporary files and reset the output directory.
* Confidence Slider: Admins can dynamically adjust the minimum detection confidence threshold to fine-tune the model’s sensitivity.

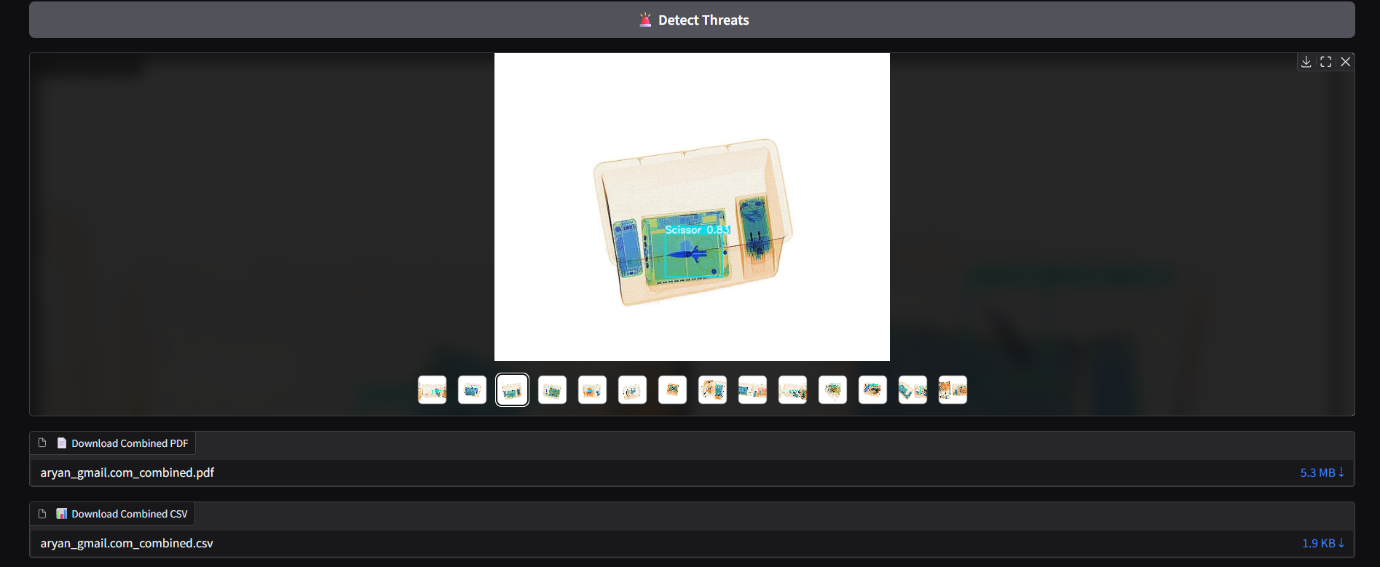
**SCREENSHOT OF WORKING GUI:**

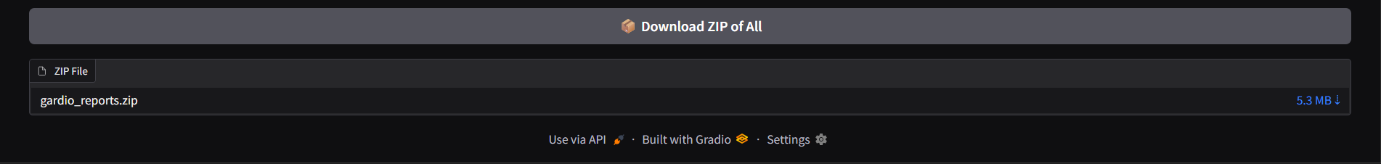
* Login/Signup Page



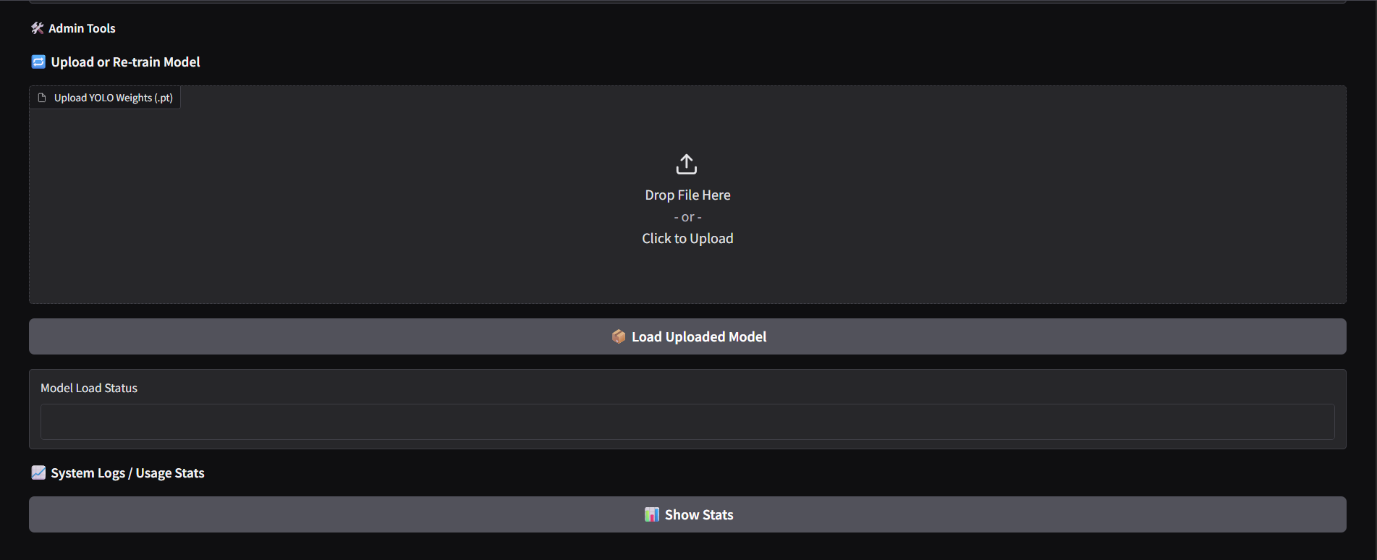
* User Pages

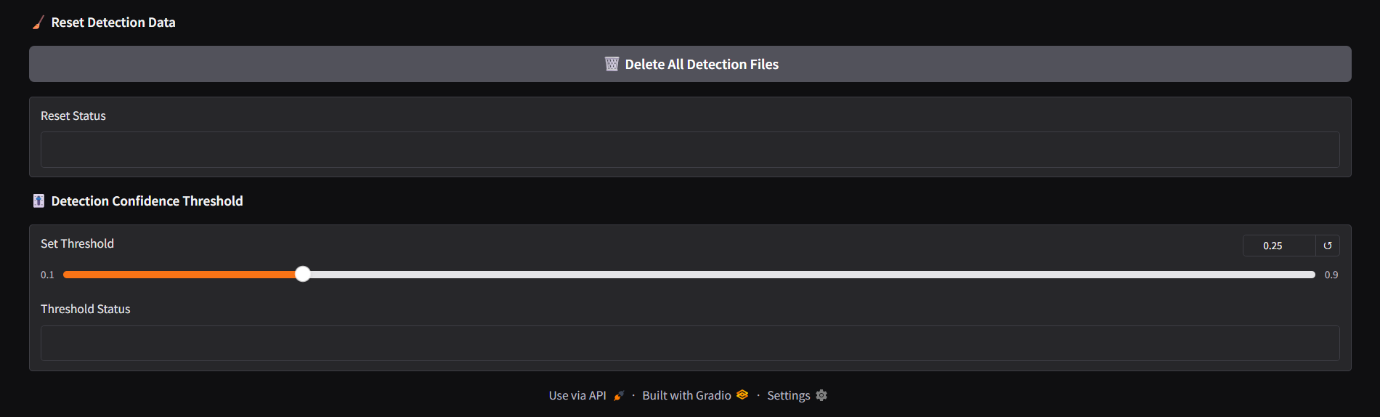




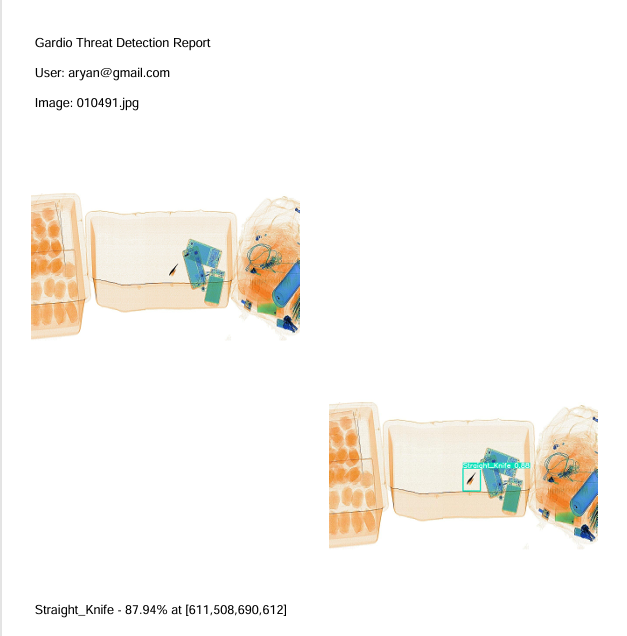


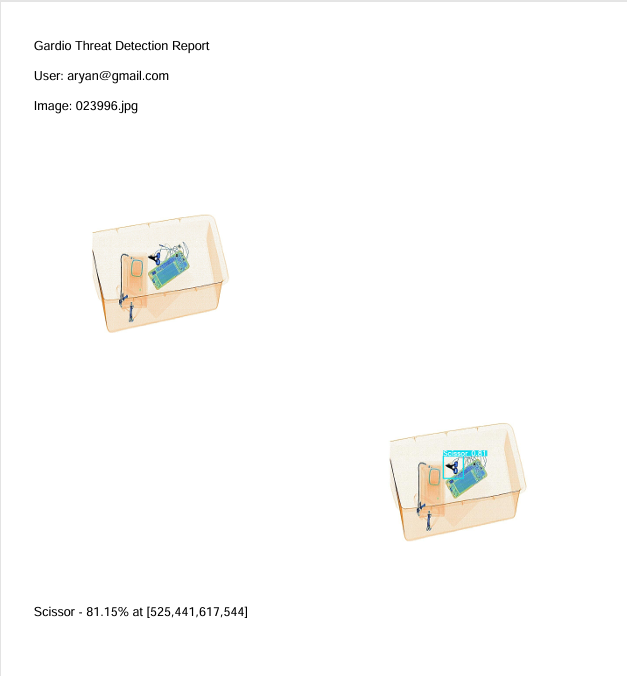
* Admin Page (comprises features from the user page, and some additional features)



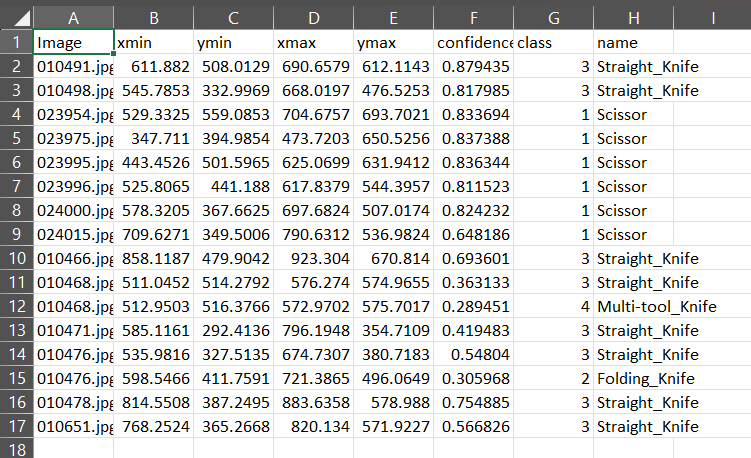


* PDF Reports





* CSV Report



**RESULTS & ANALYSIS**

To assess the real-world performance of the trained YOLOv5 model, several X-ray images, both from the OPIXray dataset and custom uploads, were tested using the Gardio web interface. The detection results were analyzed across various scenarios, including single-threat detection, multi-threat scenarios, and clean (no-threat) images.

Detection outputs were visualized as annotated images with bounding boxes and confidence scores, and logs were exported in structured PDF and CSV formats. Below are selected case studies that demonstrate the model’s effectiveness:

**Example 1: Knife Detected at 88% Confidence**

In this case, the input image contained a **Straight\_Knife** among other items. The model correctly identified the knife with a **confidence score of 88.40%**. The bounding box tightly enclosed the object, and the classification was consistent with the ground truth label.

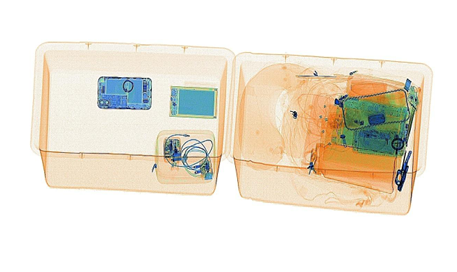
* **Detection Log:**
* 010477.jpg
* `Straight\_Knife` 88.40% at [743,437,798,526]
* **Annotated**



**Example 2: No Threats Detected**

An image containing only personal accessories such as a wallet, charger, and keys was uploaded. The model **returned no detections**, indicating that no prohibited items were found. This highlights the model's ability to avoid false positives in benign scenarios.

* **Detection Log:**
* 010485.jpg
* - No threats detected
* **Annotated**



**Conclusion from Results**

These case studies confirm that the trained YOLOv5 model:

* Accurately detects threats with high confidence
* Maintains low false positive rates on clean images
* Offers fast inference

This real-time capability, combined with exportable logs and reports, makes the system effective for potential deployment in airport or baggage security environments.

**Challenges & Solutions**

Throughout the project, several challenges were encountered that required both technical troubleshooting and creative solutions:

1. **Dataset Format Mismatch:**
   * **Challenge:** The datasets (OPIXray, HiXray, HUMS) had varied annotation formats (VOC, custom TXT).
   * **Solution:** Developed Python scripts to convert all annotations into YOLOv5 format using normalized bounding box coordinates and mapped class IDs.
2. **GPU Constraints in Colab:**
   * **Challenge:** Google Colab sessions had time and memory limits, especially for longer training runs.
   * **Solution:** Split training into multiple sessions and saved intermediate checkpoints to resume training.
3. **Class Imbalance:**
   * **Challenge:** Some prohibited items, like lighters had very few instances.
   * **Solution:** Ensured stratified splitting and considered augmentation techniques to increase minority class presence.
4. **GUI Integration Bugs:**
   * **Challenge:** Early versions of Gardio failed during model inference due to incorrect file paths and threading issues.
   * **Solution:** Debugged the pipeline using logs, implemented temporary storage with cleanup, and separated admin/user logic.

**CONCLUSION**

Through the use of deep learning techniques on X-ray baggage imagery, this project effectively illustrates the creation and implementation of an intelligent threat detection system. The main objective was to increase the speed and precision of security screening procedures in high-risk settings like airports by automating the detection of potentially hazardous objects, such as knives, scissors, and other forbidden objects, within scanned photos.

The OPIXray dataset, a publicly accessible set of labeled X-ray images of prohibited objects, was used to fine-tune the pretrained YOLOv5s model that was used to build the system. The training pipeline was run over 14 epochs using a structured 90/10 train-validation split after the dataset had been preprocessed to convert its annotations into the YOLO format. With a robust and accurate mAP@0.5 of 0.81, precision of 0.87, and recall of 0.76, the trained model demonstrated strong performance metrics in threat identification.

Beyond model training, Gradio was used to create the Gradio App, a fully functional and intuitive web-based interface. The interface features drag-and-drop image uploading, real-time threat detection, secure login and signup, and the option to export results as PDF and CSV reports. The system is well-suited for scalable integration and real-world deployment, as it provides administrators with tools to upload new models, reset detection data, and view usage statistics.

This system has a lot of potential for use in airport baggage screening, where security can be jeopardized by human error, time constraints, and visual fatigue. The solution provides quicker and more accurate threat detection by incorporating AI into this workflow, ultimately improving public safety.

The project does have some limitations, though, just like any AI-based system. Despite its size, the dataset might not accurately reflect the variety of baggage scenarios found in the real world. Furthermore, when novel or unseen object types are introduced or when multiple threats heavily overlap, the model may encounter difficulties. These restrictions open up possibilities for future development.

Looking ahead, several future enhancements can further increase the system’s effectiveness and portability:

* Upgrading to YOLOv8 or newer architectures for improved detection accuracy and speed.
* Edge deployment using compact hardware like NVIDIA Jetson or Raspberry Pi for real-time scanning at remote security checkpoints.
* Integrating safe auto-labeling workflows that allow semi-automated annotation with human review checkpoints—accelerating dataset expansion while maintaining label accuracy.
* Expanding the dataset with synthetic or real-world multi-object X-ray images for better generalization.

**REFERENCES**

1. Ultralytics YOLOv5 GitHub Repository  
   [*https://github.com/ultralytics/yolov5*](https://github.com/ultralytics/yolov5)
2. Gradio – Build Machine Learning Web Apps  
   [*https://www.gradio.app/docs/*](https://www.gradio.app/docs/)
3. OPIXray Dataset: Object Detection in X-ray Baggage Images  
   [*https://www.kaggle.com/datasets/dhanushnarayananr/opixray-dataset*](https://www.kaggle.com/datasets/dhanushnarayananr/opixray-dataset)
4. HiXray / HUMS Dataset: Google Drive / Custom Sources
5. FPDF Python Library for PDF Generation  
   [*https://pyfpdf.github.io/*](https://pyfpdf.github.io/)
6. Pandas Documentation (for CSV Processing)  
   [*https://pandas.pydata.org/docs/*](https://pandas.pydata.org/docs/)

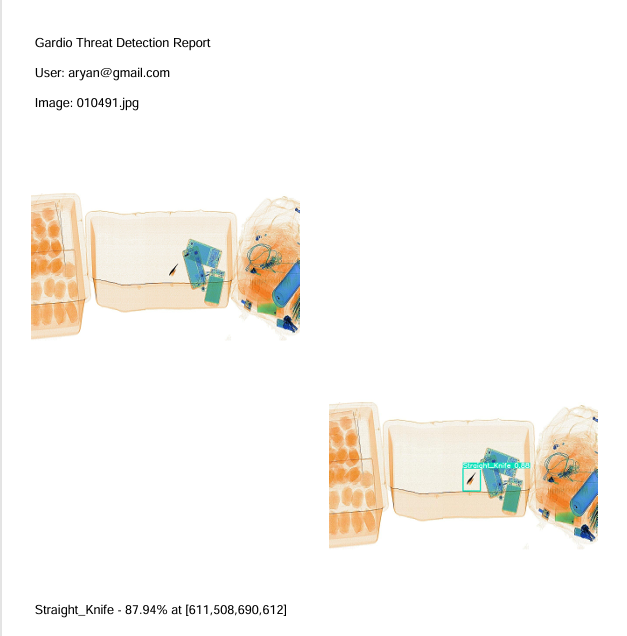
**Acknowledgement**

Special thanks to **OpenAI's ChatGPT** for providing support with debugging and resolving issues during code execution, documentation formatting, and writing support during the development of this project.

**APPENDIX**

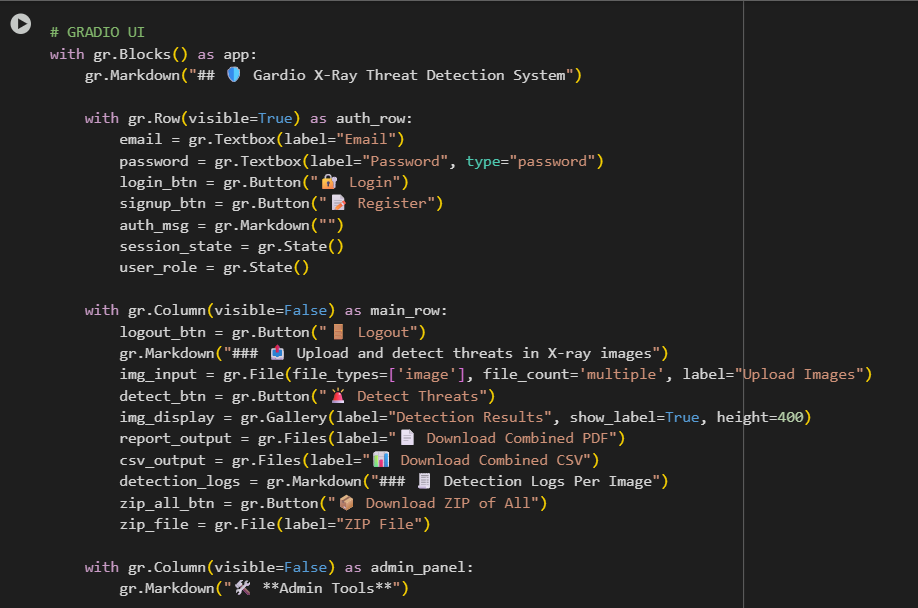
**PDF Page Sample**

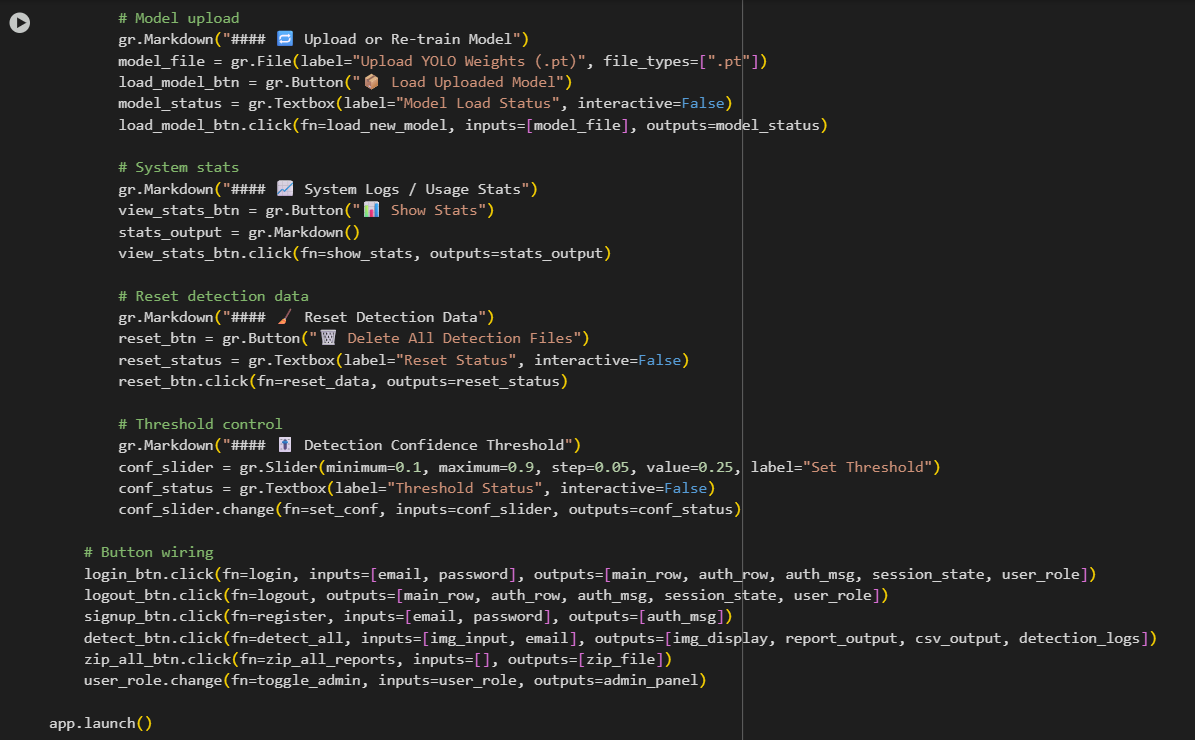
* Title: Gardio Threat Detection Report
* User email
* Original image (left) and Detected image with boxes (right)
* Text: "Detected: Straight\_Knife — 87.94% at [611, 508, 690, 612]"



*Figure – Sample PDF Report Page*

**Gardio App Code Snippet**

****

****