



UML501

Machine Learning

Project report

Detection of weapons in baggage X- Rays using Faster R-CNN

Submitted By:

Abhijeet Singh Malhi 101603006

Akspreet Kaur 101603026

Amandeep Singh 101610007

Abstract

Scanning baggage by x-ray and analyzing such images have become important technique for detecting illicit materials in the baggage at Airports and other security checkpoints. This project is based on the method for detecting concealed weapons, like gun, needle, etc. in the baggage by employing image segmentation method to extract the objects of interest from the image followed by applying feature extraction methods. Finally, the objects are classified using Faster R-CNN as illicit or non-illicit object. The dataset used for training and testing purposes was GDXray-Grima database of X-ray images. The model accurately predicts the presence of guns, knives, razor-blades and pins in the images.

1. Introduction

1.1. Need of the System:

The increasing threat nowadays by terrorists especially in airports revealed the necessity to enhance the checking equipments and techniques to detect and recognize concealed weapons. Today, most of the criminal activities are taken place using hand- held arms particularly gun, pistol and revolver. The crime and social offence can be reduced by monitoring such activities and to identify antisocial behavior so that law enforcement agencies can take appropriate action in early stage. The solution of this problem is to deploy surveillance system or control cameras with automatic handheld gun detection and alert system.



Fig 1.1 X-ray scan of a bag checked in at the airport

As seen in the above image, today's X-Ray systems are effective in producing an accurate image, however, they are prone to human errors as they need to be continuously monitored. In last few years, deep learning has established a landmark in the area of machine learning particularly in object detection, classification and image segmentation. Faster R-CNN achieved best results so far in classical image processing problems such as image segmentation, classification, detection and coordinates of the object in several applications.

1.2. Applications of Proposed System:

The application in this project, once embedded along with the X-ray machine softwares at various security checkpoints will be able to detect the presence of illicit items in real time. Once a reliable model is trained and tested, it can even eliminate the need for any human interference. Moreover, the script will be able to work on any type of X-ray machine if trained with the correct set of images. Any sort of weapon can be detected using a pre-trained R-CNN model.

1.3. Challenges in development:

The initial course of action for our team was to build a CNN model that could classify whether the image had a weapon or not. However, that presented two difficulties:

- i) The model would cause problems if the bag had multiple illicit items.
- ii) This type of model was just not practical.

After this we shifted our focus on developing an object detector that not only classifies the image, but also tells us where the weapon lies in the image but can also detect multiple types of weapons depending on the dataset that was used for its training.

2. Existing Work

For the detection of threat items, many types of imaging system exist. X-ray imaging systems and MMW (Millimeter wave imaging) are used and x-ray imaging system is widely used for carry-on bags. The techniques used for analyzing these x-ray images are pseudo-coloring and segmentation-based techniques. Pseudo-coloring process is the one in which the objects inside the bag are given different colors based on their material type. In segmentation-based methods, the x-ray images are segmented to extract the objects of interest. Using these methods, satisfactory results are produced and assisted human for detecting the threat items. X-ray photons, however, penetrate most materials. As a result, all objects along an x-ray path attenuate the x-ray and contribute to the final measured intensity. In the x-ray community, a common way of disambiguating objects is through CT reconstruction. This is typically obtained through the filtered back-projection algorithm. Although several X-ray technologies based automatic systems exist for threats detection, only a few of these systems make use of the well-established pattern recognition and machine learning techniques. A research publication from NIT Kurukshetra also makes use of the faster R-CNN model to detect handheld guns which can be useful in CCTV surveillance.

3. Data Collection and Data Preparation

3.1 Data Collection

The GDX-ray Grima database provides a comprehensive set of X-Ray images of baggages containing various items like guns, knives etc. These X-Ray images were taken by rotating the baggage through 360 degrees and clicking pictures after a certain interval. This gave us enough data so as to make our model capable enough to detect weapons no matter at what angle or position they were kept in the bag. However, due to the huge amount of images, limited computing power and limited manpower we used a relatively small set of 498 images for building this model.

Out of these 100 images were kept for testing and 398 images were kept for model training.



Fig 3.1 A sample image in the dataset used for training

3.2 Data Preparation

LabellImg, a python based tool, was used to mark the locations of weapons in every image of the training as well as the test set. The tool generates an xml file for every item in an image, which contains the coordinates of the rectangular box that is holding the item. This helps the model to learn only certain specific features of the image.



Fig 3.2 Bounding boxes made using LabellImg tool

These xml files are combined to form a csv file, so that it can be easily handles by our python script to train the model.

```
def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                    int(root.find('size')[0].text),
                    int(root.find('size')[1].text),
                    member[0].text,
                    int(member[4][0].text),
                    int(member[4][1].text),
                    int(member[4][2].text),
                    int(member[4][3].text)
                    )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df
```

Fig 3.3 Code snippet used to combine the xml files

	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	B0001_000	2688	2208	blade	1715	1040	2072	1317
3	B0001_000	2688	2208	blade	1690	1017	2054	1349
4	B0001_000	2688	2208	blade	1668	1027	2040	1347
5	B0001_000	2688	2208	blade	1629	1022	2020	1349
6	B0003_000	900	1430	blade	284	658	321	969
7	B0003_000	900	1430	blade	278	636	306	969
8	B0003_000	900	1430	blade	277	619	302	932
9	B0003_000	900	1430	blade	287	585	308	930
10	B0005_000	850	850	blade	270	251	529	441
11	B0005_000	850	850	pin	231	245	391	393
12	B0005_000	850	850	pin	282	492	479	601
13	B0005_000	850	850	pin	273	490	476	593
14	B0005_000	850	850	pin	215	272	398	421

Fig 3.4 The csv file generated containing the coordinates of boxes

4. Working of the Proposed System

The proposed system makes use of the faster R-CNN model instead of the conventional CNN and also made use of the concept of transfer learning. The reason for selecting the faster R-CNN model over CNN, speaking in technical terms, was because in a CNN model, the number of output layers is constant, whereas we do not know beforehand the number of objects that are going to be present in the image, and hence the number of output layers could change.

4.1 Architecture of Faster-RCNN

The input images are represented as Height×Width×Depth tensors (multidimensional arrays), which are passed through a pre-trained CNN up until an intermediate layer, ending up with a convolutional feature map. Using the features that the CNN computed, it is used to find up to a predefined number of regions (bounding boxes), which may contain objects. The variable-length problem is solved in the RPN by using anchors: fixed sized reference bounding boxes which are placed uniformly throughout the original image. Using the features extracted by the CNN and the bounding boxes with relevant objects, we apply Region of Interest (RoI) Pooling and extract those features which would correspond to the relevant objects into a new tensor. Then the model only has to classify the boxes and adjust them so that they fit better.

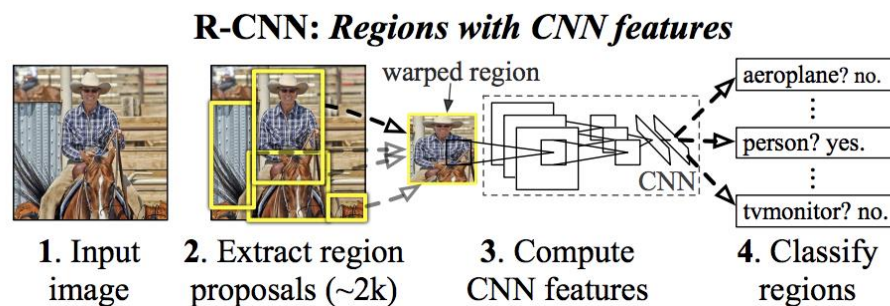


Fig 4.1 A basic architecture of the R-CNN model

4.2 Transfer learning

This is another concept that we used in this project. With the help of transfer learning, instead of training a neural network from scratch, we use the weights of an already trained neural network that solves a problem quite similar to ours. The pre-trained model might not be 100% accurate on our model, but it saves a lot of time and computation power required to train the model from scratch. A pre-trained model however, does require some

fine tuning to better suit the need of our model. Also, since many of the pre-trained models available are trained on very large datasets, they do a good job of generalizing the images.

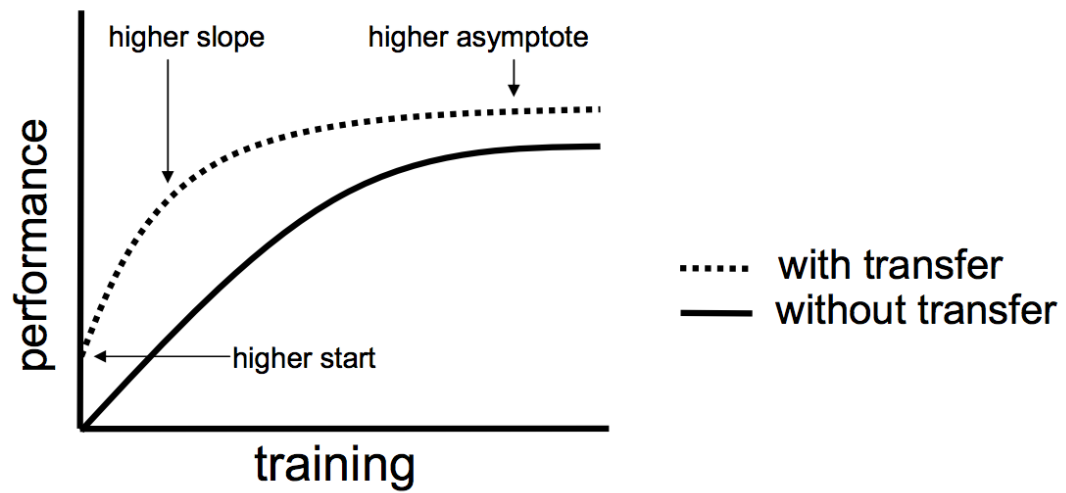
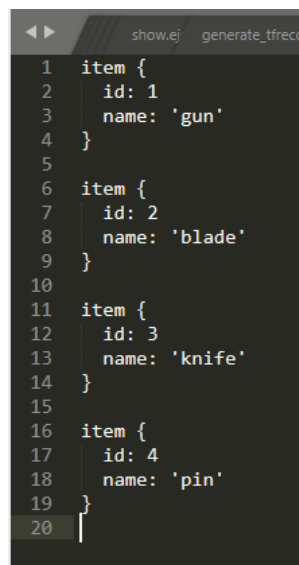


Fig 4.2 A graphical visualization of performance of a model with transfer learning vs without transfer learning

5. Training of the Model

The model being built was different from a conventional CNN model, and required a high computation power as it had to perform all the functions of a CNN while keeping track of the placement of the weapons in the images. So, we decided to use the TensorFlow API, and a pre-trained model. The pre-trained model used was that of a model that detects the kind of playing cards in an image. Since the problem on which the model was trained on was pretty similar to the problem at hand, we decided to implement it.

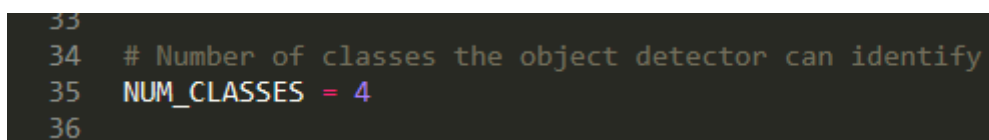
We had to define 4 major classes that we want to detect using our detector, i.e. gun, knife, blade, pin.



```
1 item {
2   id: 1
3   name: 'gun'
4 }
5
6 item {
7   id: 2
8   name: 'blade'
9 }
10
11 item {
12   id: 3
13   name: 'knife'
14 }
15
16 item {
17   id: 4
18   name: 'pin'
19 }
20
```

Fig 5.1 Label map used in our model

Also, the output classes had to be modified according to our model to make accurate predictions.



```
33
34 # Number of classes the object detector can identify
35 NUM_CLASSES = 4
36
```

Fig 5.2 The number of classes

6. Testing of the Model

After roughly 40,000 iterations the total loss of the model was below 0.05 constantly, which was sufficient as any further training could cause the model to overfit. The following code snippet allowed us to test the object detector on a random image picked from the GDXray dataset, mentioned earlier.

```
image = cv2.imread(PATH_TO_IMAGE)
image_expanded = np.expand_dims(image, axis=0)

# Perform the actual detection by running the model with the image as input
(bboxes, scores, classes, num) = sess.run(
    [detection_boxes, detection_scores, detection_classes, num_detections],
    feed_dict={image_tensor: image_expanded})

# Draw the results of the detection

vis_util.visualize_boxes_and_labels_on_image_array(
    image,
    np.squeeze(bboxes),
    np.squeeze(classes).astype(np.int32),
    np.squeeze(scores),
    category_index,
    use_normalized_coordinates=True,
    line_thickness=8,
    min_score_thresh=0.80)
```

Fig 6.1 Code for classifying the object and drawing a box on the output image

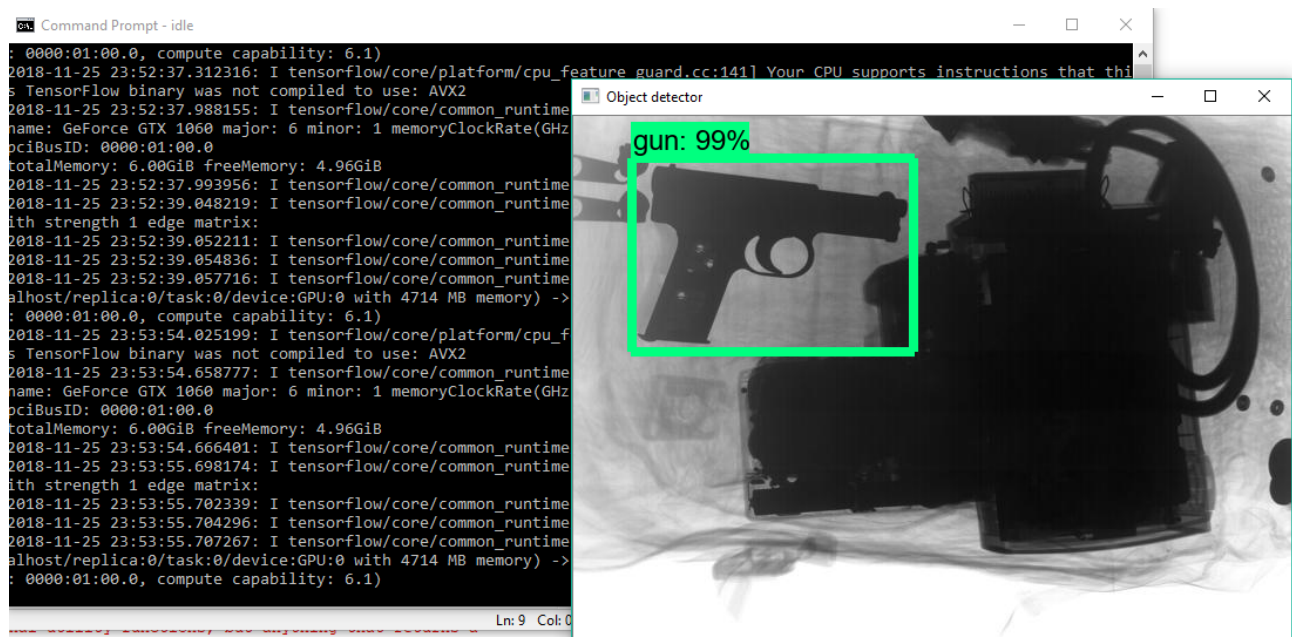


Fig 6.2 Detection of a gun accurately

7. Results and Discussions

The following is the progression of the loss curve of our model in its training phase

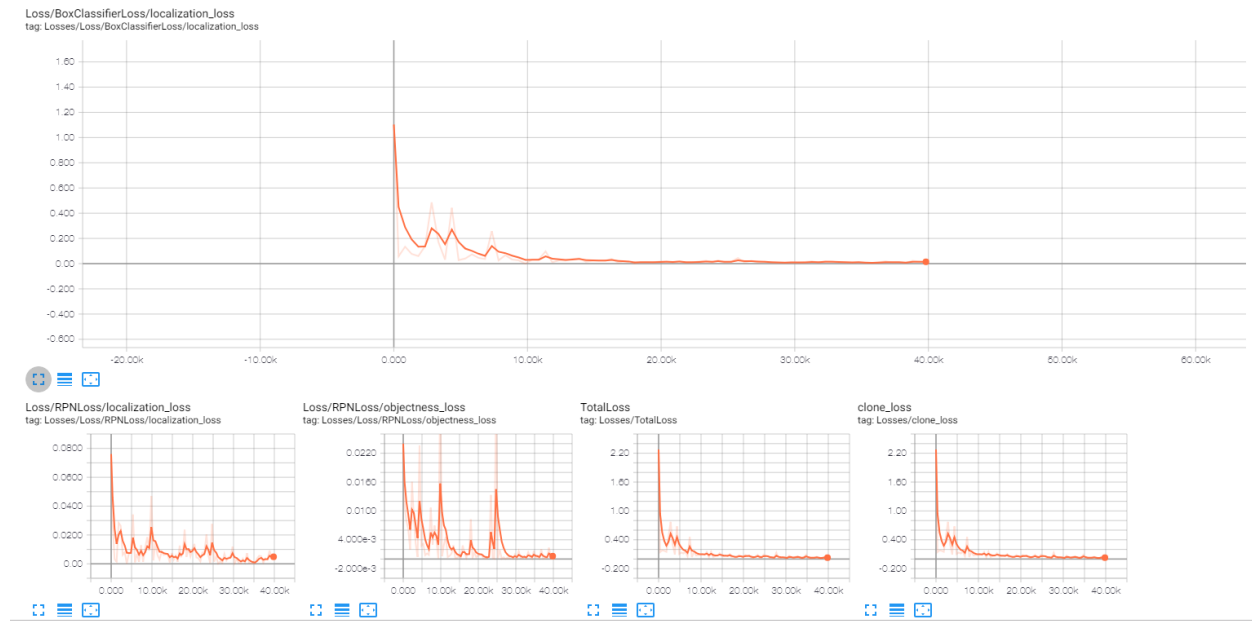


Fig 7.1 Loss graph based on Box classifier metric

As you can see, the loss decreases exponentially and ultimately becomes constant. The metric that we have used to compute the loss factor of our model is the Box classifier loss. This particular loss tells us whether the particular item was correctly classified or not. There are a few other loss metrics too, however they all show more or less the same results.

8. Conclusions and Future Scope

From the above data, we can see that the model can detect most images with a 99% accuracy and the model has a loss of less than 0.05 for this particular set of images.

This particular model was trained on a relatively small set of images, and only on images that were taken from one type of X-Ray scanner, however, if the more images can be added from other types of scanners, the model would be greatly adaptable and versatile in many situations. And not only images, the model could also be modified to detect while scanning in real time. Combined with the right resources and dataset, this model can be of great use on a global level.

9. References

1. <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>
2. GDX-Ray dataset: <http://dmery.ing.puc.cl/index.php/material/gdxray/>
3. <https://machinelearningmastery.com/transfer-learning-for-deep-learning/>
4. <https://www.tensorflow.org/>
5. LabelImg: <https://github.com/tzutalin/labelImg#labelimg>
6. Gyanendra Kumar Verma and Anamika Dhillon. 2018 - A Handheld Gun Detection using Faster R-CNN Deep Learning (2018)