Gun Detection in Real Time Videos Using Convolutional Neural Networks

Rohini Kapoor
University of Massachusetts
Amherst
rohinikapoor@cs.umass.edu

Rahul Handa
University of Massachusetts
Amherst
rhanda@cs.umass.edu

Abstract

Like many other computer vision problems, there still isnt an obvious or even best way to approach Object Detection, it varies for each application domain. Given the current scenario regarding gun violence around us, we try to implement an approach which helps us by detecting guns being held by people of interest within a video. Real-time object detection is a promising application of Convolutional Neural Networks (CNNs). Some of the previous work has been focused on weapon-based detection within infrared data for concealed weapons. For our project, we first describe a custom classification network created for detecting guns and finally we use a Tensorflow-based implementation of the SSD[3] network as an integrated network for detecting and classifying guns.

1. Introduction

This has become an American routine: After every mass shooting, the debate over guns and gun violence starts up once again. Maybe some bills get introduced. Critics respond with concerns that the government is trying to take away their guns. The debate stalls. So even as America continues experiencing levels of gun violence unrivaled in the rest of the developed world, nothing happens no laws are passed by Congress, nothing significant is done to try to prevent the next horror. One way to reduce this kind of violence is prevention via early detection so that the security agents or policemen can act. We started thinking about this problem from a technological perspective. In particular, one innovative solution for this issue can be to equip surveillance or control cameras with an accurate automatic handgun detection system. In the last few years, deep learning in general and Convolutional Neural Networks (CNNs) in particular have achieved superior results to all the classical machine learning methods in image classification, detection and segmentation in several applications. Instead of manually selecting features, deep learning CNNs automatically discover increasingly higher level features from data [4]. We aim at developing an armed person detector in videos using CNNs. We start with a straight forward approach of classification using a deep network and applying it at every frame of the videos. Once we have this baseline setup completed, we then move on to object detection based methods.

Object detection is not as trivial as classification, we have to solve two problems at the same time: localization and classification. First, we used a region proposal network to generate regions of interest and then we use either fully-connected layers or position-sensitive convolutional layers to classify those regions. It sounds pretty intuitive but there has been a lot of research that went into reaching these two steps. Some of the models that deal with this are R-CNN[1], Faster R-CNN[2] and SSD.
Using CNNs to automatically detect pistols in videos faces several challenges:

- Pistols can be handled with one or two hands in different ways and thus a large part of the pistol can be occluded.
- The process of designing a new dataset is manual and time consuming.

2. Related Work

2.1. Gun Detection

The first and traditional sub-area in gun detection focuses on detecting concealed handguns in X-ray or millimetric wave images. The most representative application in this context is luggage control in airports. The existent methods achieve high accuracy by using different combinations of feature extractors and detectors, either using simple density descriptors [5], border detection and pattern matching [7] or using more complex methods such as cascade classifiers with boosting [8]. The effectiveness of these methods made them essential in some specific places. However, they have several limitations. As these systems are based on metal detection, they cannot detect non metallic guns. They are also not precise because they react to

all metallic objects. They all do material detection and need heavy equipment(which are expensive) combined with some form of feature detection.

2.2. Object Detection

Classic object detection with the sliding window apprach, which considered multiple varied size windows, and ran a classifier on each window was a very slow process and was not used for real time detection. However, Region Proposal methods, which selects actual candidate regions using detection proposal methods speed up this process substantially. The first detection model that introduced CNNs under this approach was Region-based CNNs (R-CNN). It first scanned the input image for possible objects using an algorithm called Selective Search, generating 2000 region proposals. It then ran a CNN on each region proposals to extract features, and then fed these to an SVM to classify each box, and also a linear regressor to tighten the bounding box of the object, if such an object exists. R-CNN was a big improvement on the sliding window approach.
Fast R-CNN was similar to R-CNN, but improved the speed by running only one CNN on the entire image first, and then applying selective search method on the features. Faster R-CNN further improved the speed by replacing the selective search algorithm by region proposal network (RPN). It then first generated regions of interest using RPN, and then classified those regions.
Single Shot detection model (SSD) made a further improvement in speed. It does both region proposal and classification in a single step. SSD uses non-maximum suppression to group together highly-overlapping boxes into a single box. It also uses hard negative mining to balance classes during training. As our aim was to do object detection in real time, we use the SSD model for this project. However, what we gain in speed, we also compromise in accuracy.

3. Dataset

3.1. For classification

To create our dataset for the classification task, we scrapped around 2500 images from the internet, of people holding guns, and of just guns. We downloaded images of all types- different orientations of the gun, some images are a little blurred, the people are holding guns at different angles and some are even holding multiple guns in their hands.

Out of these, we used 2455 images for training and 119 for testing. We labelled one class as triggered(people with guns) and the other as non-triggered(just guns). We had 1186 images of the first class and 1269 images for the second class. To run it through our network, we had to reshape each image to be of the size 128*128.

3.2. For object detection

For object detection, we reuse some of the images we downloaded for classification. We trained our custom model to detect two classes - gun and hand. For this task, we also scrapped a few images of human hands, people holding stuff for our training. Since our aim is to recognize people holding guns and not just a gun present withing a frame, we wanted the network to learn

- Guns
- hands
- People holding objects
- People Holding guns

We had a total of 541 images, and we divided it into 491 for training and 50 for test. We used LabelImg[6] software to annotate these images. The software allowed us to draw bounding boxes in the image, and it was saved in .xml format. Now, tensorflow only accepts data in a particular format called TFRecords, so, we need to create TFRecords for our training. We do this by writing a script to convert the xml file to csv format using pandas and the XML library in python3. We then convert those files to the TFRecord format using the tensorflow api[9]. The TFRecord is required to train the data. During training, it saves various check points, which we then use to generate inference graphs, which is used in our object detection.

4. Technical Approach

4.1. Classification

To classify our images, we tested with various deep CNN models. The one we finally used had an architecture as follows-
$CNN- > CNN- > Pooling- > CNN- > CNN- > pooling- > CNN- > CNN- > pooling- > FC\ layer- > FC\ layer- > FC\ layer$

The architecture can be seen in Figure 1
We used leaky ReLU as an activation layer between the CNN layers and sigmoid activation between the Fully Connected layers. For the final output layer, we used a softmax layer to predict the output label. For each CNN layer, we used a filter size of 5*5 moving at a stride 1. We used a 2*2 max-pool layer with a stride of 2. For the leaky ReLU, we used the 'leaky' constant to be 0.01. We used cross entropy loss for cost calculation, and Adam as our optimizer. The architecture required the input image to be of the size 128*128. For training, we used a batch size of 10 and trained the model for 15 epochs. The loss curve per epoch can be seen in Figure 2.

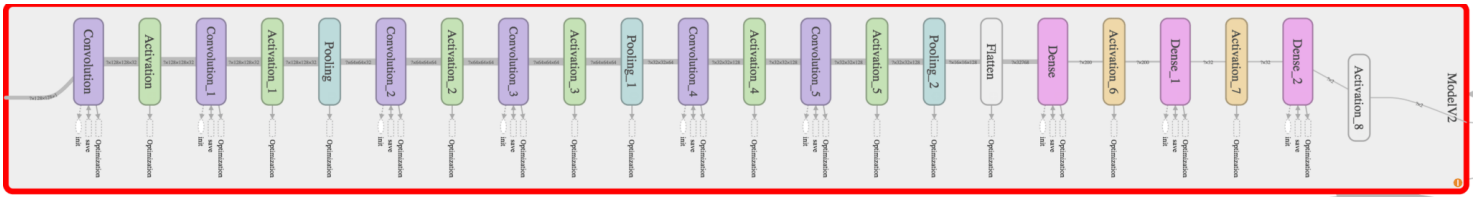

Figure 1. CNN Architecture

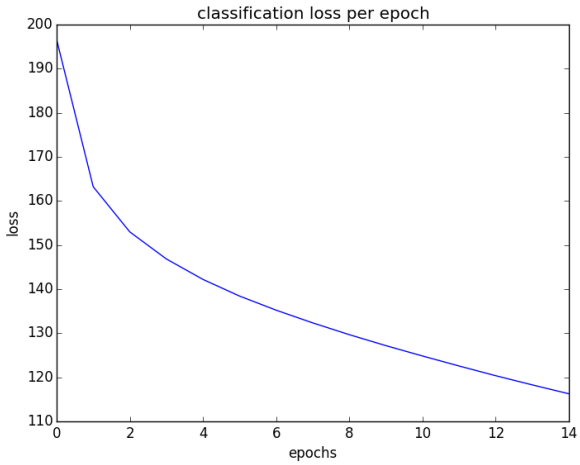

Figure 2. Loss per epoch

We ran the model on our test set and obtained an accuracy of 75%. We also applied to the classifier to each frame of a video.

4.2. Custom Gun Detection

For a task like gun detection, it is also important to know if a gun is in the frame, which direction it points towards. A normal classification task can only detect if a gun is present in a frame, it cannot track where. Hence to address this question, we train our custom model to detect guns along with human hands.

For this task, we use the Single Shot Detection(SSD) model which has been trained on the MS COCO dataset. It is faster than Faster R-CNN and other models, but is slightly less accurate than them. As our aim is to detect guns in live surveillance videos, we opted for the faster model.

As the name suggests, the SSD model performs region proposals and region classifications in the same step. It simultaneously predicts the bounding box and the class as it processes the image. At each location, it checks for different sizes of boxes, and predicts class probabilities and bounding box offset.

Because we are labeling hands and guns, from the training examples, we can see that usually the guns will be horizontal in nature and hardly be held vertically. Hence, we ran our model on aspect ratios 1,2,3 and 0.5. This allows us to check for guns mainly in the horizontal direction.

We also resized each image to be of 200*200 pixels to increase training speed. We incorporated dropout in the box predictor with a keep probability of 0.8 to induce stochasticity in our model. We have used weighted sigmoid for our classification loss and weighted L1 for our localization loss. We have given equal weightage to both these losses for our total loss. When we trained for a smaller set(100 examples) we noticed that we got a large percentage of false positives. For this reason, for every positive sample mined, we train 4 negative samples for the hard-negative mining. We have set the score threshold to be 1e-6, so that we only classify those boxes which lie above the threshold. As we have only 2 classes, setting a higher score helps to increase the training speed without losing out on important boxes. We have set the IoU(intersection over union) threshold to be 0.7, which also limits the number of boxes classified. The final output layer is a sigmoid which predicts the class scores. We have used RMS prop optimizer for gradient updates. We run our model for 200,000 steps, first for 100 annotated examples(and just one class) and then for 500 examples(for 2 classes). The batch size used is 3. (Anything more than that was giving a memory error on the system). The total loss curve for the training can be seen in Figure 3.

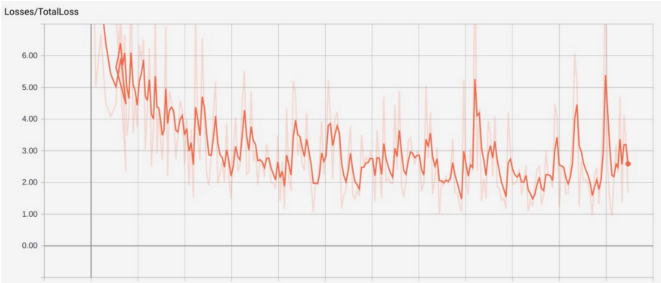

Figure 3. Total Loss

5. Results

For our classification, we tested on around 120 images and obtained an accuracy of about 75%. We also tested the classifier on videos. What we observed was for most cases, when a gun was on screen, the classifier was working correct, but it was giving many false negatives. This was also one reason we wanted more images for the hard

negative binding.

For our object detection, we ran our model on a few images, and also on videos. A few results can be seen below:

Figure 4. A perfect example

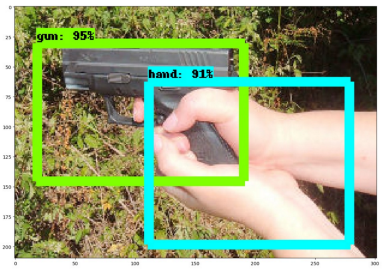

Figure 5. Hand and gun both detected

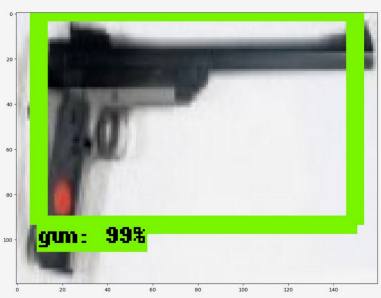

Figure 6. Gun detected in a very low resolution picture

Figure 7. Hand Detected

Figure 5 is an example where adding more training data and training over more iterations improved the results tremendously. In the model where we had just trained with a 100 examples, the model did not detect any object for the same image.

6. Conclusion and Future Work

Training the custom classifier from scratch even on a GPU was very time consuming. One of the things we would like to do is train for even longer duration with more annotated data on a high end gpu or using amazon ec2 instances. With better hardware and more time even faster-rnn based model can be used for detection as it gives a much higher accuracy. We could experiment and compare the results between both the implementations.

We can also pre-process the videos (like increasing their contrast, brightness, etc.) to reduce false positives. We also want to experiment with some hyper parameters so that we can get a better threshold for gun detection.

References

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik *Rich feature hierarchies for accurate object detection and semantic segmentation* arXiv: 1311.2524.pdf

[2] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks* arXiv:1506.01497

[3] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications* arXiv:1704.04861

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks.* In Advances in neural information processing systems, pages 10971105, 2012

[5] Greg Flitton, Toby P Breckon, and Najla Megherbi. *A comparison of 3d interest point descriptors with application to airport baggage object detection in complex ct imagery.* Pattern Recognition, 46(9):24202436, 2013.

[6] An image labelling tool created by tzutalin *https://github.com/tzutalin/labelImg*

[7] Richard Gesick, Caner Saritac, and Chih-Cheng Hung. *Automatic image analysis process for the detection of concealed weapons. In Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, page 20. ACM, 2009

[8] Zelong Xiao, Xuan Lu, Jiangjiang Yan, Li Wu, and Luyao Ren. *Automatic detection of concealed pistols using passive millimeter wave imaging.* In 2015 IEEE International Conference on Imaging Systems and Techniques (IST), pages 14. IEEE, 2015.

[9] A library created by cdatitran *https* : *//github.com/datitran/raccoon*$_d$*ataset*

Appendix

This section is for describing the distribution of work and shows some examples where are classifier did not work.

Rahul did the initial setup of all the libraries required for tensorflow. We have used an Nvidia Geforce 940MX present on the Lenovo 460p. We created a separate virtual environment in order to use python 3 with cuda 8.5. There is an issue with streaming videos using opencv 3 on linux so we had to use scikit-video library for capturing video frames. Both of did the data scrapping for our dataset and we got all the images from google image search.

Rohini annotated all the images by setting up the labelimg tool on her system.

Rahul worked on the custom CNN classifier and Rohini worked on the object detection architecture. Both of us worked on the scripts for changing the file formats for training the tensorflow model. We then toyed with the different models and tried a few different hyperparameters to see what works best for our application.

Earlier, we had annoted only 100 images, and had trained only for 1 class(gun). A few images from the previous model can be seen below. The loss function from training the model is shown in Figure 11.
(We can see that Figure 5 and 9 are the same. Nothing was detected in the old model, but it detected both clearly with the new one). We feel that if we train with more images, the model might perform better. We might want to look into images such that we have enough examples of the guns being held in all directions. A few videos that we have trained can be found here- `https://drive.google.com/drive/folders/1yiwg5kcGz-e45ZQ6bODZt2B3Ns7jiWHI?usp=sharing`

Our dataset and other required files can also be found in the same drive.

Figure 8. Gun detected with the old model

Figure 9. Nothing detected in this very clear picture

Figure 10. The background color makes this picture hard to detect

Figure 11. Loss with old image dataset