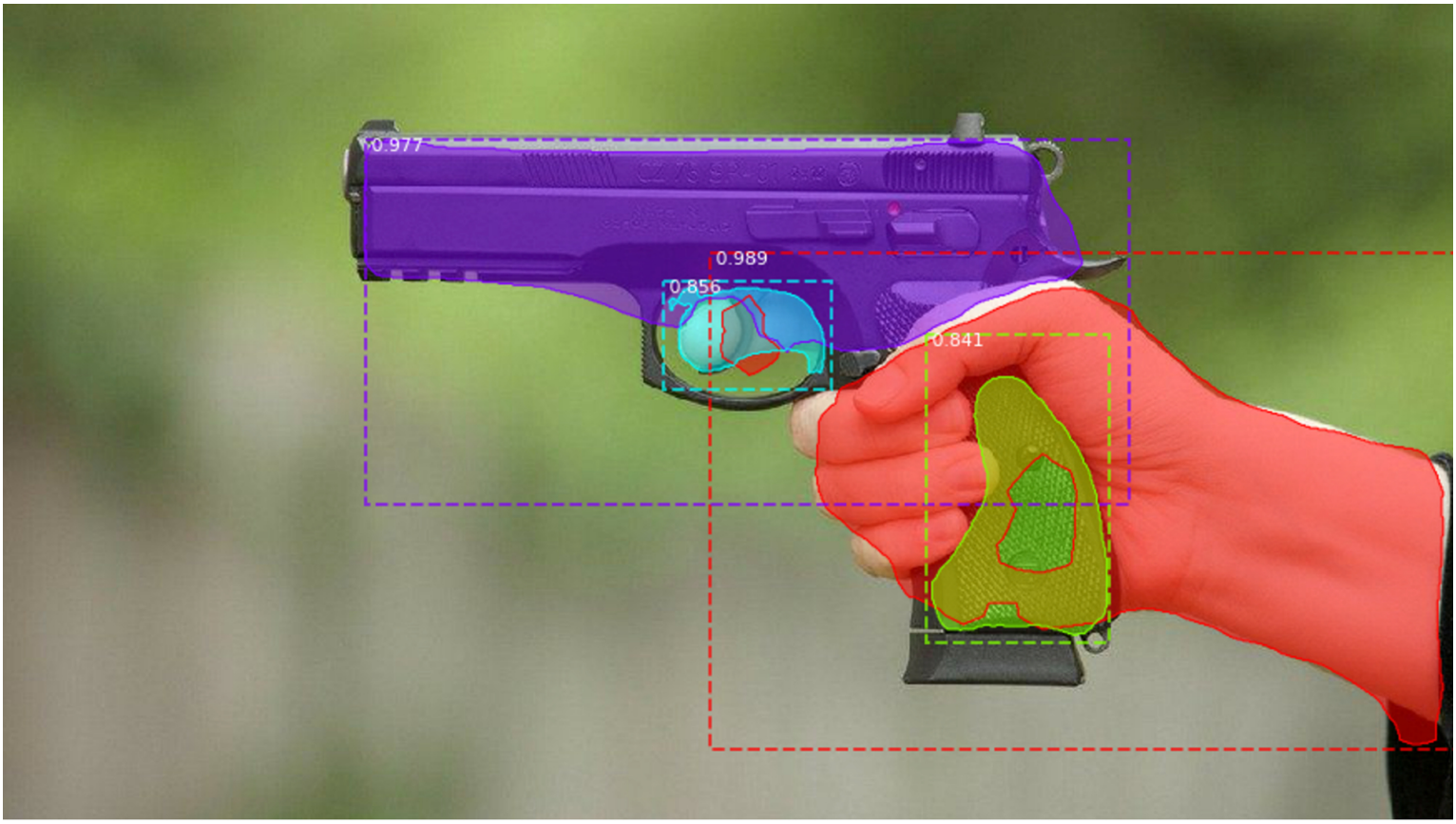
Recognizing Firearms from Images and Videos in Real-Time with Deep Learning and Computer Vision

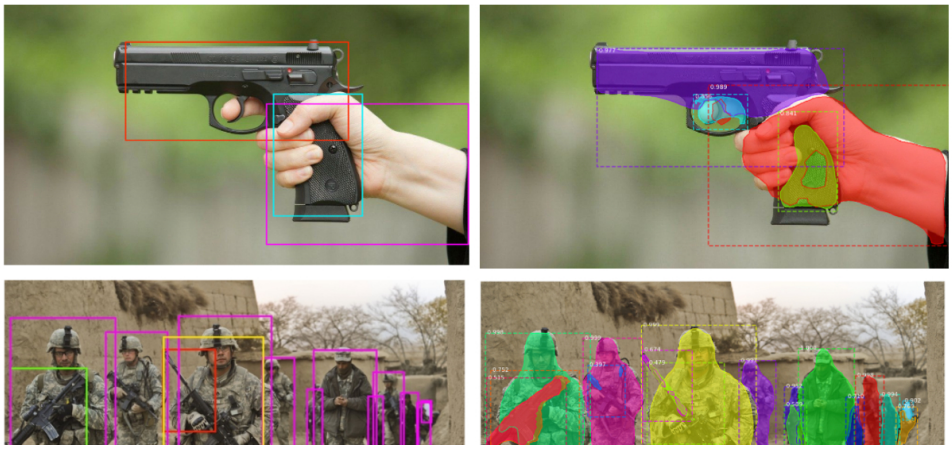
Tony Wang Aug 11, 2019 · 20 min read ★



Sample Pistol Model Output

Author's Note: This post consists of sections from a technical report I wrote in December 2018. I decided to share my findings in response to the recent accumulation of mass shootings.

TL;DR: I trained ML models that recognize certain firearms in images and videos, sometimes in real-time. Here are some of my results.



. . .

FYI, this is a long read, so this is a good place to stop reading.

. . .

1 Introduction

On October 1st, 2017, an avid gun collector and gambler began firing onto the Route 91 Harvest Festival Concert. In addition to the 58 who died, at least 851 were injured (422 of them by gunfire). The shooter killed himself after the roughly 11-minute attack. His suite contained a cache of 24 legally purchased guns, mostly semi-automatic rifles, many of them fitted with bump stocks that cause semiautomatic rifles to fire rapidly like automatic rifles [1].

On May 18th, 2018, 8 students and 2 teachers were fatally shot while 13 others were injured in a school shooting at Santa Fe, Texas. The shooter held a pump-action shotgun and a .38 revolver, both legally, and critically injured a police officer during cross-fire contact later that day [1].

On November 7th, 2018, 13 people, including a police officer and the perpetrator, were killed in a mass shooting in Thousand Oaks, California. With a legally purchased .45 Glock 21 pistol and a high-capacity magazine, the shooter fired approximately 30 rounds and injured 12 other victims [1].

As of November 11, 2018, there have been 49,366 recorded gun violence incidents in the United States this year alone, resulting in 12,588 deaths and 24,420 injuries, 2,449 of which were teenagers and 577 were children. Over the past decade, gun violence has resulted in tens of thousands of deaths and injuries annually — that is 100,000 recorded victims from 2008 to 2018. Records also show that over 400 *mass shootings* occur yearly —

more than one mass shooting a day on average [2].

According to the 2018 Small Arms Survey, U.S. civilians account for 393 million (about 46 percent) of the worldwide total of civilian held firearms, amounting to 120.5 firearms per every 100 residents, whereas U.S. state and local police hold only 822,000 firearms in total [3]. The vast difference between civilian-owned firepower and law enforcement firepower has been tremendously disproportional in the 21st century U.S. Many gun violence crimes happen in surveillance-covered areas, yet most systems demand human moderation, so very few crimes have been reported through these surveillance systems.

A special surveillance system that monitors firearm usage around the states in real-time using not human moderation, but computer vision, could aid law enforcement substantially on not only the cessation but the prevention of gun violence incidents and attacks. This report explores multiple potential technologies and concludes on one single solution tested with real-world usage in a controlled environment. In the following sections of the report, a "firearm" refers to a small arm potentially owned by civilians. The audience of this report is expected to have knowledge of basic machine learning and deep learning algorithms.

2 Problem Statement

Mass surveillance implementations for the sake of national security are not seldom seen, as there have been numerous attempts by the National Security Agency as well as the Federal Bureau of Investigation. This report, however, only investigates an implementation of a surveillance system that does not involve illegally gathering personal data. It focuses on implementing an image processing deep learning model and inputs CCTV footages from around areas with high-volume gun violence incidents.

2.1 The Engineering Problem

The main engineering problem incorporates recognizing and categorizing firearms within images and videos in real-time. The first step is to recognize the presence of firearms and differentiate image contents with firearms from contents without. After the basic recognition, the solution should be capable of categorizing different classes of small arms, ranging from handguns, rifles, carbines, sub-machine guns, assault rifles to light machine guns. Lastly, the system should be capable of continuous development on enhancements of speed, accuracy, and additional features.

2.2 Requirements

The absolutely highest priority when implementing a solution to the engineering problem is accuracy. When civilians' lives are at stake, even the slightest misjudgment could lead to tragic circumstances. The false positive of the predictions made should be less than 10% and the false negative should be no more than 3%. A false positive could easily be prevented entirely using human moderation; therefore false negatives are much less acceptable than false positives.

The second priority is the processing speed of the solution. Image processing has been a highly time-consuming and resource-oriented task. To process images and videos in real time requires a highly optimized solution running on extremely powerful machines. Since most CCTV footages contain a codec of 25 frames per second, the processing time per image should be less than 0.04 seconds.

Thirdly, the system should be able to handle a high throughput, since there could be a large number of processing requests coming from many different feeds. A realistic requirement is to be able to compute multiple images in parallel.

Lastly, the system should be modular and easy to implement across a large scale of different environments. It should be as effortless to implement on the fields of a high school as on the streets of downtown Manhattan.

3 Solutions

There are generally two different approaches when it comes to image object detection — classical mathematical model, and machine learning model. This section focuses on the machine learning approach and explores three kinds of potential deep learning solutions for recognizing and categorizing firearms in images and videos.

There are usually two different approaches regarding machine learning object recognition in images and videos — image classification and object detection. In Figure 1 below, the picture on the left represents image

classification and the one on the right represents object detection [4].

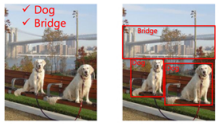


Figure 1 Image Classification vs Object Detection [4]

3.1 Object Detection Frameworks

Training accurate ML models that are able to localize and identify multiple objects in a single image remains a great challenge in the fields of image processing and computer vision [5]. However, there exist many object detection frameworks that allow fast prototyping and training with prebuilt templates and functionalities without concerns on the exact implementation of machine learning methodologies. This section focuses on two distinct frameworks: TensorFlow Object Detection API and DeepDetect Deep Learning API.

3.1.1 TensorFlow Object Detection API

TensorFlow is an open source ML framework for high-performance mathematical computations. Its flexible architecture allows fast deployment of services across a variety of platforms (CPUs, GPUs, TPUs) [6].

The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. Models can receive images as inputs and output categories, confidence and a coordinated bounding box around the object detected.

One could train target-specific models based on unique datasets and generate prediction outputs using such a model. Figure 2 below demonstrates how the API functions when a model trained with daily objects is used [6].

Figure 2 TensorFlow OD API Sample Output [6]

As can be seen from the picture, humanoid figures are recognized with high confidence while the parachute is wrongly identified as a kite. This is because the model was not trained on a dataset that includes parachutes. Instead, the model found the next closest thing that it could recognize. It did, however, drew the bounding box correctly around the parachute. With rapid training on a decently-sized dataset on different types of firearms, the TF OD API can be extremely powerful.

The advantage of using this API is that TensorFlow is one of the most mature open-source ML platforms with high efficiency in both training and processing, with consistent support and documentation from its product owner, Google. However, the on-going development is a double-edged sword. Constant patches to this platform mean fewer vulnerabilities and more robust system, but rapid updates often lead to a more deprecated code base. Therefore, it is more suitable for fast prototyping with a quick turnaround time than for a large-scale project that will be continuously developed and deployed in the long term. [6]

3.1.2 DeepDetect Deep Learning API

DeepDetect is a machine learning API written in C++11. It relies on external machine learning libraries through a very generic and flexible API. This API runs on Caffe, Caffe2, XGBoost, TensorFlow, T-SNE, and Dlib. Below in Table 1 is a comparison decision matrix between which library to choose as the background environment. Each library has its own compatible features with this API. The score received is 1 for a compatible feature and 0 for a non-compatible one. [7]

	Training	Prediction	Classification	Object Detection	Segmentation	Final Score
Caffe	1	1	1	1	1	6
Caffe2	1	1	1	1	0	5
XGBoost	1	1	1	0	0	3
Tensorflow	0	1	1	0	0	2
T-SNE	1	0	0	0	0	1
Dlib	0	1	1	1	0	3

Table 1 Deep Learning Frameworks Comparison for DeepDetect [7]

Since the training, prediction, classification, object detection, and segmentation are equally important to this project's needs, this is not a weighted decision matrix. From the matrix, it can be concluded that Caffe is the best choice.

Caffe is a deep learning framework, similar to TensorFlow in workflow and service pipelines. It is developed by Berkeley AI Research (BAIR). One distinct advantage of the Caffe implementation of DeepDetect is its speed. As can be seen in Figure 3 below, models trained on most common datasets, such as Res101 and Googlenet have impressive processing time. [7]

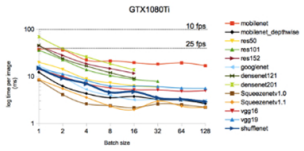


Figure 3 Performance Evaluation of Caffe [8]

3.2 Mask R-CNN

Ever since Alex Krizhevsky, Geoff Hinton, and Ilya Sutskever won ImageNet in 2012, Convolutional Neural Networks (CNNs) have become the gold standard for image classification. In 2014, Regions with CNNs (R-CNN) was introduced by a team in UC Berkeley for the purpose of object detection as an extension from image classification [5]. Figure 4 below showcases the pipeline of how R-CNN functions as an image object detection implementation [5].

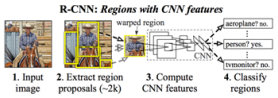


Figure 4 R-CNN Pipeline [5]

In 2015, a team at Microsoft Research proposed Faster R-CNN, in which only a single CNN needs to be trained. Faster R-CNN is, therefore, a simplified R-CNN solution with almost no cost for regional proposal. [5]

Mask R-CNN, developed by the Facebook AI Research Group, extends from Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression. This results in pixel-level image segmentation. [4]

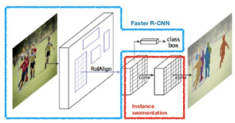


Figure 5 Mask R-CNN [5]

There are many implementations of Mask R-CNN. This section includes two most documented and supported implementations: Matterport Implementation on TensorFlow, and Facebook Detectron Implementation on Caffe.

3.2.1 Matterport TensorFlow

This is an implementation of Mask R-CNN on Python 3, Keras, and TensorFlow. The model generates bounding boxes and segmentation masks for each instance of an object in the image. It's based on Feature Pyramid Network (FPN) and a ResNet101 backbone [9]. Its installation instructions are clear and the usage is straightforward. This implementation seems to be mainly maintained by employees at Matterport (a 3D imaging company) and PhilFerriere, an ex-Microsoft AI. Figure 6 below is a sample output of the implementation using a model trained on the COCO dataset.

Figure 6 Matterport Mask R-CNN Sample Output [9]

Based on Matterport's implementation of Mask R-CNN, firearm recognition

and categorization can be done simultaneously.

3.2.2 Facebook Detectron

Detectron is Facebook AI Research's software system that implements a variety of object detection algorithms, including Mask R-CNN. It is written in Python and powered by the Caffe2 deep learning framework [10]. The implementation is extremely clean, scholarly written and easy to extend and maintain. Figure 7 below is a sample output of the implementation using a model trained on the COCO dataset.

Figure 7 Facebook Detectron Sample Output [10]

3.3.3 Comparison

Figure 8 below represents sample outputs from both implementations, with Matterport on the left and Detectron on the right. Both are using weights pre-trained on the COCO datasets.

Figure 8 Mask R-CNN Implementations Comparison [8]

As can be seen, the fundamental implementation is very similar. So, both implementations provide equivalent results. There are, however, some differences in performance between these two implementations.

Table 2 following shows three separate tests run on Detectron and Matterport under three similar environments.

Implementation	Python Version	DN Framework	Operating System	Time per image (1920×1080)	Time per image (640*480)
Facebook Detectron	2.7	Caffe2	Linux	640ms	316ms
Matterport	3.5	Tensorflow 1.4	Linux	290ms	190ms
Matterport	3.5	Tensorflow 1.4	Windows 10	620ms	490ms

Table 2 Mask R-CNN Implementations Performance Benchmark [8]

As can be seen, Matterport running on Linux has the fastest running time, meaning it can better handle real-time image processing given the right hardware.

The advantages of Mask R-CNN on Matterport lie in the implementation's flexibility since it is an open-ended framework with no restrictions on development that an API would bring. It would not, however, be cost-efficient to implement into pre-existing systems, since its actual functionality requires more time and resources to set up than an API library.

3.3 Darknet YOLO

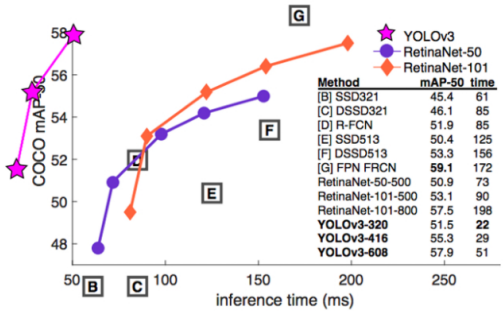
The You Only Look Once (YOLO) model treats object recognition as a regression problem instead of a classification problem. YOLO sees the entire image during training and has much better performance than Mask R-CNN [11]. Therefore, YOLO is considered a true real-time object detection system.

Darknet is an open source neural network framework that supports YOLO natively. It is written in C and CUDA, is fast, easy to install, and supports

CPU and GPU computation. Below in Figure 9 are two sample outputs from Darknet YOLO. On the left is an image output whereas on the right is a screenshot of video output.

Figure 9 Darknet YOLO Sample Output [11]

One of YOLO's distinct advantages is that it is incredibly fast, therefore fantastic for real-time image and video processing. Figure 10 below shows the great difference in efficiency between YOLO v3 and another commonly used image classification backbone, RetinaNet.



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 10 YOLO Inference Time Comparison [11]

Another benefit of using YOLO is that predictions are made with one single trained network, meaning models can be trained from end to improve accuracy. YOLO also accesses the whole image when predicting bounding boxes, producing fewer false positives when detecting objects [11].

4 Engineering Analysis

This section contains the decision matrix on the machine learning technologies from the solution section. It also contains documented real-world implementations of two most suitable technologies chosen from the decision matrix.

4.1 Decision Matrix

This section uses a weighted decision matrix to select better solutions out of the possible ones with engineering analysis. The matrix takes into account all four variables mentioned in the requirement section of the problem statement: accuracy, speed, throughput, modularity. Table 3 below shows the matrix in full detail. Scores are given on a scale from 0 to 100, with relative competence against other technologies as well as the requirement specifications.

Solution\Rubrics	Accuracy	Speed	Throughput	Modularity	Final Score / 100
Rubrics Weight	35%	30%	15%	20%	100%
TensorFlow Object Detection API	60	55	80	90	67.5
DeepDetect Deep Learning API	50	60	60	80	60.5
Matterport Mask R-CNN	80	75	80	90	80.5
Facebook Detectron	75	40	50	90	63.5
Darknet YOLO	70	100	95	80	84.5

Table 3 Weighted Decision Matrix

Accuracy was assigned 35% of the final weight since it is the most important factor in the requirements. Matterport TensorFlow implementation of all the ML solutions has the highest accuracy, not only because it is the most matured system of all, but also because it has the ability to provide both bounding boxes and segmentation of the object detected. Using the same Mask R-CNN architecture, Facebook Detectron's accuracy is the next highest. TensorFlow OD API and DeepDetect API both

have lower accuracy than Darknet YOLO due to the fact that they are API solutions with limitations on functions and operations.

Speed is the second-highest weighted variable among the requirements, weighing 30% of the final score. Darknet YOLO with its incredible speed was given a score of 100. Detectron with its lowest speed benchmark was given a score of 50. TensorFlow OD API and DeepDetect are similar in speed, receiving a 55 and a 60 respectively based on their benchmarks. Matterport Mask R-CNN received a score of 75, for its possibility of optimizations on speed.

Throughput is the lowest weighted variable since it can be altered greatly based on hardware. YOLO received the highest score of 95, because of its special characteristic of scanning the entire image only once. Although deep learning methods for image segmentation seldom come with high throughput support, Matterport Mask R-CNN and TensorFlow OD API both received the second-highest score of 80. This was due to TensorFlow's native multi-GPU processing option. DeepDetect and Detectron both received relatively lower scores by comparison.

Modularity is the last variable, weighted at 20% of the final score. Since both TensorFlow and Caffe demonstrate great modularity, solutions that use those frameworks received scores of 90. DeepDetect and Darknet YOLO are, on the other hand, less documented with a more easily mutated code base, yet they are still used by many professional software services. Therefore, both receive scores of 80.

As can be seen from the decision matrix, Darknet YOLO is the best solution, with a final score of 84.5, followed by Matterport Mask R-CNN, with an 80.5.

Therefore, both Darknet YOLO and Mask R-CNN will be implemented as fully functional services for benchmarking.

4.2 Real-world Implementation

This section displays example outputs from Matterport Mask R-CNN as well as Darknet YOLO when implemented in the same environment. The models used are trained with the same datasets.

4.2.1 Darknet YOLO

Darknet YOLO was implemented on a 2018 MacBook Pro and a model was trained using pre-trained COCO weights as well as images of certain categories. Figure 11 below is the output from the rifle model trained with a rifle dataset of 50 images. Respective bounding boxes were created manually around the rifle for each of the images and fed into training.

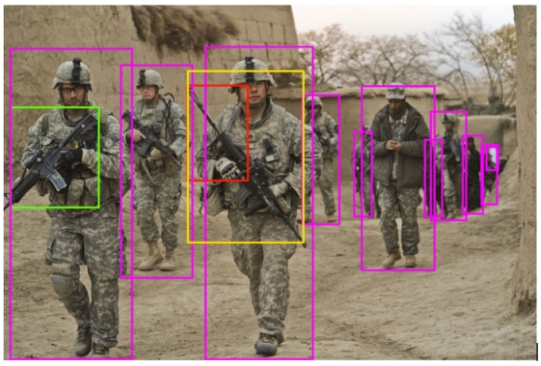


Figure 11 Darknet YOLO Rifle Model Output

An image of the U.S. Marines on patrol was fed into the model and the model successfully detected two out of three M16 rifles in the image. It was not able to detect the third rifle due to YOLO's single scanning characteristics, which usually leads to faster speed and lower accuracy. The purple boxes represent the humanoids the model detected using COCO dataset. This can be used for threat analysis combined with firearm detection for future work.

A pistol model was also trained using the YOLO implementation with 100 pistol images because the pistol images are much easier to obtain and manipulate. For training data, bounding boxes around not only the pistol but around the magazines and triggers were drawn. Due to the high-volume dataset, the pistol model is more confident and accurate. Figure 12 below shows a sample output of the Darknet YOLO pistol model.

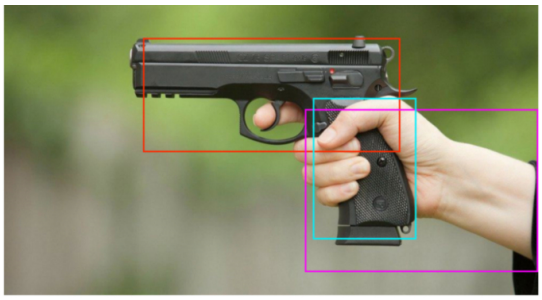


Figure 12 Darknet YOLO Pistol Model Output

As can be seen in the sample output, the red box represents the model's interpretation of the pistol itself. The cyan box represents its interpretation of a magazine, whereas the purple box represents the hand.

Both the rifle model and the pistol model only spent less than 0.1 seconds on processing each image. This is a good demonstration of how the model can be used for quick analysis of the occurrences of firearms in videos.

4.2.2 Matterport Mask R-CNN

Matterport Mask R-CNN was implemented on a 2018 MacBook Pro and a model was trained using pre-trained COCO weights as well as images of certain categories. Figure 13 below is the output from the rifle model trained with a rifle dataset of 50 images. Respective contour masks were created manually around the rifle for each of the images and fed into training.

Figure 13 Matterport R-CNN Rifle Model Output

An image of the US Marines on patrol was fed into the model and the model successfully detected all three M16 rifles in the image. However, the contours drawn by the model did not cover the full span of two rifles. This is likely due to the complexity of the image and the size of the dataset. Should more data be added to the dataset for training, it is certain that the segmentation output will be more accurate.

A pistol model was also trained using the Matterport Mask R-CNN implementation with 100 pistol images because the pistol images are much easier to obtain and manipulate. For training data, contour masks around not only the pistol but around the magazines and triggers were drawn. Due to the high-volume dataset, the pistol model is more confident and accurate. Figure 14 below shows a sample output of the Mask R-CNN pistol model.

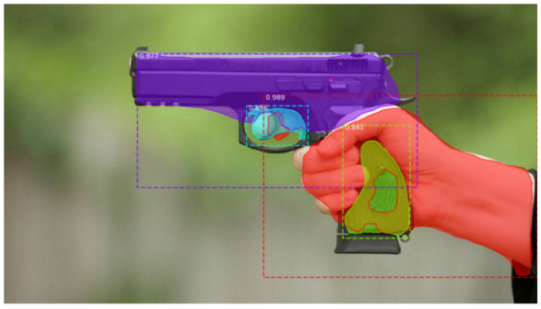


Figure 14 Matterport R-CNN Pistol Model Output

As can be seen, the purple contour covers the pistol, the cyan contour covers the trigger, the green contour covers the magazine chamber and the

red contour covers the hand. This information provides incredible insights into the occurrences of firearms in images.

However, the processing speed of both the rifle model and the pistol model is quite slow — both at around 1.5 seconds per image. That is hardly enough for real-time video processing. The accuracy is relatively high compared to YOLO and with similar dataset and similar run-time environment, it did recognize more objects with higher confidence.

5 Conclusions

From the analysis in the report body, it was concluded thatboth Darknet YOLO and Mask R-CNN should be implemented as a combined solution to the real-time object detection system for firearms. When using deep learning methods for object detections, speed and accuracy are the two vital components to measure performance with. It is almost certain that there will be a trade-off between speed and accuracy, so for the implementations of a real-time detection system, the accuracy of the results can be greatly compromised.

Various object detection APIs were investigated. They were provided by machine learning frameworks such as TensorFlow OD API and DeepDetect APIs for their quick development cycle and templating. However, it turns out that while APIs are easy to use, their performance falls short when compared to actual code-based implementations. This is because APIs offer less control and manual tuning that could improve performance.

Then, Mask R-CNN methodology and its implementation were explored. Mask R-CNN offers instance-level segmentation and classification, meaning the model not only returns object detection, but also draws a bounding box as well as a contour around the area on the image the model predicts to be a firearm. The implementations of Mask R-CNN provide promising accuracy and customizability compared to the deep learning APIs, yet they are still not ideal for real-time video processing.

Lastly, Darknet YOLO is an implementation of YOLO (You Only Look Once) that produce results with high speed. Models only need to be trained once, even for inferences of different sizes. This significantly improved the ease of implementation relative to other methods. However, YOLO's performance on accuracy is slightly worse than Mask R-CNN's. This is because Mask R-CNN uses thousands of iterations through the single image while YOLO only does it once.

Using a weighted decision matrix with metrics including accuracy, speed, throughput, and modularity, it can be seen that both YOLO and Mask R-CNN are ideal solutions, with their own trade-off between speed and accuracy. Therefore, YOLO and Mask R-CNN were implemented on a personal laptop for benchmarking and comparison. Both are trained with identical datasets in a similar timeframe. From the real-world implementation results, Mask R-CNN does have a lower false negative, but YOLO retains its power in speed, able to perform true real-time video processing.

It is concluded that a combination of YOLO and Mask R-CNN should be used for the final solution. YOLO can be used for the preliminary scan on video footages in real-time, and Mask R-CNN for the secondary scan should the preliminary scan return any positive identifications on the presence of firearms. This way, the speed and throughput requirements of the system are met, yet it does not lose its accuracy.

6 Recommendations

Based on the analysis and conclusions in this report, it is recommended that Darknet YOLO and Matterport Mask R-CNN should be implemented together as a combined solution. There are a few recommendations on different aspects of the solution.

First, the costs of the concluded implementation can be categorized into two main sources –training cost and running cost. Training cost comes from dataset collection, model and weights training, and initial human intelligence moderation. Running cost comes from running the service that continuously processes video footage, either with cloud computers or physical machines. It is recommended to utilize the dataset collected for both YOLO model training as well as Mask R-CNN model training, even though YOLO training requires only bounding boxes while Mask R-CNN requires segmentation masks. This reduces the costs of data collection by half.

Next, the solution needs a group of data scientists, software developers and

human intelligence workers to implement. Human intelligence workers will collect training data on different categories of firearms including bounding boxes and segmentation masks. Data scientists will then use the data to train the YOLO model as well as the Mask R-CNN model. As soon as the models finish training, the system can be maintained by the software developers. It is recommended that the post-training implementation should be fully-automated for self-correction. This will ensure both models to continuously train on new data reflect by users and will increase accuracy over time.

One other recommendation is on the future potential improvements of the design. Currently, the concluded design uses Darknet YOLO for real-time monitoring and sends footages with positive recognition results to Mask R-CNN in the backend for better accuracy. This might introduce false negatives since YOLO might not recognize certain objects with accuracy as high as Mask R-CNN and miss footage to send to Mask R-CNN for further predictions. A potential solution is to periodically send one frame of the live footage to Mask R-CNN to prevent the missed predictions of YOLO implementation.

Lastly, the system is artificial and deep learned, so its validity and correctness need to be continuously monitored and moderated. Since the implementation is solving a problem that is clearly of high stakes, the accuracy of the prediction should be tested before and during production. It is recommended that developers use precautions when testing the system before putting the service to production.

Additional Sample Outputs

Glossary

AI: Artificial Intelligence.

API: a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.

Bump stock: devices used with a semiautomatic firearm to increase the firearm's cyclic firing rate to mimic nearly continuous automatic fire.

CCTV: Closed-circuit television.

CNN: Convolutional Neural Networks.

COCO: a large-scale object detection, segmentation, and captioning dataset.

CUDA: Compute Unified Device Architecture.

Mask R-CNN: Mask Recurrent Convolutional Neural Network.

OD: Object detection.

TF: TensorFlow.

TPU: Tensor processing unit. A chipset specifically designed for Google's TensorFlow framework.

YOLO: You only look once.

References

[1]

D. L. a. C. A. Bonnie Berkowitz, "The terrible numbers that grow with each mass shooting," Washington Post, [Online]. Available: https://www.washingtonpost.com/graphics/2018/national/mass-shootings-in-america/?noredirect=on&utm_term=.119b20bb4794. [Accessed 18 November 2018].

[2]

"Gun Violence Archive," [Online]. Available: https://www.gunviolencearchive.org/. [Accessed 18 November 2018].

[3]

"Small Arms Survey," [Online]. Available: http://www.smallarmssurvey.org/. [Accessed 18 November 2018].

[4]

E. R. Davies, Computer Vision 5th Edition, Academic Press, 2017.

[5]

I. G. a. Y. B. a. A. Courville, Deep Learning, MIT Press, 2016.

[6]

EdjeElectronics, "TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10," [Online]. Available: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10. [Accessed 18 November 2018].

[7]

jolibrain, "Deep Detect," [Online]. Available: https://github.com/jolibrain/deepdetect. [Accessed 18 November 2018].

[8]

J. H. F. H. B. S. X. B. Y. Z. L. Z. Gui-Song Xia, "A Benchmark Dataset for Performance Evaluation of Aerial Scene Classification," [Online]. Available: http://captain.whu.edu.cn/project/aid/. [Accessed 18 November 2018].

[9]

Matterport, "Mask R-CNN," [Online]. Available: https://github.com/matterport/Mask_RCNN. [Accessed 18 November 2018].

[10]

Facebookresearch, "Detectron," [Online]. Available:

https://github.com/facebookresearch/Detectron. [Accessed 18 November 2018].

[11]

J. a. F. A. Redmon, “Darknet YOLO,” [Online]. Available: https://pjreddie.com/darknet/yolo/. [Accessed 18 November 2018].

Published in **Unsung** · May 14, 2018

UW三年

UWaterloo是个神奇的地方，更加神奇的是她的Engineering，读此专业的人从四面八方而来，背景也千奇百怪。作为Eng必须要Co-op，所以就这么在每四个月一换的学习期的痛苦和工作期的快乐中，走过了三年。工程系学习期的煎熬在和同学们的打打闹闹下就过去了，而工作期相比之下就像是个美化了的，有薪水的假期。不论什么系，这三年印象中最深的并不是学习经历，不是工作经验，而是人。学习期是认识固定的一帮人，不管工作离开多远，回到滑铁卢一定会和这些伙伴聚到一起，非常有归属感。工作期是认识不固定的各种人，社会人士，职场同事，且都见识甚广。大学对我来说，知识并不是首要。在信息技术如此发达的今天，谁还需要大学的辅助以读书学习？这些因UW相聚相识的人们，才是对我来说最宝贵的收获。

如下流水账似的记录：

第一个宿舍聚会

Published in **Unsung** · Jan 15, 2018

记 相机历史

. . .

一零零九年，秋，我手里拿到了第一部相机。

数码单镜头反光照相机，还是可更换镜头的一部入门级摄影机器。零九年我十一岁，刚开始中学；听许多朋友说，高中了才会后悔初中没有多参加社团，想进入所谓的社团们玩玩。且参加社团可以名正言顺的翘课，每周五的最后一节可以不上，直接去社团。于是我报名了摄影社，心想没有比摄影更轻松的社团了罢 — — 就是按按快门，洗洗照片而已。一心不想上课的我，求着爸和妈赞助属于我自己的相机。顺便还可以有个新玩具玩玩。

这玩玩就是六年。

Read more in Unsung · 6 min read

Published in **Unsung** · Oct 31, 2017

#4 solution to everything

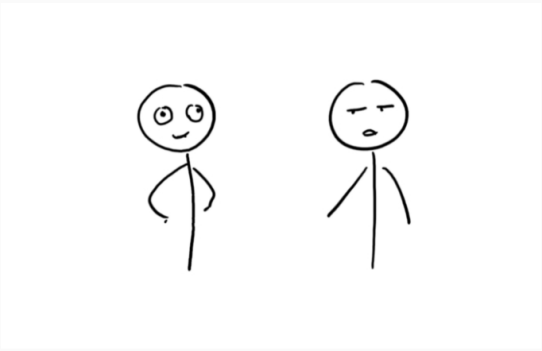
John: “I’m sick but I have work tmr, what do I do?”

Joe: “Just figure it out.”

Read more in Unsung · 2 min read

Published in **Unsung** · Oct 31, 2017

#3 english as a second language



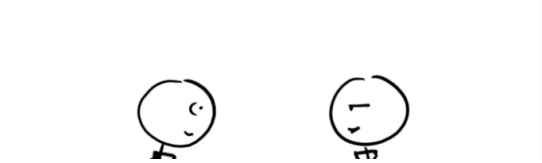
Que Xia Xiaomi Huawei Toyota: Yo bro, watch out for the lake!

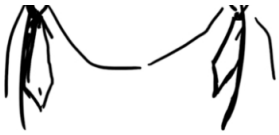
Bob: What lake?

Read more in Unsung · 1 min read

Published in **Unsung** · Oct 31, 2017

#2 the interview





Bruce Wang: Hi my name is Bruce and I'm perfect for this position.

Interviewer: Why is that?

Read more in Unsung · 1 min read

More From Medium

Livestock monitoring using Machine Learning and Computer Vision approach
Quantum in The Startup

How to Save Model Training Time Using Callbacks
Asep Saputra in codestorm

PRADO: Text classifier for mobile applications
Vaibhav Tiwari in Analytics Vidhya

Serving ML with Flask, TensorFlow Serving and Docker Compose
Usen Osas in Analytics Vidhya

Practical AI : Generate English pronoun questions from any story using neural coreference...
Ramsri Goutham in The Startup

A Gentle Introduction Into The Histogram Of Oriented Gradients
Karthik Mittal in Analytics Vidhya

How social scientists can use transfer learning to kickstart a deep learning project
Onyi Lam in Pew Research Center: Decoded

Creating an Audio Deepfake With YouTube
Alan Liu