Prohibited Item Detection in X-ray Scans with Faster R-CNN

Nikolaos Prevolis | Stefanos Tzaferis
Eleftheria Galatsiatou | Peter Kayiwa

Spring Semester 2025

Contents

This report is part of our project for the class “AI lab: computer vision and NLP” during our erasmus mobility semester and is accompanied by a set of python files and a ipynb notebook showcasing the training process.

Chapter 1

Abstract

Millions of pieces of luggage are being scanned in the world's airports every day. In most airports the process is still overseen by a human.

This project aims to assist these people in their job, by highlighting potentially suspicious items, making the risk of oversight and human error smaller while rendering the security checks easier and safer for everybody involved.

This paper presents the design and implementation of X-Ray R-CNN, a deep learning model based on the Faster R-CNN architecture, built to identify prohibited items in baggage scans. We leverage a pre-trained VGG16 network as a feature-extraction backbone and develop custom Region Proposal Network (RPN) and Region of Interest (ROI) Head modules based on the implementation described in the original work [1]. The model is designed to operate on the Hi-Xray dataset [3], a large-scale collection of security inspection images containing eight classes of prohibited items.

We detail the entire pipeline, from data preprocessing and anchor generation to the two-stage training process and loss calculations, presenting a complete framework for this challenging detection task.

The model is in the end exported into a user-friendly UI using opencv and gradio.

Chapter 2

The Data

The High-quality X-ray security inspection dataset contains a number of images from airport scanners and their annotation files. All images of HiXray dataset are annotated manually by professional inspectors from an international airport, and the standard of annotating is based on the standard of training security inspectors. The dataset is comprised of 45364 images – 36295 for training and 9069 for testing.

The 8 classes introduced are the following:

- Portable Charger (1)
- Portable Charger (2)
- Mobile Phone
- Laptop
- Tablet
- Cosmetic
- Water
- Non metallic Lighter

The images in the dataset are of arbitrary sizes so there is the need to transform them before feeding them into the neural network.

2.1 The custom dataset

For the needs of the project a custom dataset had to be implemented so the data can be used in the training and evaluation of the model. The custom dataset format has three objectives:

Establishing Data-Label Correspondence

A fundamental challenge in any supervised learning task is linking an input to its correct label. The script establishes a simple but powerful rule: for every image named X.jpg, its corresponding ground-truth information must be in a file named X.txt. This organizational principle is the foundation upon which the entire data loading process is built. It ensures that the model is always shown an image and told the correct answer for that specific image.

Data Standardization and Normalization

An image is a collection of pixels, but their values and dimensions can vary wildly. The transform pipeline addresses this. It enforces a standard "grammar" on the input data: It ensures all images are represented as numerical tensors and it scales pixel values to a consistent range (e.g., 0 to 1), preventing large pixel values from disproportionately influencing the model's learning. This step is conceptually equivalent to ensuring all ingredients in a recipe are measured using the same units (e.g., grams, not a mix of cups and ounces).

Structuring Information for Task-Specific Learning

Our model needs to perform two distinct tasks: classifying objects and localizing them with a bounding box. It cannot learn if the information is presented as a single, undifferentiated block of data. The script solves this by structuring the output into a package. The image tensor is kept separate from the target dictionary. Within the target, bounding box coordinates (bboxes) are kept separate from class IDs (labels).

This structure is critical because the different parts of the model's loss function know exactly where to look for the information they need. The

classification loss function looks at the labels, while the bounding box regression loss function looks at the bboxes. This separation of information is essential for the multi-task learning that defines modern object detectors.

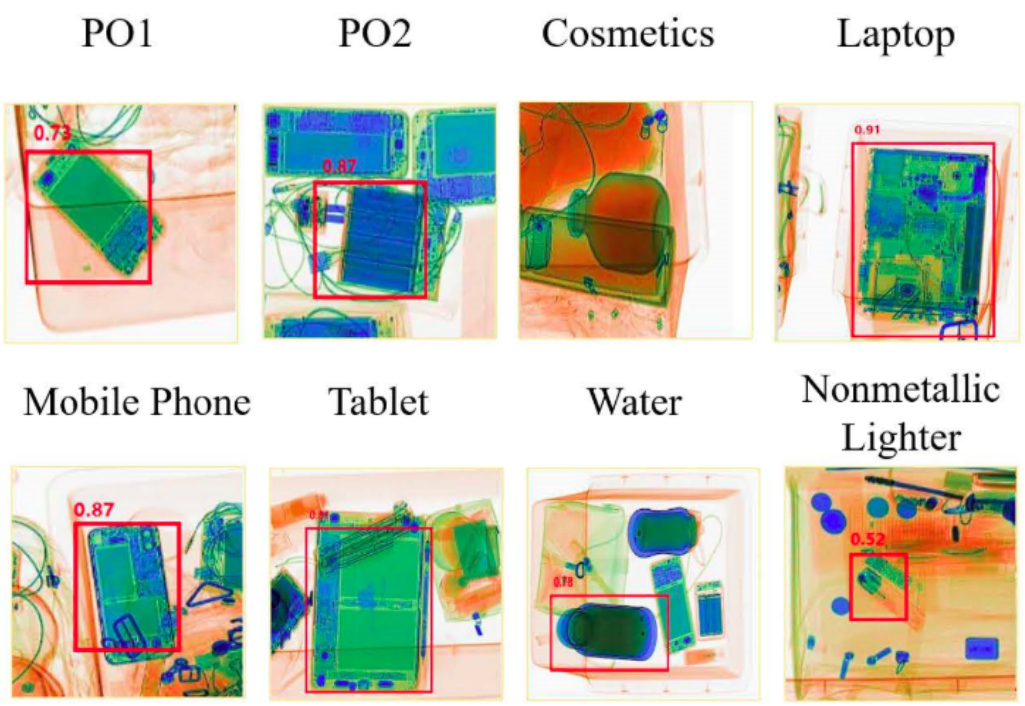

Figure 2.1: Example for each class

Chapter 3

The Model

For the academic purposes of this project a model was implemented from scratch rather than using a pre-trained faster r-cnn model.

Faster R-CNN, introduced by Ren et al. in 2015, is a two-stage object detection framework that builds on earlier R-CNN models (R-CNN and Fast R-CNN), achieving significant speed and accuracy improvements.

It unifies region proposal and object detection into a single, end-to-end trainable network. It uses a Region Proposal Network (RPN) to efficiently generate candidate object regions, eliminating the need for slow external proposal methods like selective search.

3.1 Architecture Overview

Faster R-CNN consists of the following key components:

- Convolutional Backbone (Feature Extractor)
- Region Proposal Network (RPN)
- Region of Interest (RoI) Pooling
- Classification and Regression Head

3.1.1 Convolutional Backbone

The input image is passed through a pre-trained convolutional neural network (this implementation uses a VGG16 pre-trained model). The output is a feature map that encodes spatial and semantic information of the image.

3.1.2 Region Proposal Network (RPN)

The RPN is a fully convolutional network that slides a small network over the feature map.
At each spatial location, it:

- Uses anchor boxes. Anchors are predefined boxes of different sizes placed on the feature map. Each one represents a possible object location.
- Predicts whether an anchor contains an object (objectness score).
- Refines the anchor coordinates via bounding box regression.

The RPN network predicts whether each anchor box is background or foreground.

It outputs a set of region proposals, each defined by its coordinates and score.

It also aims to refine the anchor boxes as to align them better with the actual objects.

Loss function is defined by a combination of the classification loss, that reflects the rpn's ability to distinguish between foreground and background and the regression loss, that represents its ability to fit objects within the anchor boxes precisely.

Layer	Type	Input Dim	Output Dim	Activation
1	Conv2d	3*3	512	ReLU
2	Classification (Conv2d)	1*1	9	-
3	Regression (Conv2d)	1*1	36 (9x4))	-

Table 3.1: RPN architecture

3.1.3 Region of Interest (RoI) Pooling

These region proposals are used to crop and resize regions from the feature map into a fixed spatial size (7x7). This step ensures that each proposal can be processed by fully connected layers regardless of its original size.

3.1.4 Detection Head (Classifier and Regressor)

Each pooled RoI is passed through fully connected layers.The network outputs two things, the Softmax classification scores and the bounding box refinements to improve localization accuracy.

Finally, Non-Maximum Suppression (NMS) is applied to filter overlapping predictions.

For simplicity's sake, the model caters only to single batch input.

Layer	Type	Input Dim	Output Dim	Activation
1	Linear	512*7*7	4096	ReLU
2	Linear	4096	4096	ReLU
3	Classification (Linear)	4096	9 (8 classes + BG)	-
4	Regression (Linear)	4096	36 (9x4)	-

Table 3.2: ROI head architecture

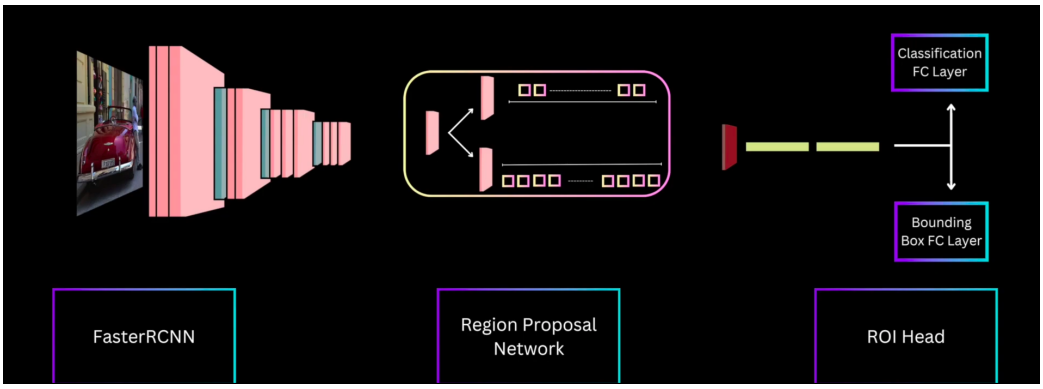

Figure 3.1: Visualization of the model

3.1.5 Intersection over Union

IoU (Intersection over Union) is a metric that measures the overlap between two bounding boxes. Two overlapping boxes of different sizes shouldn't be considered equally good if one almost completely contains the other.

By dividing by the union, IoU penalizes predictions that are too large or too small, even if they partly overlap. IoU is scale-invariant: It doesn't matter whether the objects are large or small; the IoU stays consistent as a quality metric. This makes it ideal for object detection where objects can appear at different sizes in images.

Therefore it is very a very useful tool for the objectness score assignment during the pass from the RPN and for the NMS.

3.2 The Training

The entire network is trained end-to-end using a multi-task loss function. The total loss is the sum of four individual losses: RPN classification loss, RPN regression loss, ROI Head classification loss, and ROI Head regression loss. The RPN loss is defined as:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

where i is the index of an anchor, p_i is the predicted objectness probability, and p_i^* is the ground-truth label (1 for positive, 0 for negative). t_i and t_i^* are the predicted and ground-truth bounding box regression parameters, respectively. L_{cls} is Binary Cross-Entropy loss, and L_{reg} is Smooth L1 loss. The ROI Head loss is analogous, but L_{cls} becomes Cross-Entropy loss over all classes.

The model was trained using a Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.001 and momentum of 0.9. A 'MultiStepLR' scheduler was used to decay the learning rate during training. The resulting accuracy after evaluating

Chapter 4

Inference and UI

4.1 The script

For practical application and evaluation, the project is equipped with a comprehensive inference script. This script provides a versatile interface for applying the trained X-Ray R-CNN model.

It supports multiple modes of operation: a full evaluation mode that processes the entire test dataset to compute the mean Average Precision (mAP) for a quantitative performance analysis, and a sample inference mode for generating qualitative visual examples. Furthermore, the application is designed for interactive, single-image analysis. A user can provide a path to a specific image—a feature intended to be the backend for a simple user interface (UI).

In this mode, the script loads the image, performs inference, and saves a new annotated version with predicted bounding boxes, class labels, and confidence scores drawn directly onto the image, allowing for immediate visual feedback on the model's predictions for any given scan. This allows for the network to be used in a simulated real-life application.

This was made possible with OpenCV and Gradio libraries.

4.2 Mean Average Precision (mAP)

Neither precision nor recall alone tells the whole story. The mean Average Precision (mAP) is a sophisticated metric designed to capture the model's

performance across the entire precision-recall spectrum.

For each class of object (e.g., "Laptop"), the model's predictions are ranked by their confidence score. By moving down this ranked list, we can plot a precision-recall curve. The Average Precision (AP) is essentially the area under this curve, providing a single number that summarizes the quality of detections for that one class.

The "Mean": The mAP is simply the average of the AP scores across all object classes. This provides the final, single, all-encompassing metric to judge the model's overall performance. A higher mAP score indicates a better model.

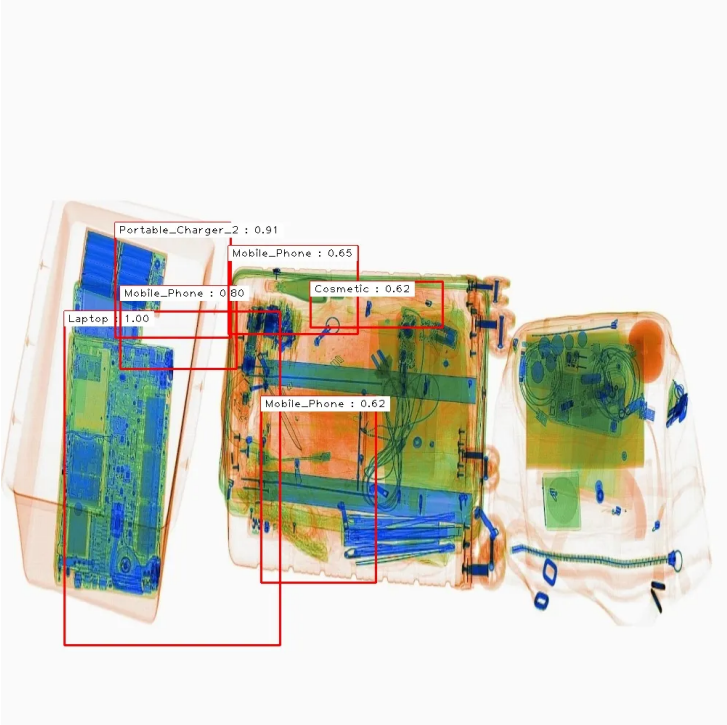

Figure 4.1: Single image output with predictions

Chapter 5

Conclusions and Future Work

Object detection is a complicated and multi-factor problem in computer vision. At this point there are many tools like frameworks and pretrained models that can assist on the development of computer vision applications like the one presented on this project.

In the future this project can be expanded and improved by performing more training loops and experimenting more with the many hyperparameters of the model. Pretrained networks like pytorch's Faster R-CNN and Yolo can be utilised to compare and challenge the model made from scratch. Also it could be extended with more data and modified to support more item classes.

Chapter 6

Bibliography

[1] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[3] W. Miao, et al., "Hi-Xray: A large-scale dataset for prohibited item detection in security inspection images," in *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 136-145.

[4] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, “Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.