The Relationship Between Accuracy and Training Stage of a Haar Cascade

Joon Kang
Science IDS
June 5th, 2020

Table of Contents

Abstract

This paper provides a systematic approach of calculating the effectiveness of a Haar Cascade model for detecting objects in a 2d space, in an attempt to construct a relationship between the accuracy and training stage. During the investigation, we used a set of 2000 positive and 1000 negative images to train 17 stages of a Haar Cascade model, which we then processed through an algorithm to calculate the accuracy for each stage of the model. We later recognized a flaw within our accuracy calculations that undervalued the effectiveness of detection in the primitive stages and readjusted our algorithm to include a weighted detection rating. This new algorithm provided a more intrinsic, truthful dataset that more closely represented the real values. Analysing the data produced from both algorithms, we were able to make two major observations. Firstly, there lies a critical point, in which the rate of perfect detections jumps abnormally from 0% to 60~70% within one stage. Moreover, there exists a second critical point that occurs after the first, in which the accuracies start to diminish as stages progress rather than correlating positively with the stages. Using these critical points in conjunction with our findings, we were able to conclude that the accuracy of the training model increases approximately exponentially until it reaches a critical point in which the accuracy starts to decline. Additionally, there exists an optimal stage somewhere between the two critical points that provides the maximum detection accuracy.

1. Introduction

Artificial intelligence (AI) is the ability of computer systems to perform tasks requiring human-like intelligence. These tasks include, but are not limited to, recognizing auditory signals, detecting and classifying objects, predicting weather patterns, calculating chess moves, and generating autocomplete suggestions on a search engine. AI is used frequently to simplify, automate, or improve upon processes in our day-to-day lives, with one notable example being Apple's virtual assistant, Siri, and consequently, AI has become an integral part of our lives that influences the decisions we make.

For all of these AI to function as intended, they are required to be trained with a dataset, where generally, more data and longer training time suggests a more accurate system. Therefore, most complex AI are trained with large amounts of data for extended periods of time to ensure that they are as accurate as possible. This is no different when training an AI tasked to detect a certain object from a given image or video. While the method of training differs depending on the model, generally, there exists a positive correlation between accuracy and the two variables, data size and training duration.

The Haar Cascade model is an example of a type of model used to identify objects from a given image or video. First proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001 [7], the model is a machine learning based algorithm that works off the basis of increasingly accurate stages and is trained by a set of positive and negative images[1]. Each stage is trained by a process known as boosting, where a weighted average of its previous stage is used to reduce bias and variance of the new one, and this process is repeated until weak models become strong ones.

This paper focuses on the Haar Cascade model and the relationship between its accuracy and the current training stage. While it is known that the model will improve with higher stages, the margin of development of each individual step is unknown due to its complex nature. This paper will attempt to develop a relationship that will provide us with insight on how the effectiveness of the model increases or decreases as stages are progressed.

2. Materials

Host Computer

A standard 15-inch MacBook Pro 2017 was used, however, any computer with a Mac, Windows, or Linux-based operating system would suffice.

[1] Positive images contain instances of a certain object while negative images do not. In our case, the object is the hedgehog sticky note.

Linux System

An Ubuntu 18.04 Server on VirtualBox was used, however, any Linux based virtual machine or cloud server would suffice.

File-Transfer Protocol (FTP) Application

Filezilla was used to transfer files between the host computer and the Linux System, however, any alternative FTP application would suffice

OpenCV Library

The OpenCV library was used to create the Haar Cascade models and implement it to detect the object

Object to be Detected

A hedgehog sticky note was used, however, any other object would suffice.

Negative Images

A set of two thousand negative images from image-net.org were used to train the model

3. Procedure

Set-up

All prerequisite libraries and compilers were installed onto the Linux System (Git, CMake, OpenCV, Python). This was done with the following commands: [5]

```
sudo apt-get update

sudo apt-get upgrade

sudo apt-get install git

git clone https://github.com/Itseez/opencv.git

sudo apt-get install build-essential

sudo apt-get install cmake git libgtk2.0-dev
pkg-config libavcodec-dev libavformat-dev
libswscale-dev

sudo apt-get install python-dev python-numpy
libtbb2 libtbb-dev libjpeg-dev libpng-dev
libtiff-dev libjasper-dev libdc1394-22-dev

sudo apt-get install libopencv-dev
```

Obtain Image of the Object

A picture of a hedgehog sticky note was taken using a camera, then, it was resized to a size of 50x50 pixels.

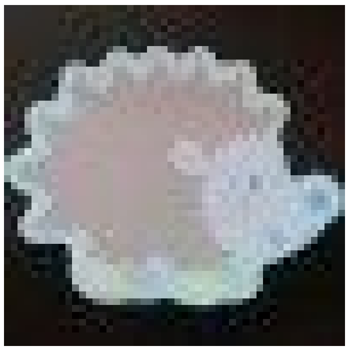


Figure 1: A 50x50 image of the object to be detected

Gather a Set of Negative Images

Using image-net.org, a set of 2000 negative images were gathered, then, they were grayscaled and resized to a size of 100x100 pixels. To ensure that the images weren't positive, all 2000 images were of trees and plants which share no resemblance to the original object.

Figure 2: Examples of negative images used to train the Haar Cascade model

Generate the Descriptor File

The descriptor file is a text file that contains data of the names and locations of the negative images, with a line for each negative image containing the relative path to said image. This was generated using a simple python script.

Upload Files to the Linux System

All gathered files were transferred to the Linux System using FTP. Upon completion, the Linux System contained the following: a 50x50 image of the object, a folder containing 2000 100x100 grayscaled negative images, and a descriptor file.

Gather a Set of Positive Images

To generate positive images, the rescaled image of the object was overlaid onto the negative images using the OpenCV library. This process automatically creates positive images from negative images without the need for manually taking thousands of photos of the object. Note that the 50x50 image is smaller than the 100x100 negative images on purpose to avoid errors when overlaying. This was done with the following commands: [5]

```
mkdir info

opencv_createsamples -img pic.jpg -bg
bg.txt -info info/info.lst -pngoutput info
-maxxangle 0.5 -maxyangle 0.5 -maxzangle
0.5 -num 2000

opencv_createsamples -info info/info.lst
-num 2000 -w 20 -h 20 -vec positives.vec
```

Since a positive image is generated for each negative image, there now is a total of 2000 positive images and 2000 negative images.

Generate the Stages of the Haar Cascades

After the set of positive and negative images were available, 17 stages of the Haar Cascade were created. This was done using the following commands: [5]

```
mkdir data

opencv_traincascade -data data -vec
positives.vec -bg bg.txt -numPos 2000
-numNeg 1000 -numStages 17 -w 20 -h 20
```

Note that only 1000 of the 2000 negative images were used to train the model. This is because not all 2000 negative images were gathered to be used to train the model, but rather to generate the positive images. To efficiently train the model, the number of positive images is double the number of negative images, thus, only a subsection of the total negative image set was used.

Film a Video Containing the Object

A 450 frame video was recorded to test the accuracy of the Haar Cascade models. The object appeared on each frame of the video.

Use the Haar Cascades to Detect the Object and Calculate the Accuracy

Since every frame of the recording contained the object, each frame was given a detection rating of either 0 or 1 depending on whether or not the Haar Cascade detected the object; a 0 meant no detection and a 1 meant successful detection. Then, by dividing the sum of the detection ratings by the total number of frames, the percent accuracy was calculated for the corresponding Haar Cascade. This was repeated for each of the 17 stages to obtain 17 accuracy values.

4. Findings and Analysis

We trained a 17 stage Haar Cascade model to detect a sticky note in the shape of a hedgehog from a set of 2000 positive images that were procedurally generated and 1000 negative images that were found on the web. Then, by separating the Haar Cascade into each of its steps, and attempting to detect the object for each individual frame of a standardized video, we were able to count the number of successful detections for a given model corresponding

	Stage																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Detection Rating Sum	0	0	0	0	0	0	0	0	0	4	285	323	343	352	345	315	289
Accuracy	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.89%	63.33%	71.78%	76.22%	78.22%	76.67%	70.00%	64.22%

Table 1: Perfect detection rates of steps of the model

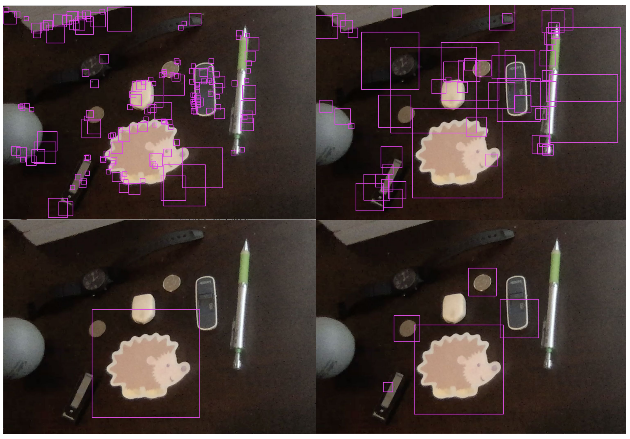


Figure 3: A visualization of the Haar Cascade model of a singular frame for stages 1, 6, 11, and 16. The top left image is stage 1, and proceeds with stages 6, 11, and 16 in a clockwise manner.

to a stage. By comparing these values to the length of the video, a generalized estimate of the accuracies of the models could be calculated.

Looking at our data, the first observation we make is the abnormal jump in accuracy between stages 9 and 10 (see Table 1). The first 10 stages, including 0, are seen to be almost never accurate, while subsequent stages have accuracies in the 60s and 70s. This can be partially accredited to how we calculated our accuracy. We considered a successful detection as a scenario where the only thing that is detected is the hedgehog sticky note. Thus, the detections by the 1st, 6th, and 11th stages of the Haar Cascade are deemed unsuccessful for this frame (see Figure 3). This suggests that despite a visible increase in the aptitude of the training model as stages progress, the accuracy of the model does not increase at a similar rate. Rather, the accuracy only starts to improve upon reaching a critical point, which in this case, is between stages 9 and 10. It is more precise to say that our data shows the relationship between the current stage and the rate of perfect detections.

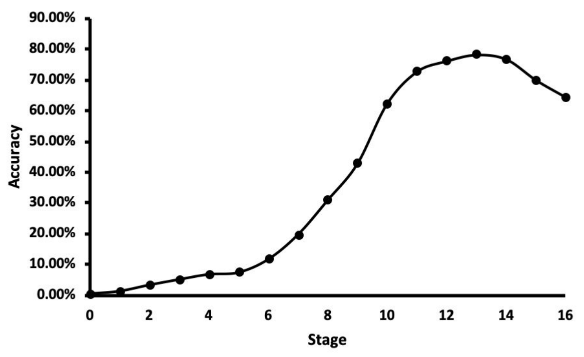
In order to find the relationship between the current stage and the accuracy, we must use a weighted value of a detection rating that ranges from 0 to 1, where 0 means no detection, 1 means perfect detection, and all values in between mean partial detection. With this new algorithm implemented, the detection rating of stages 1, 6, 11, and 16 from Figure 3 become 0.03, 0.20, 0.61, and 1, respectively. While there still exists a relatively steep incline near stage 9, this new weighted algorithm is better at truthfully representing the accuracy of stages 0 through 9 by providing it with a non-zero accuracy.

Another interesting aspect of our data is that after stage 13, the accuracy starts to decrease as the stages progress. Starting at stage 10 with a weighted accuracy of 63.35%, the accuracy peaks at 78.22% (stage 13) and proceeds to fall back down to 64.42% (stage 16). This seems to go against the notion that longer training time means a more accurate detection model, however, there is a plausible explanation. As the stages progress, the Haar Cascade model becomes more and more detailed, thus it is able to detect the object more comfortably. Yet, once the model reaches a certain critical point, the finer details in fact degrade the effectiveness of the model by making it easier to misalign with the original image of the object. The stricter conditions of the model make it harder for the object to be found in a given image or video. Combining our analysis with

	Stage																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Detection Rating Sum	2.25	5.99	14.99	23.04	30.56	33.39	52.52	88.74	139.19	192.69	280.58	327.38	342.90	352.00	345.47	315.00	289.89
Accuracy	0.50%	1.33%	3.33%	5.12%	6.79%	7.42%	11.67%	19.72%	30.93%	42.82%	62.35%	72.75%	76.20%	78.22%	76.77%	70.00%	64.42%

Table 2: Weighted accuracy of steps of the model

the graphical representation of our data (see Graph 1), we can conclude that the accuracy of the Haar Cascade model will increase approximately exponentially until a critical point in which the accuracy starts to diminish.



Graph 1: A visual representation Table 2

5. Conclusion

We have found a relationship between the number of stages trained on a Haar Cascade model and the accuracy of its detection as an approximately exponential increase until a critical point in which accuracy declines. During our search for the relationship, we also found the existence of another critical point that exists on the sudden jump of rate of perfect detection. This paper suggests that since the latter critical point appears before the former, therefore, it must be that there exists an optimal detection stage between the two points. While these findings may only hold true to the Haar Cascade model, nonetheless, it provides us with a baseline understanding that will help us create other optimized AI detection models that deviate from simple 2d detections.

References

[1] Berger, Will. “Deep Learning Haar Cascade Explained.” *Will Berger*, 17 Aug. 2018, www.willberger.org/cascade-haar-explained/.

[2] “Cascade Classifier Training.” *OpenCV*, docs. opencv.org/master/dc/d88/tutorial_traincascade.html.

[3] Gama, João, and Pavel Brazdil. “Cascade Generalization.” *Machine Learning*, vol. 41, no. 3, 2000, pp. 315–343., doi:10.1023/a:1007652114878.

[4] “OpenCV-Object-Detection-Tutorial.” *OpenCV -Object-Detection-Tutorial by JohnAllen*, johnallen. github.io/opencv-object-detection-tutorial/.

[5] sentdex. “Creating Your Own Haar Cascade OpenCV Python Tutorial.” *Python Programming Tutorials*, 11 Jan. 2016, pythonprogramming.net/haar -cascade-object-detection-python-opencv-tutorial/.

[6] villa, eLtronics. “Haar Cascade.” *Medium*, Medium, 5 June 2019, medium.com/@eltronic svilla17/haar- cascade-6c6eb4f0c2bf.

[7] Viola, P., and M. Jones. “Rapid Object Detection Using a Boosted Cascade of Simple Features.” *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, doi:10.1109/cvpr.2001.990517.