Weapon Detection

Bryan Rivas

Bryan Rivas, New York, 10031, USA

Abstract:

Gun violence and other crimes that involve weapons like knives have always been an issue of concern for public safety. This work focuses on implementing different object detections techniques and apply them to weapon detection so that we can quantify how effective they are. Weapon detection systems can be developed to be used in conjunction with surveillance systems. It is crucial to study the existing ways in which we could implement such systems so that we can prevent potential weapon-related crimes. There have been multiple approaches to tackling this problem, but the biggest challenge seems to be the speed at which the weapons are detected. Furthermore, when a weapon detection system is integrated for real-time surveillance, these systems need to respond quickly and accurately. Thus, we compare a few of the known approaches that have been research and see how they perform.

Introduction

All crimes, especially weapon-related ones can be an increasing source of stress, even more so for those people that are highly concerned with their safety. This safety issue spans globally and locally. For instance, the concern is further heightened in countries where the possession of guns is legal or was legal for a period [1]. The statistics reported by the United Nations Office on

Drugs and Crime (UNODC) in the year 2017, reveal that the number of crimes involving guns per 100,000 inhabitants is very high in many countries [1]. In addition, studies show that there is a probability of a gun-related crime or violent behavior that drastically increases just for the simple fact of having access to a gun [1].

According to The New York City Police Department, all crimes in the city have increased by 38.5% from January 2021 when compared to January 2022. The website also lists two different crime categories, these are **Transit Crimes** and **Citywide Shootings.** "Transit Crimes" have increased by 75.2%, and "Citywide Shootings" have increased by 31.6% when comparing the data from January 2021 to January 2022 [2]. A big portion of these crimes involve guns and many of these crimes take place in public spaces that are used by the citizens of New York City daily.

Related Work

In the related work that has been done on object detection involving weapons, one of the proposed models was based on the VGG-16 model which is a deep learning model consisting of a 16 layers convolutional neural network. However, the proposed model was using 25 layers and 337,671 parameters [3]. This model was able to successfully detect the weapon with an accuracy of 98.40% as opposed to the original VGG-16 model which only had an accuracy of 89.75%.

Another approach that was taken towards weapon detection was one using "Faster R-CNN Deep Learning" [4]. This approach was implemented using the convolutional neural network "MatConvNet" which is a MATLAB toolbox that implements state if the art CNNs without a Graphical Processing Unit (GPU). The main idea behind this approach was to iteratively minimize the average loss function of the model. In addition to this, they

implemented a 2-process step for classifications: (1) indirect regularization, inflicted by smaller convolutional filter sizes and greater depth , (2) use pre-initialization for definite layers [2]. Furthermore, the input size of the CovNets was a fixed 224 x 244 RGB image during training and to improve the speed for training, the researchers introduced parallelization to the mini batch gradient descent process. This had to be done because the model is very deep and training on a single GPU could take months to finish. In the end this model reached an accuracy of 98%.

The last model we looked at was the “You Only Look Once” model or YOLO for short and more specifically, YOLO V3 [5]. The yolo model uses a custom convolutional neural network that includes residual blocks, skip connections and upsampling (See figure 1 below) which makes it easier for the model to detect small objects. Yolo managed to achieve an accuracy of 98.89% and what is more surprising than that is that it is incredibly fast, the YOLO V3 model can detect objects in videos that are running at 45 frames per second.

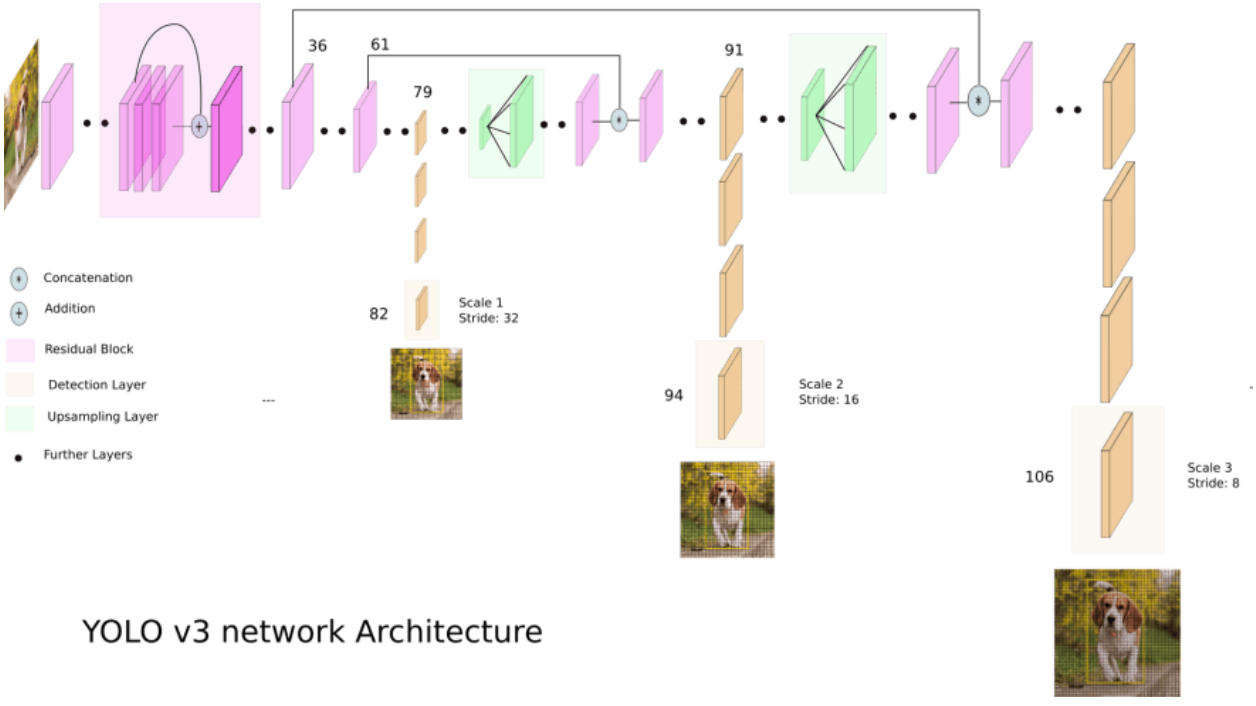

Figure 1. Simplified architecture of the YOLO V3 model.

Exploratory Data Analysis

The dataset that we chose is a subset of weapon data from the Andalusian Research Institute in Data science and Computational Intelligence. The dataset consists of 9,000 images of handguns, knives, and other objects such as wallets and cards (See figure 2 below) that was cut down to 4,000 images.

Figure 2. Example of our dataset

The source of the images is not specified in the dataset, but from taking a quick look at the images, most if not all of them seem to not have been taken from a satellite. For this reason, we decided to not check for infra-red contamination by performing the naïve red band subtraction.

We also decided to check the distribution of our data and saw that we had some class imbalance (See figure 3 below). However, in our case it would not affect us much since we have more objects of the classes that we wish to successfully detect. In addition, we were limited in the amount of data that we could use for training since we did not have the required computational power, more details in the methodology section.

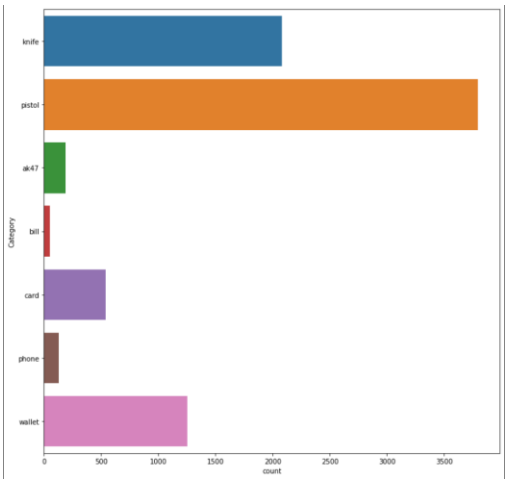

Figure 3. Class distribution of our dataset.

Methodology

Object detection is a two-step process in which we do following: (1) classification, which is using a classifier to predict whether the image contains one or more of the objects that we wish to detect and (2) detection, which is locating in which region of the image this object is located and drawing a bounding box around it at the exact location that the object is situated on the image.

To perform the first step of the two-step process described above we are going to need a good classifier, the two classifiers that we are going to implement are a neural network and a convolutional neural network.

Architecture of the Neural Network

Our neural network architecture consists of an input layer of size 150 x 150 that takes gray-scale images as input, two hidden layers of 512 neurons each and an output layer consisting of 4 neurons. The activation function of our two hidden layers is the rectified linear unit function or

“ReLU” and the activation function of our output layer is the “softmax” function since we are interested in getting probabilities of our classes as output. The model was compiled using the “rmsprop” optimizer and categorical cross entropy for the loss. We split our dataset into 90% training and 10% for testing and trained our model for 50 epochs but did not achieve satisfactory results. We obtained an accuracy of 65% on the validation dataset (See figure 4) and training the model for longer than 50 epochs did not improve performance any further.

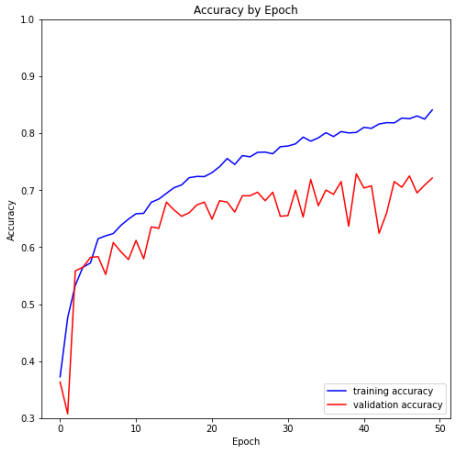

Figure 4. Accuracy by epoch of our neural network model.

Architecture of the Convolutional Neural Network

Our convolutional neural network architecture consists of an input layer that takes images of size 150 x 150 x 3 or RGB images of size 150 x150. Then we have a convolutional layer using the Conv2D net to generate 64 feature maps in which a kernel of size 3 x 3 will be applied and the kernel initializer is set to “he_uniform” so that the samples are drawn from a uniform distribution and the padding is kept the same in the rest of the layers. Then we apply a max pooling layer with a pool size of 2 x 2 and strides of size 3 x 3. Furthermore, we apply another convolutional layer followed by another max pooling layer using the same parameters we specified before. In addition, we apply two more convolutional layers, but this time we use feature maps of size 128

and every other parameter is kept the same as before. Moreover, we apply another max pooling layer with the same parameters as the other pooling layers, then we flatten the output of that last max pooling layer and add 4 fully connected layers one after the other of sizes 128, 64, and 32 respectively with a “ReLu” activation function. Lastly, we add a dropout layer and dense layer with 4 neurons as our output layer with the “softmax” activation function so that we can get the probabilities of each class as an output. This model was compiled using the Adam optimizer with a learning rate of $1*10^{-4}$ and categorical cross entropy for the loss. The model was trained on the same data as our neural network model except that this time our image data was not gray scale, the full RGB range was preserved. The performance of this model was better than our neural network model at 83% accuracy on validation data (See figure 5 below), but still not a desirable result.

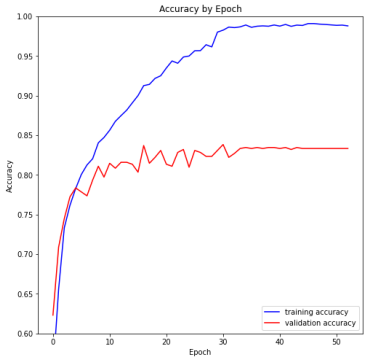

Figure 5. Accuracy by epoch in our convolutional neural network model

Improving the accuracy of our convolutional neural network

We decided to try to improve the performance of our convolutional neural network as our classifier since it had performed much better than our neural network. To do this, we used the xml files that came with our data set because they contained the (x, y) locations of the object of interest in our images as well as the width and height of said object. We parsed those xml and

extracted the region of interest (ROI) of each object for each of the images in our dataset and we then used these ROIs to create a new dataset from the original one which only contained the objects of interest (See figure 6 below). The rationale behind this was that since CNNs are really efficient at detecting characteristics in images in order to classify them, if we trained our model only on the objects that we are interested on, the CNN would have a much easier time classifying them.

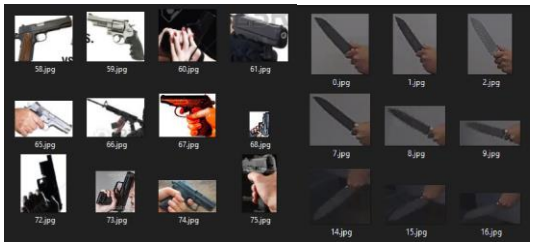

Figure 6. New dataset comprised only of the ROIs in our images.

We also decided to have a dataset of images that did not contained the object of interest, we obtained these datasets by cropping our images in sizes of 128 x 128 and ignoring the ROIs which we know from the xml files as to not capture the object of interest (See figure 7 below). We did this so that our model could be trained in what the negative class would look like so that it would be less likely to classify something that is not a weapon as a weapon.

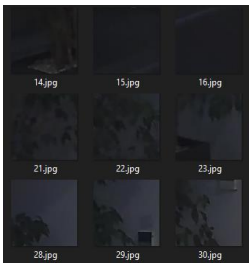

Figure 7. New dataset without the objects in the ROIs.

After training our model in this new dataset we obtained a performance of 92% accuracy in our validation set (See figure 8 below). Now that we have a good enough classifier, we are going to move to the "detection" part of the problem, but first we are going to explain three concepts that are essential to understand how detection works.

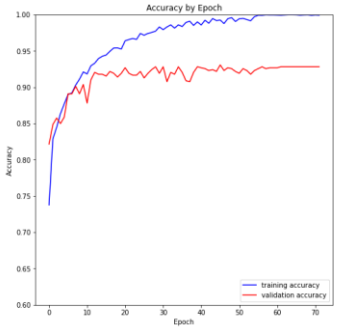

Figure 8. Accuracy by epoch of our CNN after training with the new dataset.

The selective image search algorithm

The selective image search algorithm is a process in which multiple areas of an image are chosen at random and each chosen area is modeled as a histogram of binomial distribution of foreground and background. The algorithm considers color, texture, fill and size and generates a histogram for each category. Then, the algorithm computes what is referred to as "the similarity" of a region. This computation is done by taking the minimum of the color at a "position" in the generated region, the minimum of the texture at the same "position", the size of the region at that "position" as 1 – the size at that "position" divided by the total size of the image and the fill of the region at that "position" as 1 – the size of the bounding box on that region (the size of the bounding box is the size of the chosen area by the algorithm) and then we subtract the difference in size at that "position" divided by the total size of the image. Finally, the algorithm adds up all of the previously computed values to assign it a similarity score and regions that have very

closed similarity score are combined together to form bigger regions. An intuitive approach of this algorithm can be seen in the image below [6] (See figure 9 below).

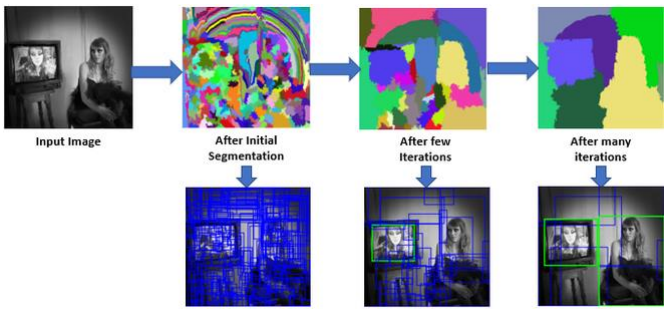

Figure 9. Intuitive selective image search illustration.

The Intersection Over the Union

The intersection over the union is a way to compute how much a bounding box is overlapping with another bounding box. The way in which this computation is performed is by calculating the area of the intersection and dividing this area by the area of the union minus the area of the intersection. This computation will give us a value from 0 to 1 that tells us how much two bounding boxes are overlapping with each other [7].

The Non-Maximum Suppression Algorithm

The non-maximum suppression algorithm is a way to reduce overlapping bounding boxes. The idea behind this algorithm is that if a box is overlapping with another box above a certain threshold, then these two bounding are drawn around the same object. How much a box is overlapping with another is obtained by computing the “Intersection Over the Union”. The non-maximum suppression algorithm also takes into account the probability scores so that at the end it can keep the box or boxes with the highest probabilities and the lowest overlap, and these boxes would be the ones that we would use to draw around a “detected object” [9].

Object Detection

Now that we have a good classifier and talked about the important algorithms that are needed to perform object detection, we are going to talk about the specific implementation details for each of the object detections approaches that we took

1. **CNN:**

   To detect an object using a CNN we grab a random image for which we wish to detect an object. Then, we perform the selective image search algorithm in said image so that we can get a list of proposed regions. These regions contain the (x, y) coordinates of the region as well as the width and the height. We extract that region from the original image and run it through our classifier, if the region is classified as part of the positive class by our classifier, we store the (x, y) coordinates as well as the width, height, and probability score of said classification, the stored data would be considered as a "box". Once we have done this for all the regions returned by the selective search algorithm, we will use the non-maximum suppression algorithm to eliminate bounding boxes that are overlapping by more than 50% as these boxes are most likely being drawn around the same object. Finally, we used the returned boxes by the non-maximum suppression algorithm and draw them around the original image and this would give us our detection. The results of the CNN detection can be found in "Appendix A".

2. **R-CNN:**

   The R-CNN implementation of object detection is identical to the CNN implementation. However, our training dataset is selected from regions that are proposed by the selective image search algorithm [8]. Since we know the ROIs for each of our images from the xml annotations that came in our dataset, what we do is compute the "Intersection Over the

Union" of a proposed region and our ROI and if the overlap is greater than 70% this region is a good region for us to use in training our model. Since the training images are also coming from the selective image search algorithm this will help increasing the performance of our classifier. The accuracy we obtained after training our CNN model in this manner was 97% on the validation dataset (See Appendix B). Object detection happens in the same way as describe before, for the results (See appendix A). One downside of the RCNN is that it takes an incredibly long time to train (See Appendix B).

3. **YOLO V3 model:**

   The YOLO V3 model is the best model for object detection today, as stated in the "Related Work" section, YOLO uses its own implementation of a convolutional neural network that includes residual blocks, skip connections and upsampling blocks which makes it easier for the model to detect small objects. The most impressive thing about YOLO is that it can do classification and detection in a single pass and it is extremely fast, the YOLO V3 model can detect objects in videos that are running at 45 frames per second. One downside of YOLO is that it takes very long to train, I trained our model for one week and three days and only obtained an accuracy of 25%. However, research shows that YOLO can reach an accuracy close to 99% in weapon detection [3]. The YOLO model can be used in libraries such as TensorFlow and OpenCV, the model will return as output a list of bounding boxes and confidence scores for which we will have to use the "Non-Maximum Suppression Algorithm" to select the best ones and then draw them in our original image. Before using the non-maximum suppression algorithm, we must make sure to turn the coordinates returned by the yolo model into the standard coordinate format, this is because YOLO uses a different coordinate system to locate the

ground truth objects in the images during training. To do this we have to multiply the first value returned by our model by the width and the second value by the height of our original image to get the center of our (x, y) coordinates, then we multiply the third value by the width and the fourth value by the height of our original image in order to get the width and height of the bounding box that we are going to draw. Lastly, we subtract the center of our x coordinate by the width of the bounding box and divide it by 2 and the center of our y coordinate by the height and divide it by 2 to obtain the (x, y) coordinates where we wish to draw our bounding box (See figure 10 below). The results of the object detection for our YOLO model can be seen in "Appendix A".

```
center_x = int(detection[0] * Width)
center_y = int(detection[1] * Height)
w = int(detection[2] * Width)
h = int(detection[3] * Height)
x = center_x - w / 2
y = center_y - h / 2
```

Figure 10. Turning YOLO output into standard coordinates.

Conclusion

In conclusion, object detection is a very complicated problem with a lot of moving parts, the YOLO model has managed to accurately and efficiently tackle all of this problems and has taken object detection into a state where weapon detection is within reach. So, as future data scientist and machine learning engineers one of the best things we can do to help further this goal along is to keep collecting good quality data so that we can train the best YOLO model possible.

References

1. Roberto Olmosa, Siham Tabika, Francisco Herrera, et al. “Automatic Handgun Detection Alarm in Videos Using Deep Learning.” *Neurocomputing*, Elsevier, 15 May 2017, https://reader.elsevier.com/reader/sd/pii/S0925231217308196?token=19BEC60C0B9B362BFDC21EDAEC22523B6C2C26789EB9C35192D5BD793AB543A1921E3BA9DAA95CA28D8AB319FC3D92C3&originRegion=us-east-1&originCreation=20220218181919.
2. “NYPD announces citywide crime statistics for January 2022.” *The official website of the City of New York*, 3 February 2022, https://www1.nyc.gov/site/nypd/news/p00036/nypd-citywide-crime-statistics-january-2022.
3. Kaya V, Tuncer S, Baran A. Detection and Classification of Different Weapon Types Using Deep Learning. Applied Sciences. 2021; 11(16):7535. https://doi.org/10.3390/app11167535
4. Verma, Gyanendra K, and Anamika Dhillon. *A Handheld Gun Detection Using Faster R-CNN Deep Learning*. https://www.researchgate.net/profile/Gyanendra-Verma/publication/322133989_A_Handheld_Gun_Detection_using_Faster_R-CNN_Deep_Learning/links/5b0f8719aca2725783f41ead/A-Handheld-Gun-Detection-using-Faster-R-CNN-Deep-Learning.pdf.
5. Narejo, Sanam, et al. “Weapon Detection Using Yolo V3 for Smart Surveillance System.” *Mathematical Problems in Engineering*, Hindawi, 12 May 2021, https://www.hindawi.com/journals/mpe/2021/9975700/.
6. “OpenCV Selective Search for Object Detection.” *PyImageSearch*, 17 Apr. 2021, https://pyimagesearch.com/2020/06/29/opencv-selective-search-for-object-detection/.
7. “Intersection over Union (IOU) for Object Detection.” *PyImageSearch*, 30 Apr. 2022, https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.
8. “R-CNN Object Detection with Keras, Tensorflow, and Deep Learning.” *PyImageSearch*, 17 Apr. 2021, https://pyimagesearch.com/2020/07/13/r-cnn-object-detection-with-keras-tensorflow-and-deep-learning/.
9. Redmon, Joseph. *Yolo: Real-Time Object Detection*, https://pjreddie.com/darknet/yolo/.
10. “Yolo Object Detection with Opencv.” *PyImageSearch*, 13 May 2022, https://pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/.

Appendix A

Figures of the object detections results

This appendix contains all the figures and captions for the object detection results of our different models.

Figure 11. Results of the CNN model object detection

Figure 12. Results of the CNN model object detection.

Figure 13. Results of the YOLO V3 model object detection

Appendix B

Figures of the R-CNN performance and training time

This appendix contains the figures detailing the performance of our R-CNN classifier on the validation data as well as the time taken to load the data.

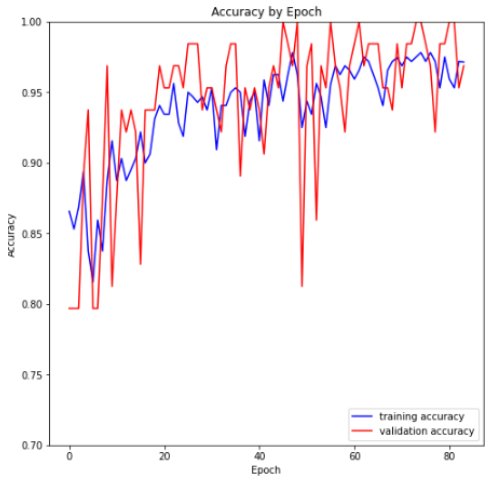

Figure 14. Accuracy by epoch of our R-CNN model in the training and validation data

```
3459it [8:54:24, 17.02s/it]
working on file:  armas (2999).jpg
3460it [8:54:29, 13.61s/it]
working on file:  armas (3000).jpg
3461it [8:54:32, 10.41s/it]
working on file:  armas (3000).jpg
3462it [8:54:34,  7.95s/it]
working on file:  armas (3000).jpg
3463it [8:54:37,  6.20s/it]
working on file:  armas (3000).jpg
3464it [8:54:39,  9.26s/it]
```

Figure 15. Time taken to load all the training data for our R-CNN. 8 hours, 54 minutes and 39 seconds to load the entire dataset.