Real-Time Weapons Detections with YOLO and Faster R-CNN

Pietro Basci
Politecnico di Torino
S266004@studenti.polito.it

Gabriele Mariane
Politecnico di Torino
S259162@studenti.polito.it

François Morillon
Politecnico di Torino
S274881@studenti.polito.it

Abstract

This paper is about real-time weapons detection. We want to train and test a Faster R-CNN in order to make it detect weapons in a real-time fashion. Faster R-CCN exploits region proposal algorithms to detect objects inside proposed region. We are interested in finding weapons when they're held by people. To do this we decided to use YOLO to detect people and then to use Faster R-CNN to detect weapons where we have found people. We want to find out if the usage of YOLO increases the speed of our network. Object Detection is a very important field in computer vision since it has many applications, from artificial intelligence devices or machines, like self-driving cars, to security issues, like face-detection, or medical issues. In our case we think it could be helpful for security issues, since fast identification of armed people can be very important on preventing crimes. We make our code publicly available at [13].

1. Introduction

The detection of weapons is an important mater for public safety in crowded spaces such as airports, railway stations or malls. It can contribute in preventing robberies and reducing the number of potential victims in case of terrorist attack. The problem of weapons detection has been addressed in different ways such as metal detector or infrared detection for concealed weapons.

What we are interested in is weapons detection in real-time videos gathered by CCTV with the use of new deep learning techniques. By creating a system able to trigger an alarm as soon as a weapon appears in an image, it is possible to reduce the time required by security agents to act. In addition to surveillance, another possible use of this system is in detecting the presence of weapons in photos and videos uploaded on internet, especially on social networks.

The first model that we tried is Faster R-CNN which is slower but more accurate with respect to other object detection models such as YOLO or SSD. We evaluated it using a wide range of metrics that give us informations about the quality of predictions as well as the inference time. Since the research has security applications, we tried to reduce as much as possible false positives in order to reduce false alarms.

Then, we tried to improve this method by chaining YOLO and Faster R-CNN. Considering that the main problem in this field is the speed of detection we thought that reducing the effort of the Faster R-CNN maybe could have helped on that.

The idea was to reduce the case in which we wanted to search for weapons. So, assuming that weapons inside a frame is always held by humans, we decided to use YOLO to detect the presence of people inside a frame and then we can simply discard frames without people and crop the other frames around every person found and feed the Faster R-CNN with these section to further analyze them.
This decision made it a bit more difficult since it force us to change our Dataset and also to make the two nets work together, which revealed itself to be not very easy in terms of code.

2. Related Work

The problem of weapons detection has been already addressed in different papers.

Justin Lai and Sidney Maples in 'Developing a Real-Time Gun Detection Classifier'[1] from Stanford University developed a real-time gun classifier using Overfeat-3 which allowed to achieve good accuracy (0.89) but it was still too slow for real-time applications (1.3s/image).

Olmos et al. in 'Automatic Handgun Detection Alarm in Videos Using Deep Learning'[4] used for the first time R-CNN for this problem. They made a comparison between the sliding window approach and the region proposal one and demonstrated that the latter performs much better for this problem especially in terms of speed. They achieved 0 False Negative (Recall 100%) but still high False Positive (57) on a test set of 608 images.

Another paper 'Automated Detection of Firearms and Knives in a CCTV Image' proposed by Matiolanski et al. [10] addressed the problem with sliding window approach applied on specific regions of images, i.e. near human silhouettes.

Other works concerns the analysis of infrared and X-Ray images. The analysis of infrared data for weapons detection, by Bandyopadhyay et al. [2], proposes an algorithm to detect weapons combining normal images (RGB) with IR images.

The Damashek & Doherty research about gun detection [3] instead, is about detecting guns in X-Ray Airport images using a parametric edge matching algorithm called Chamfer Matching.

3. Dataset

3.1. Overview

The dataset that we used in this work has been collected by Olmos et al.[7]. It is composed by 3000 images of guns with a good number of images of guns in a context. Every image is associated to an XML file with bounding boxes annotations (Pascal VOC Format). The original idea was to enlarge it with other images of different types of arms as we wanted to make the system able to recognize a large set of weapons from knives to guns and rifles. But due to time constraint, we were unable to do that as it requires too much time to build a good dataset.

For being able later to evaluate our model, we kept some samples for validation and test phases: we randomly split the dataset in three sets: Training Set (2000 images), Validation Set (500) and Test Set (500).

3.2. Data Augmentation

Image captured by cameras can be characterized by different level of brightness, contrast or noise due to the low quality of the sensors. So, in order to improve the variety of dataset image and help the model to generalize we used data augmentation. While training the network, we randomly change the level of brightness, contrast and saturation of the input image and we also randomly horizontally flip the image with a 50% probability. This randomness adds variety to the training data and helps avoid overfit.

4. Methods and algorithms

In object detection we have to deal with speed/accuracy trade-offs: Faster R-CNN is slower but more accurate, YOLO is much faster but not as accurate especially with small objects. The first model that we consider in this work is Faster R-CNN with Resnet-50 FPN backbone that should ensure good performance in terms of accuracy. We start from a Faster R-CNN pretrained on COCO dataset and fine-tune the model on the new dataset. Then, we try to improve this method by using YOLO to detect people inside a frame and once found, we feed the Faster R-CNN with the region found in the previous step to further analyze its content. In this way YOLO works as a sort of 'Potential Risk Areas Generator' which can help to reduce the computation needed at inference time since a frame with no people inside does not requires to be analyzed by the second model and to further reduce possible false positives as the second net acts only on a reduced area.

4.1. Transfer Learning

In order to achieve very good performance, a Deep Convolutional Neural Network needs very large dataset to train good features. Since our dataset is very small, to obtain better result the Transfer Learning technique has been considered. Transfer Learning is a technique which focuses on storing the knowledge gained while solving one problem and applying it to a different but related problem. The idea is to start from a CNN pre-trained on a large dataset and fine-tune the model on the new dataset. In this way, weights learned by training on a large related dataset are used as a starting point for training on the small dataset. So, the pre-trained Faster R-CNN on COCO dataset has been fine-tuned on our dataset.

4.2. Faster R-CNN

Faster R-CNN was proposed by S.Ren et al.[5] and it is one of the state-of-the-art object detection models. It is composed by two modules: the first is a deep fully convolutional network that proposes regions, and the second module is the Fast R-CNN detector [8] that uses the proposed regions. The idea is to exploit the Region Proposal Network (RPN) to predict proposals directly from the feature map. In Faster R-CNN model, convolutional layers are shared among the two modules. The architecture of the Faster R-CNN is shown in Figure 1.

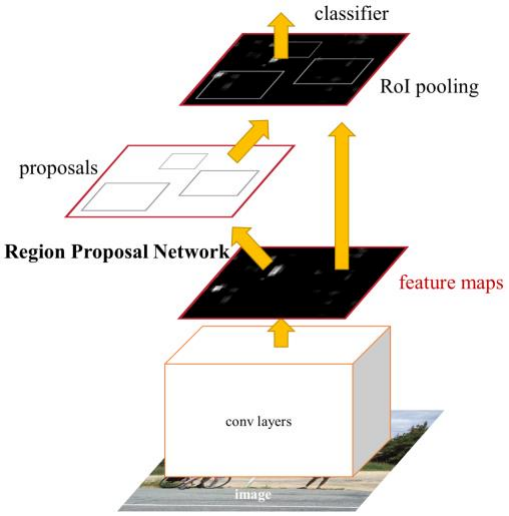

Figure 1: Faster R-CNN architecture [1]

4.2.1 Loss function

The training phase in a Faster R-CNN proceed trying to jointly minimize 4 losses:

1. RPN Objectness;
2. RPN Regression box;
3. Final Classifier loss;
4. Final Regression box.

So, we use a multi-task loss, given by the sum of the previous 4 losses, to jointly train for classification and bounding-box regression.

4.2.2 Feature Pyramid Network (FPN)

Feature Pyramids are a basic component in recognition systems for detecting objects at different scales. But recent detection model have opted to use only single scale features for faster detection (simple CNN). An alternative is to reuse the pyramidal feature hierarchy computed by a ConvNet. The Feature Pyramid Network proposed by T. Lin et al.[11], exploits the inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids with marginal extra cost. They developed a topdown architecture with lateral connections for building high-level semantic feature maps at all scales. This solution shows significant improvement as a generic feature extractor in several applications. Figure 2 shows the differences between these approaches.

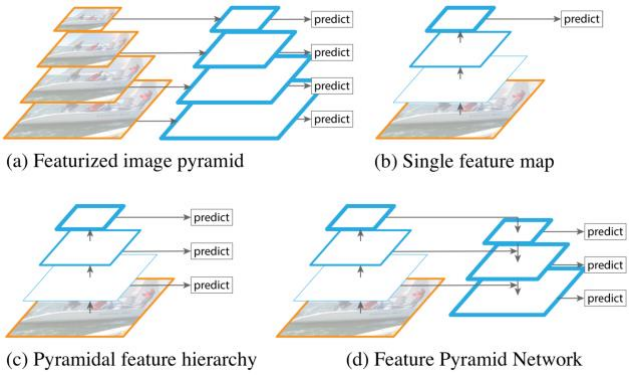

Figure 2: Pyramid Features[11]

Differently from the standard Faster R-CNN which works on a single feature map extracted by convolutional layers, with the introduction of FPN we generate a pyramid of feature maps and based on the size of the ROI (generated by RPN) the feature map layer in the most proper scale to extract the feature patches is selected. We adopted this technique in our system since it can help a lot in finding small objects such as weapons (RoI with small scale are mapped into a fine-resolution level).

4.3. Evaluation Metrics

Object detection is an example of multi-task learning. Its objectives are indeed two: localize an object by predicting the coordinates of the bounding boxes around it and classify the object by predicting class to which it belongs. Therefore, in order to properly evaluate the performance of an object detection model we have to consider different metrics.

The first important concept in object detection model evaluation is Intersection over Union (IoU) also known as Jaccard Index. It is a metric which evaluate the quality of a box prediction. Specifically, it computes the common area of two bounding boxes (intersection) divided by the union of the areas of the two boxes.

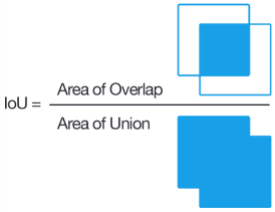

Figure 3: Intersection over Union

Once defined this concept, we set a threshold value for the IoU to determine if a detection of an object has to be considered valid or not. After few test, we set the IoU_threshold to 0,5 so that: if IoU is >= 0,5 the detection is considered as True Positive; if IoU is < 0,5 it is considered as False Positive; instead, when an object is really present in an image but the model fails to find it, it is considered as a False Negative. A True Negative, instead, is every part of the image where there are no objects and the model do predict nothing. The latter is not meaningful in object detection so we ignore it. Other metrics considered are Precision, Recall and F1-Score defined as:

Precision = TP / (TP + FP)

Recall = TP / (TP + FN)

F1-Fcore = (2*Precision*Recall) / (Precision + Recall)

We consider also the Average Precision (AP) and Average Recall (AR) which are very popular metrics in object detection. We used the coco api[14] to compute these metrics. For COCO, AP is the average over multiple IoU (the minimum IoU to consider a positive match). For instance, an AP@[0.5:0.95] corresponds to the average AP

(traditionally known as mAP) for IoU from 0.5 to 0.95 with a step size of 0.05.

Finally we evaluate the inference time needed by the model expressed in s/frame and so the relative fps (frame per second).

4.4. Results

Figure 4 shows the losses and the accuracies trend during the trining phase. The results achieved by this model are reported below.

4.4.1 Statistics

	Training Set	*Validation Set*	*Test Set*
True Positive	1101	493	474
False Positive	19	25	30
False Negative	36	81	98
Precision	98.3	95.2	94.0
Recall	96.8	85.2	82.9
F1-Score	97.6	90.3	88.1
Accuracy	95.2	82.3	78.7

Table 1: Performance metrics

Metrics related to Average Precision and Average Recall are shown in Table 2.

Faster R-CNN		*Validation Set*	*Test Set*
AP@[0.5:0.95]	area=all	0.650	0.654
AP@[0.50]	area=all	0.936	0.923
AP@[0.75]	area=all	0.682	0.709
AP@[0.5:0.95]	area=small	0.289	0.300
AP@[0.5:0.95]	area=medium	0.402	0.436
AP@[0.5:0.95]	area=large	0.713	0.726
AR@[0.5:0.95]	area=all	0.600	0.604
AR@[0.5:0.95]	area=all	0.697	0.710
AR@[0.5:0.95]	area=all	0.701	0.712
AR@[0.5:0.95]	area=small	0.360	0.423
AR@[0.5:0.95]	area=medium	0.521	0.560
AR@[0.5:0.95]	area=large	0.755	0.773

Table 2: COCO api metrics

Inference time 0.11 s/image which corresponds to about 9 fps on a K80 GPU.

Thanks to the use of an high confidence threshold (0.9), we obtained a number of False Positive (25 and 30 on validation and test set) which is much lower then the one obtained by others research (57 in [4]). But, on the other hand, so high value for confidence threshold lead to an increase of False Negative, which is still too large.

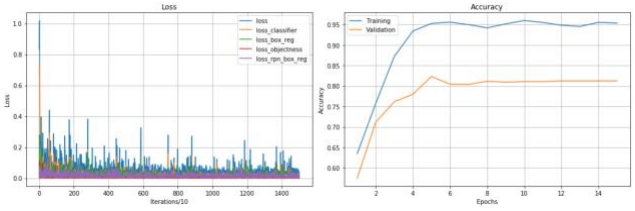

Figure 4: Losses and Accuracies trend

Figure 5: Output frames with bounding boxes

4.4.2 Visualization

Figure 4 shows the output of the Faster R-CNN on a CCTV frame which is the localization of the weapon and classification as a gun. In order to solve problem of multiple overlapped predicted boxes (Figure 5) we adopt the technique of Non-maximum Suppression (NMS) which removes lower scoring boxes which have an IoU greater than iou_threshold with another (higher scoring) box.

5. Improved Model

In this section we describe the method to improve the previous system. As introduce above, the idea behind is trying to reduce the number of frames that have to be analyzed by Faster R-CNN. This idea is based on the assumption that weapons inside a frame is always held by humans so we can reduce the number of frames that Faster R-CNN have to detect by separating frames with people inside from frames with no one in advance.

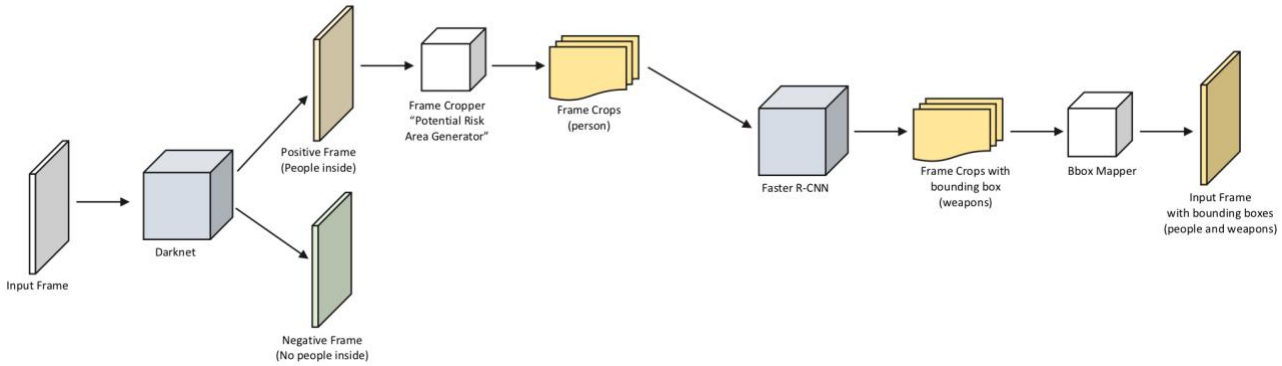

Figure 6: Final architecture

In order to do that, we chained two different nets: YOLO which is faster but less accurate especially on small objects and Faster R-CNN which is slower but more accurate.
The objective of the first net is to to detect the presence of people inside a frame. If no people are inside the frame, it is simply discarded, otherwise a crop of the frame is generated around every person found and are used to feed the Faster R-CNN to further analyze its content.

In this way YOLO works as a sort of 'Potential Risk Areas Generator' which can help to reduce the computation needed at inference time since a frame with no people inside does not requires to be analyzed by the second model and to further reduce possible false positives as the second net acts only on a reduced area. The resulting structure of the whole system is shown in Figure 6.

5.1. YOLO

YOLO uses a different approach from the other object detection Nets. It consists on the application of a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.
This model looks at the whole image at test time so its predictions are informed by global context in the image. It also makes predictions with a single network evaluation. This makes it extremely fast.

5.2. Chaining the Nets

We integrated in our system the YOLOv3[9] which is implemented using Darknet available on the GitHub repository[12]. For our usage, we need YOLO only to detect people, so we reused the net pretrained on COCO dataset which already includes the 'person' class.

So, every image is scanned one time by YOLO, which, as we said before, predicts bounding boxes for every person (if any), and it do it very fast. Then we check if YOLO found some objects in the image, and, if so, we check if some of these objects has been classified as 'person'.

If YOLO doesn't find any person in the image it is discarded, otherwise, for each person found, we give as input to Faster R-CNN a cropped version of the image, which contains only the box detected by YOLO, slightly enlarged.
The expected improvements of this algorithm consist in the fact that Faster R-CNN, which is much slower than YOLO, contributes to the total time only when YOLO finds a person, while, for the rest of time, the Net proceed at YOLO fps.
We estimated the impact on the detection time as:

Detection time = yolo_inference_time + #people*faster_r_cnn_inference_time

5.3. Results

Figure 5 shows the output of the system on an image with a person holding a gun.

Figure 7: Output frame with person and weapon bounding boxes

Unfortunately we were unable to test the model because the dataset considered does not respect our assumption. Indeed, it contains a lot of images of guns without people which significantly damage the performances of our model. To properly test it, we need a test set composed by images of people holding guns, otherwise all images that contains

only guns, according to the system functioning, are wrongly classified as negative. However, since we reused the same Faster R-CNN, we estimate that performance in terms of accuracy should be comparable or even better then the previous system, as Faster R-CNN acts on a subsection of the frame, while we expect an improvement in terms of average frames per second classified.

6. Conclusion

In this work we addressed the problem of the weapons detection. The first model that we tried reached an accuracy lower with respect to the current state of the art but conversely we get a number of False Positive which is about the an half of the results of other papers. And this aspect is positive since it dramatically reduce the number of false alarms. Then we introduced a method to try to improve the speed of the first model but due to time constraint we were unable to properly test it, as it requires us to improve the dataset with new image with people holding guns.

7. Future Work

As future work, we think that it is possible to further improve the performance of the first model by doing hyper-parameter tuning. Then, other images should be gathered and used to evaluate the performance of the second method that we introduced.

References

[1] Justin Lai, Sydney Maples, *Developing a Real-Time Gun Detection Classifier*, project for Computer Vision Course: CS231N, Stanford University, 2017.

[2] Samir K. Bandyopadhyay, Biswajita Datta, and Sudipta Roy *Identifications of concealed weapon in a Human Body* Department of Computer Science and Engineer, University of Calcutta, 2012.

[3] Aaron Damashek and John Doherty *Detecting guns using parametric edge matching* Project for Computer Vision Course: CS231A, Stanford University, 2015.

[4] Roberto Olmos, Siham Tabik, and Francisco Herrera *Automatic Handgun Detection Alarm in Videos Using Deep Learning* Soft Computing and Intelligent Information Systems research group, Department of Computer Science and Artificial Intelligence, University of Granada, 2017.

[5] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*, arXiv:1506.01497, Cornell University, Computer Vision and Pattern Recognition Section.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, arXiv:1506.02640, Cornell University, Computer Vision and Pattern Recognition Section.

[7] https://sci2s.ugr.es/weapons-detection

[8] R. Girshick, "Fast R-CNN," in IEEE International Conference on Computer Vision (ICCV), 2015

[9] Joseph Redmon, Ali Farhadi, *YOLOv3: An Incremental Improvement*, arXiv, 2018.

[10] A. Matiolanski, M. Grega,P. Guzik and M. Leszczuk "Automated Detection of Firearms and Knives in a CCTV Image", AGH University of Science and Technology, Poland, 2016.

[11] T.Lin, P. Dollar, R. Girshick, K. He, B. Hariharan and S. Belongie in "Feature Pyramid Networks for Object Detection", Facebook AI Research, 2017.

[12] https://github.com/pjreddie/darknet

[13] https://github.com/SadCl0wn/mirto

[14] https://github.com/cocodataset/cocoapi