Gun Identification Using Tensorflow

Mitchell Singleton, Benjamin Taylor, Jacob Taylor,
and Qingzhong Liu(✉)

Department of Computer Science, Sam Houston State University,
Huntsville, TX 77341, USA
{mitchellsingleton, benjamin.taylor,
jacobtaylor, liu}@shsu.edu

Abstract. Automatic video surveillance can assist security personnel in the identification of threats. Generally, security personnel are monitoring multiple monitors and a system that would send an alert or warning could give the personnel extra time to scrutinize if a person is carrying a firearm. In this paper, we utilize Google's Tensorflow API to create a digital framework that will identify handguns in real time video. By utilizing the MobileNetV1 Neural Network algorithm, our system is trained to identify handguns in various orientations, shapes, and sizes, then the intelligent gun identification system will automatically interpret if the subject is carrying a gun or other objects. Our experiments show the efficiency of implemented intelligent gun identification system.

Keywords: Tensorflow · Gun detection · Video surveillance

1 Introduction and Background

Automatic video surveillance is utilized by many people and organizations in the video surveillance category of their physical security protocol. CCTV and webcams are used in many products, from specialized doorbells with cameras and microphones that allow for two-way communication and recording when an event is triggered, or Closed-Circuit Television's (CCTV) which are used to aid security personnel in preventing or helping mitigate an incident. These systems use video in some form of method to record what the camera sees. While it may be easy for a homeowner to review a single system that has 1 or 2 cameras, the task becomes an arduous and tedious even when combing through multiple video streams if they implement a larger array of cameras. Depending on an organization's footprint and their implementation of their physical security plan, security personnel may be monitoring 100's of video streams [1, 2]. These streams should be prioritized in order of importance to determine the number of personnel needed and how many streams can be observed per personnel.

Recently, there has been an increasing number of incidents involving handguns in various situations. To detect handguns, in [3], several models were utilized to detect pistols. These models used the VGG-16 based classifier and then classified the detection of the gun with either the sliding window or the region proposal approach. The authors went with the region proposal approach due to its faster detection speed –

L. Meng and Y. Zhang (Eds.): MLICOM 2018, LNICST 251, pp. 3–12, 2018.
https://doi.org/10.1007/978-3-030-00557-3_1

but also tested against the sliding window approach. To achieve a real-time outdoor concealed-object detection with passive millimeter wave imaging [4], the authors used a passive millimeter wave (MMW) imaging system ran at or around 94 GHz with 1 Hz frame rate. To attain automatic recognition, they used both global (aligning the body and the background together) and local (inside of the body is processed) segmentation levels to find the concealed objects. To automatically detect firearms and knives in a CCTV image, the authors in [1] used an approach utilizing MPEG-7 visual descriptions and a principal component analysis (PCA). They took each frame and removed everything except the foreground of the image which included the erosion and dilation. To detect weapons in surveillance camera images [5], the authors improved detecting weapons in CCTV via the Histogram of Oriented Gradients (HOG) method for classification. In [6], a framework was developed to identify general objects in video using still image processes along with object tracking through the 4th dimension of time. In [7], a real-time detection, tracking and classification of natural video was examined. The method uses background subtraction and maximally stable extremal region (MSER) detection. However the method didn't successfully reach real-time. Recently a method to detect visual gun by using Harris interest point was proposed [2], the author uses a visual gun detection based on FREAK descriptor to recognize guns. The system processes an image, performs color segmentation, performs a boundary extraction and compares against a similarity higher than 50% and then executes an alarm. This algorithm was accurate 84.26% of the time and shows promise.

In this paper, we focus on detecting guns vs no guns. We propose a detection method that is built on a lean convolutional neural network optimized for speed. This paper is organized into four additional sections. Section 2 explains the definitions and tools where we will explain the CNNs and tools used to train our algorithm. Section 3 describes the methodology and the effectiveness of our algorithm, followed by our conclusions in Sect. 4.

2 Definitions and Tools

Tensorflow (https://www.tensorflow.org/) is a platform that is based on dataflow graphs and is useful in training with deep neural networks [8]. Tensorflow was created by Google and it is used in their native applications and in dozens of machine learning projects such as: facial recognition, image processing, speech recognition, and extracting information such as license plate numbers. Tensorboard, a web application, is a visualization for understanding Tensorflow trainings and monitoring the training progress in a web browser [9]. It showcases visualizations such as: graphs, scalars, and images. Tensorboard shows cross entropy and accuracy depicted into a graph for convenient visualization of training results. Tensorboard can be started from an Anaconda prompt and uses port 6006 on the localhost for its process.

OpenCV (https://opencv.org/) is an open source computer vision library that contains image processing functions and over 2,500 algorithms used for things like facial recognition [10]. OpenCV can accelerate CUDA and OpenCL GPUs. OpenCV supports deep learning platforms like Tensorflow. OpenCV is built using a layering

process. You have the OS, then the languages and algorithms, the core of OpenCV, and at the bottom the hardware acceleration layer (HAL) [11].

MobileNet is a collection of vision models for Tensorflow that are mobile-oriented and used to enhance visual recognition [12]. It is essentially a convolutional neural network (CNN). MobileNet, unlike a typical CNN, separates the convolution into a 3×3 depth wise convolution and a 1×1 pointwise convolution. The reason MobileNet is used more than a traditional CNN is the computation time is much quicker with their architecture, however the accuracy can be slightly lower. Depending on the architecture, such as MobileNet_0.50.224 or 0.50.192, it can change the width multiplier and the image resolution. The image resolution is the 192 or 224 and the width multiplier is the .50. This will help the object detection computation time of the image processing the smaller the image resolution. The width multiplier can be ranged from 0.25, 0.50, 0.75, and 1.0. The image resolution can be 128, 160, 192, or 224.

ImageNet (http://www.image-net.org/) is a very large dataset of images (over 14 million) with about 22,000 categories and are collected by human labelers [13]. ImageNet compiles hundreds of thousands of thumbnails and URLs of images just like Google, a search engine, does. ImageNet is organized via synsets which is a concept of WordNet. ImageNet was created to be utilized by researchers as a primary resource for images.

Anaconda (https://www.anaconda.com/) is used for large quantities of data processing, predictive analytics, and scientific computing [14]. It has over 1000 packages in its repository that can be installed, 150 of which come pre-installed with Anaconda. It is an open-source distribution of Python used to ease and simplify package administration and distribution. The packages are managed by a package management system within anaconda called conda.

Nvidia Compute Unified Device Architecture (CUDA) is utilized by many as their primary method to process images because it essentially groups the cores of GPUs into a vector and can then be programmed to decrease processing time on the large sets of data versus CPUs that do not run on a parallel throughput architecture [15].

Shi-Tomasi corner point detection is an enhancement to the Harris corner detector. The way Harris corner detection works is that the pixel is calculated and if above a precise value then it's marked as a corner. To score it, two eigen values are used to check if it's a corner during detection and then inputted into a function which manipulates them. Shi-Tomasi decided to enhance this method and only rely on the two eigen values, not the entire Harris formula. This is because their model relies solely on the tracker's accuracy [16].

3 Methodologies

The dataset pictures were gathered from the Sun database [17], Internet Movie Firearm Database [18], Pixabay [19], ImageNet [20] and reduced in number to 3363 gun images and 11834 non-gun images. The dataset was trained several times adding or removing pictures from the datasets to improve accuracy.

The method utilized in this paper was to retrain multiple object detection models (inception_v3, MobileNet_0.50_192, MobileNet_0.50_224, MobileNet_1.0_224) for a

binary classification of an object into either gun or no gun using a training dataset composed of firearms and non-firearm pictures. See Fig. 1 for the MobileNet architecture training flow chart.

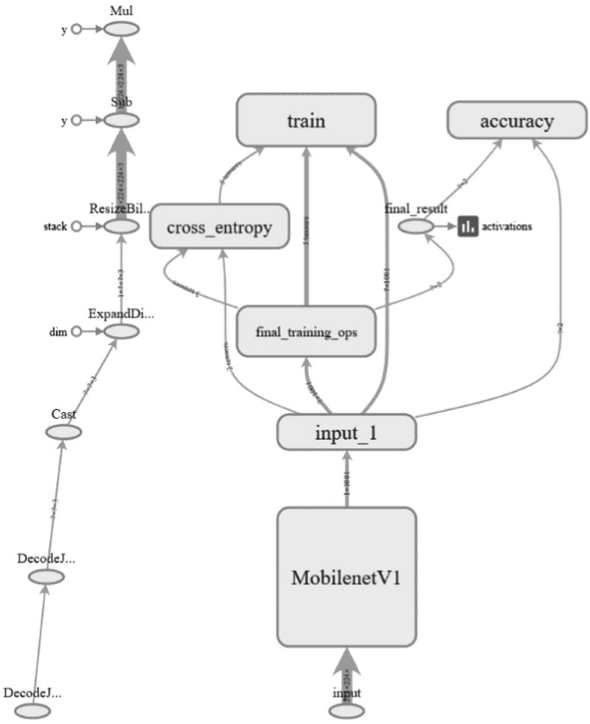

Fig. 1. MobileNet 1.0 224 training flowchart

The use of Tensorboard allowed watching the progress of the retraining and in Figs. 2 and 3 the visualization of the progress of the retraining can be seen from step 60 to step 4000 respectively.

Once the retraining was complete, Shi Tomasi key point detection was used to identify the parts of a picture that should be looked at by the binary classification. The assumption being that the key points would be less than the number of sliding windows across the whole picture.

The key point locations are then each used to crop out a section of the image and the retrained model is called to determine if the cropped section contains a gun or not. If a gun is found, then the cropped section has a rectangle drawn on the picture in the same location as the cropping. In the next section, we will review the results of our current models and some examples of identified pictures in the gun category and correctly identified pictures in the non-gun category.

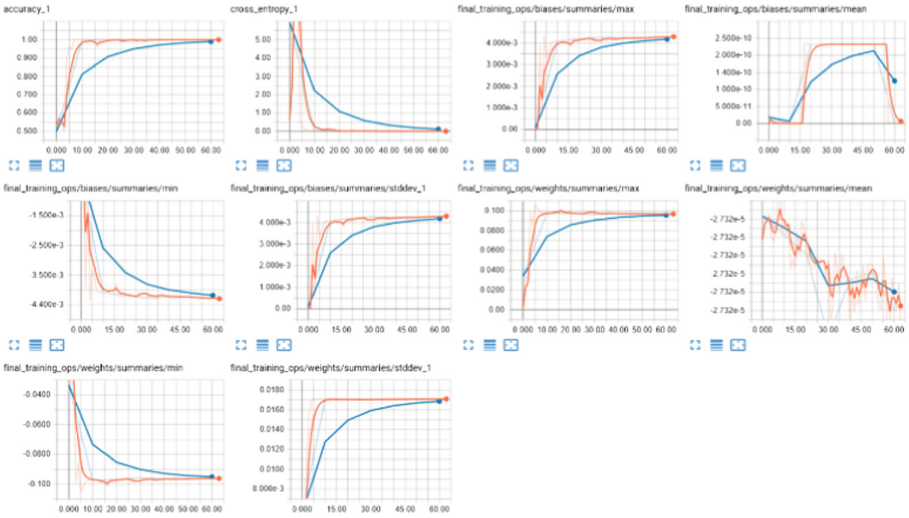

Fig. 2. Starting out the retraining

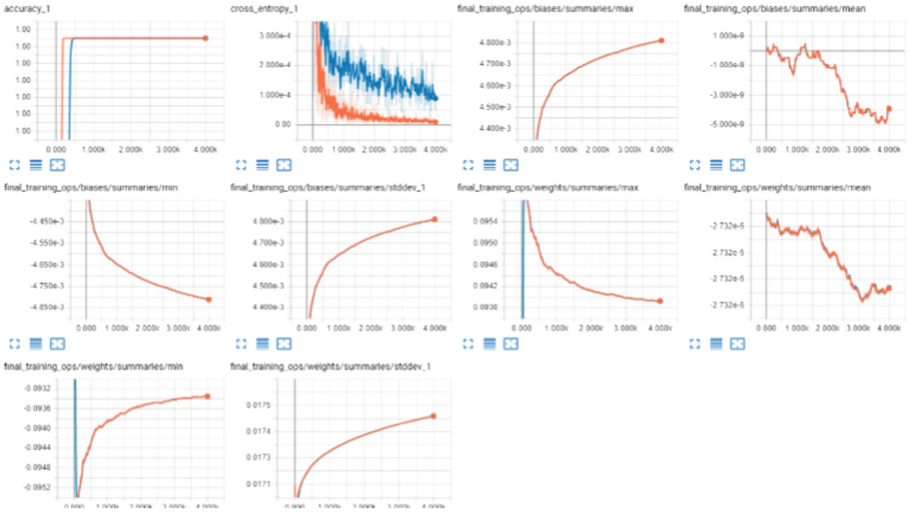

Fig. 3. Finished with the retraining

4 Results

The gun detection was trained and tested on the following datasets described in Sect. 3. The images detected in a fast method and we would like to expand upon static images to real time video as our algorithm is improved. MobileNet was trained in various step counts and tested for accuracy. In one training iteration, we trained 3000 steps as seen in Table 1 and resulting in 97.89% accuracy. In another iteration, we trained the

MobileNet 1.0 224 model 5000 steps and the accuracy increased to 100%. Our most accurate training algorithm was trained on 795 guns and 3944 non-guns. Testing using the results from this training resulted in the least false-positives compared to our dataset that included more pictures in the gun dataset. Results from this iteration of testing gave a 96% accuracy when testing against non-gun photos with the guns filtered at 89%. When testing against photos with guns, we have an accuracy of 86.67% out of 30 photos. These calculations can be referenced in Table 1 (Fig. 4).

Table 1. Model detection results

	Pictures	Correct	Incorrect	Correct
Non-gun	25	24	1	96%
Gun	30	26	4	86.67%

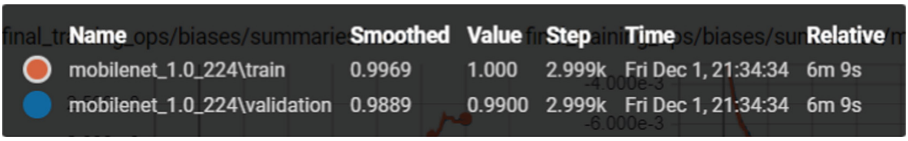

Fig. 4. MobileNet 1.0 224 retraining results

Our theory is accuracy will go up if we can preprocess the images with by subtracting the background, but this will have to be completed in another project. Some photos were missed because there is a limit of 25 corners maximum per image. Increasing this number will increase the amount of time the image must be processed, but we believe this will be minimized by removing or subtracting the background. Increasing the quality of our gun data set should improve the accuracy of the gun detection rate and will allow us to increase from 89% sensitivity without increasing false alarms, which can lead to alarm fatigue. We have included examples of correctly identified guns in Figs. 5, 6, and 7. The detection algorithm perform a gray transform. In Fig. 5, there were three positive hits. The background was uniform and the detection could identify the gun at least 3 times. It also correctly did not detect a gun on the subject's face, hair, earing, or shoulder. In Fig. 6, the subject had multiple hits on

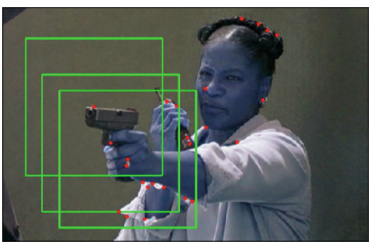

Fig. 5. Successful detection of a gun

various parts of the gun. In future iterations of the code, we will want to decrease the output of overlapping gun identifiers. If it finds multiple positive hits within x pixels, then either combine or ignore x screen draws.

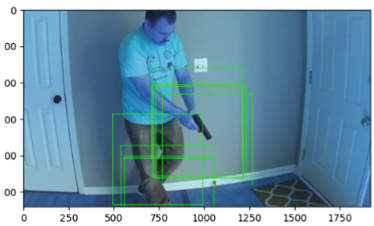

Fig. 6. Successful detection of a gun

In Fig. 7, you will see multiple identifiers on the base of the gun, on the barrel of the gun, and on a pistol like device in the bottom left corner. The background is the most difficult of the three different gun detection examples as the bricks do have similar hard angles that can be found in a gun. None of the bricks were identified as a gun so we are happy to report the background training with multiple forms of background photos in non-gun dataset has improved upon our earlier training and results. Again, we believe minimizing the number of identified boxes will clean up the display and will aid in the time of displaying the results in real time when this algorithm is improved and created for a real-time video system.

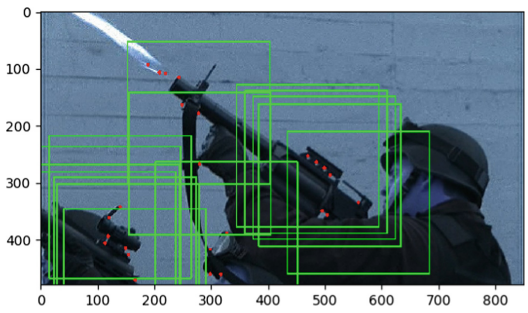

Fig. 7. Successful detection of a gun

We have also included correctly identified photos with no guns in Figs. 8 and 9. In Fig. 8, there are multiple hard edges in the grill and around the tires that have a gun shape. The algorithm was smart enough to not incorrectly give a false positive in all the dots that were run against. Other thing to note in the picture is the background was uniform with only having shadows at the bottom of the car. The shadows did not affect the detection of the algorithm and it performed its output correctly.

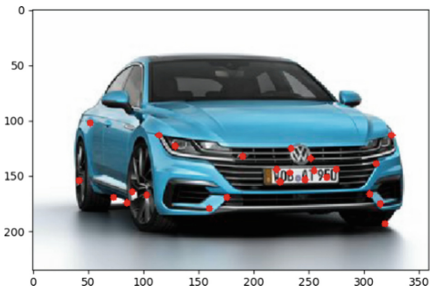

Fig. 8. Successful non-detection of a gun

Finally, our last example is shown in Fig. 9. In this example, the clouds and background did not affect the performance of the non-gun identification. The glasses and shadows in the pictures were also not a problem with the non-gun identifications.

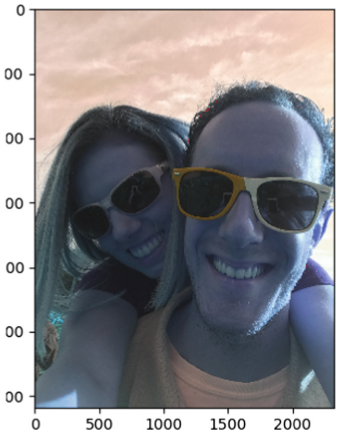

Fig. 9. Successful non-detection of a gun

5 Conclusions and Future Work

By looking at the presented photos we have successfully re-trained an existing MobileNet Neural Network of varying granularity.

The assumption that using key points alone would allow testing fewer places in an image proved to be inaccurate and was problematic when a test picture had more key points than the maximum amount defined for detection or when there were many key points that were detected in the same place. It was observed that in some cases so many key points were detected on non-gun objects that the gun never got assigned a key point and thus a gun was never classified because that area in the image was not evaluated. It was also observed that when many key points were detected in the same

place on a gun that the classification was ran on the same area of the picture for multiple overlapping classifications of gun.

One way to work around this issue would be to increase the maximum number of key points, however without any mitigation this would cause the total number of places in the picture needing to be looked at for classification to increase and remove the benefit of using the key point detection to decrease the instances of classification versus a sliding window over the whole picture.

Adding a processing step after the key point detection to remove key points by using a geometric density algorithm would allow keeping the number of classifications to a minimum. For example, in our test pictures, many of our key points are close enough that any gun detection boxes are overlapping.

Using a second step to remove key points would benefit both busy test pictures where the maximum number of key points was preventing a gun object from getting any key points and non-busy test pictures where all the key points are close to each other.

Adding another pre-processing step to remove the backgrounds would greatly increase the gun detection and decrease the false-positives. Finally, obtaining more pictures and continually training the dataset will allow it to continually learn and be more accurate.

To conclude, we have presented an early gun identification algorithm that we believe can be improved upon to allow detection of guns in images, video, and other applications. The goal is to give security personnel extra time to respond to a real gun alert.

References

1. Grega, M., Matiolański, A., Guzik, P., Leszczuk, M.: Automated detection of firearms and knives in a CCTV image. Sensors **16**, 1–16 (2016). https://doi.org/10.3390/s16010047. ISSN 1424-8220
2. Tiwari, R.K., Verma, G.K.: A computer vision based framework for visual gun detection using harris interest point detector. Procedia Comput. Sci. **54**, 703–712 (2015). https://doi.org/10.1016/j.procs.2015.06.083
3. Olmos, R., Tabik, S., Herrera, F.: Automatic Handgun Detection Alarm in Videos Using Deep Learning. arXiv:170205147 cs (2017)
4. Yeom, S., et al.: Real-time outdoor concealed-object detection with passive millimeter wave imaging. Opt. Express **19**, 2530–2536 (2011). https://doi.org/10.1364/OE.19.002530
5. Vajhala, R., Maddineni, R., Yeruva, P.R.: Weapon Detection in Surveillance Camera Images (2016)
6. Kang, K., Ouyang, W., Li, H., Wang, X.: Object Detection from Video Tubelets with Convolutional Neural Networks. Presented at the June (2016)
7. Ray, L., Miao, T.: Towards Real-Time Detection, Tracking and Classification of Natural Video. Presented at the June (2016)
8. Abadi, M.: TensorFlow: learning functions at scale. In: Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, p. 1. ACM, New York (2016)

9. Angermueller, C., Pärnamaa, T., Parts, L., Stegle, O.: Deep learning for computational biology. Mol. Syst. Biol. **12**, 878 (2016)
10. Pulli, K., Baksheev, A., Kornyakov, K., Eruhimov, V.: Real-time computer vision with OpenCV. Commun. ACM. **55**, 61–69 (2012). https://doi.org/10.1145/2184319.2184337
11. Kaehler, A., Bradski, G.R.: Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. O'Reilly Media, Sebastopol (2016)
12. Howard, A.G., et al.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:170404861 cs (2017)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Commun. ACM **60**, 84–90 (2017). https://doi.org/10.1145/3065386
14. Kadiyala, A., Kumar, A.: Applications of Python to evaluate environmental data science problems. Environ. Prog. Sustain. Energy **36**, 1580 (2017)
15. Saha, M.D., Darji, M.K., Patel, N., Thakore, D.: Implementation of image enhancement algorithms and recursive ray tracing using CUDA. Procedia Comput. Sci. **79**, 516–524 (2016). https://doi.org/10.1016/j.procs.2016.03.066
16. Shi, J., Tomasi, C.: Good Features to Track. Cornell University, Ithaca (1993)
17. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: large-scale scene recognition from abbey to zoo. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3485–3492. IEEE (2010)
18. Internet Movie Firearm Database. http://www.imfdb.org/
19. Pixabay. https://pixabay.com
20. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 248–255. IEEE (2009)