Haar and MTCNN: face detection & blurring performance metrics in video and real-time

Matthew Gee
26337031
Gina Cody School of Engineering and Computer Science
Concordia University
Montreal, Quebec
matthew.ryan.gee@gmail.com

Abstract – A Haar-Feature Cascade Classifier Face uses a static kernel to read image data and detect edges; through taking rectangular slices of an image, the classifier finds contrasting dark & light regions to match against a list of predetermined features. The use of "Integral Images" permits rapid feature determination while AdaBoost incentivizes the model to work with only the most-relevant features. The cascade is the process of eliminating regions of images containing low levels of information, allowing the algorithm to focus on areas of interest. The Multi-Task Convolutional Neural Network relies on a dynamic kernel that is determined through training; by using three sequential neural networks – proposal, verification, and refinement - to identify features, remove false-positives, and remove additional false-positives. While both models provide adequate results, their inherent differences lead to variance in false positives and efficiency.

Index Terms – Haar, Cascade, Haar-features, Haar-like features, Integral Image, MTCNN, Multi-Task Convolutional Neural Network, CNN, Machine Learning, Face Detection, Facial Recognition, Computer Vision, Image Processing, AdaBoost

1. Introduction

Facial recognition, like many other simple tasks, is significantly more complex when not-computed by a human brain. After decades spent relegated to the realms of science fiction, it has only been recently that computational facial recognition has been a feasible and practical task. With the advent of digitization and hardware miniaturization, object detection – face detection in particular – has become an increasingly popular field of study in computer science. While algorithms for detecting patterns in images have existed since as early as the 1960's, actively performing these calculations on moving images was impossible until recent advancements in CPU and GPU technology – *central processing units* and g*raphic processing units*, respectively.

Early experiments in facial detection were performed at the Panorama Research Institute in California. At the onset, much of the data was

manually computed, researchers would annotate images with the location of specific features such as the chin, nose, ears, and pupils. The distance between these points would then be calculated and ratios of the distances and direction would be stored. These ratios would then be applied to other images and objects with features matching these rations would be returned to the user.

Generally speaking, facial recognition can be split into four sequential processes. The first step is to determine the region of an image which contains a face so that the resulting steps have less data to process. The second step is to normalize the region containing the face so that its colour, orientation, and general image properties are muted such that the geometric data can be focused on. The third process is determining individual features of a face, such as the eyes, mouth, and edges, so that the final step can compare them as a feature vector against previously collected facial imaging data.

Modern approaches to face detection rely on machine learning to determine features rather than manual data transcription, allowing automation to replace the work that was once performed manually.

While modern implementations of facial detection algorithms are much more accurate and refined, these methods are still not infallible – Haar in particular has difficulty when portions of the face are obscured or when the face is not directly facing the lens. Illumination and glare can result in both the MTCNN and the Haar classifier to fail in face detection as it obscures edges of objects.

3. Methods

A. Motivation & Problem Statement

The goal of this experiment was to implement facial blurring techniques using Gaussian Blur and to measure feasibility of doing this in real-time while broadcasting live video feed. With the rapid adaptation of facial recognition technology, all online images of a party will someday be inseparable from their identity.

B. Haar-Based Classifiers

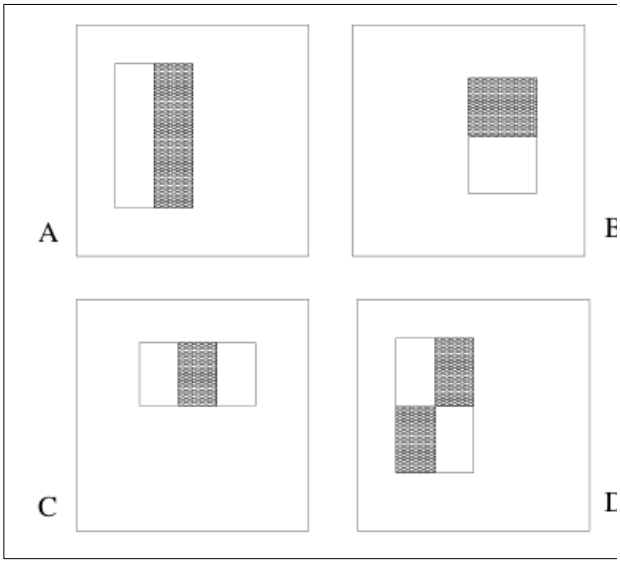

Figure 1: An example of kernels used by the Haar-based feature cascade classifier[2]

The Haar-based feature cascade classifier is a machine learning model using a cascade function. Cascade classifiers are trained on labelled data which is positive (contains the object to be

detected) or negative (absence of object to be detected).

Haar-based classifiers use rectangular features divided into two regions that subtract the sum of pixels of one region from the other. If the value is much greater than zero it indicates an area of interest – however, these kernels result in many false positives, neighbouring regions that differ greatly but are ultimately unimportant. For this reason, cascade classifiers often implement ADAboost, *Adaptive Boosting*, which uses decision trees and weighted sums to make decisions regarding which features detected are relevant. As AdaBoost removes false positive features, eventually the model narrows down the features to a single window which coincides with a face.

$$F_T = \sum_{t=1}^{T} f_t(x)$$

Figure 2: Boost Classifier Equation

C. Convolutional Neural Networks

Neural networks are loosely modelled after human neurons on a computational level. Neural networks consist of nodes that sum the weights of various inputs and then pass on the value to nodes of a different layer if a threshold value is reached. Weights are quantitative values given to features that indicate how relevant they are to prediction, for example having a nose would have a higher weight a scar since a nose is a better indicator of a face. Weights are initially set to random values but altered with time to reflect how relevant they actually are.

The Multi-Task Convolutional Neural Network consists of three sequential neural networks. The first later is a proposal network that attempts to identify objects that are vaguely face-like; however this results in many false positives. These false-positives are handled by the second verification layer which eliminates bounding boxes with low confidence levels. Finally, the third layer, refinement, further removes false positives until only the most-certain objects remain.

D. Gaussian Blur

Gaussian blurring, also known as Gaussian smoothing, is a technique used in noise and detail reduction. It is often used in image processing for the sake of removing extra details from images – since feature detectors are often looking for changes in gradient, having too many sharp edges will result in unnecessary analysis of textured regions. By blurring an image, details are obfuscated and feature-detectors can focus on the lines and contours rather than textures. In this application, Gaussian Blur was simply applied as a means of hiding the face.

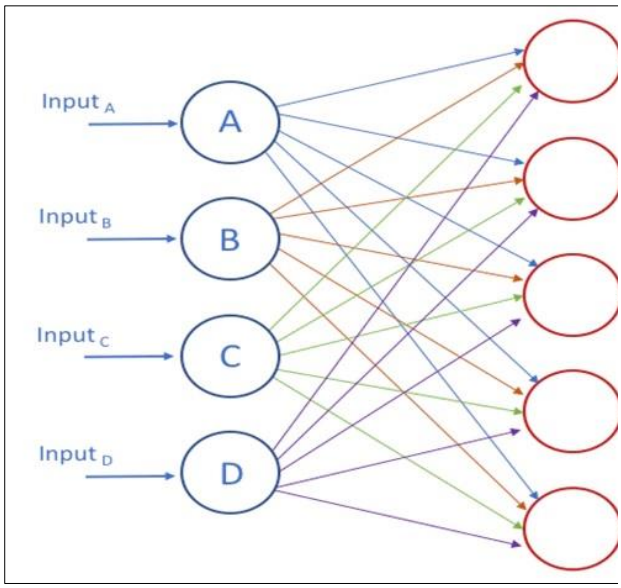

Figure 3: Basic structure of Neural Network [3]

4. Experiments

The MTCNN and Haar-feature cascade classifier were used in three separate experiments. Once on a video and twice on live-streamed webcam footage.

A. Dataset

Analysis was performed by both face detectors on a short video of the author as well as real-time webcam footage of the author. While comparing the results on the same video provides metrics for comparison, the use of a webcam permitted insight into how these detection systems would function in real-life applications.

The Haar-Feature Cascade system was trained on the haarcascade_frontalface_default.xml dataset while the MTCNN was trained on the facenet_keras.h5 dataset. The Haar Cascade dataset is made available through OpenCV while the facenet dataset was

B. Baselines

The video used in this experiment had a runtime of 0.62 milliseconds and contained 69 frames. Any runtime longer than 0.62 indicates that the model applied causes slowdown due to

computational complexity and any number of faces not equal to 69 indicates the presence of false negatives or false positives.

The webcam operates at 30 frames per seconds when unencumbered by computations so any FPS value less than 30 while running the models indicates slowdown.

C. Implementation

Experiments were performed on a Acer Swift SF315-51G, Windows 10 64bit workstation running on an Intel Corei5 825OU: Kaby Lake-R processor with 4 cores, 32 processing threads, 1.60 GHz specification, 8GB RAM, and an NVIDIA GeForce MX150 graphics card. The program was written in Python 3.6.8 using the PyCharm 2020.3.2 IDE with libraries provided by OpenCV 4.5.1.48, Tensorflow 2.4.1, Keras 2.4.3, Pillow 8.2, numpy 1.19.5, and MTCNN.

Table 1: Video Performance Metrics

	MTCNN	Haar
Frame Rate	1.37 FPS	11.14FPS
Recall	1.00	1.0
Precision	1.00	0.945
Playback	50.2 s	6.5

D. Evaluation Metrics

The success of each model is determined by the number of faces detected as well as the rate of slowdown applied to the media as the computations are performed. As a face is present in every frame of the test video, fewer faces than frames indicates a lower success rate and more faces than frames indicates the presence of false positives in the data.

The accuracy of each model can be measured using precision and recall. Precision is the amount of relevant items selected of all items selected while recall is the amount of found-relevant items as a percentage of all relevant items.

The slowdown was measured by computing how long it took to process the video file in comparison to the duration of the video when no calculations are performed.

E. Image Segmentation Performance in Video

The MTCNN found 69 faces in the 69 frames, indicating a precision and recall of 100%. Meanwhile, the Haar Classifier obtained 73 faces in the 69 frames – indicating the presence of false-positives – resulting in 100% recall but 94.5% precision. The video used for comparison in this experiment has a duration of 62 milliseconds when played without computing faces.

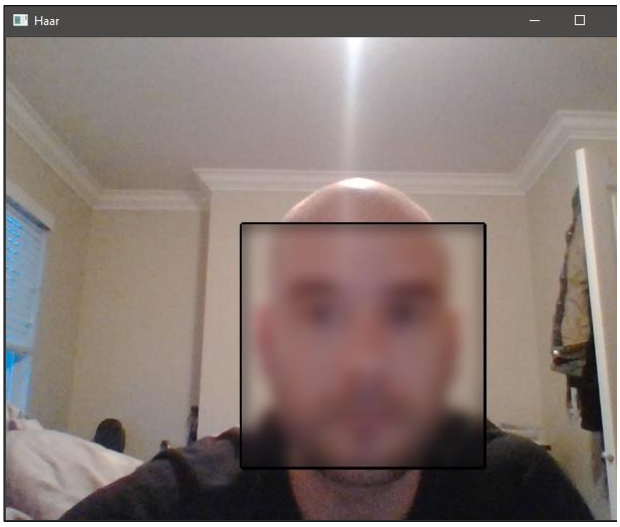

Figure 4: Results obtained from the Haar-based feature cascade classifier

While performing face detection with MTCNN and Haar the playback lasted 50.2 seconds and 6.5

milliseconds respectively. The MTCNN classifier increased videoplay back by a factor of 80.96 while the Haar Classifieronly increased playback by a factor of 1.04. The MTCNN played the video at 1.37 frames per second while the Haar classifier did the same at a rate of 11.14 frames per second. These results indicate that using the MTCNN model on live-footage is not feasible with the hardware available to the author.

F. Image Segmentation Performance in Real-Time

The MTCNN found 5 faces in the 6 frame webcam experiment, and 24 faces in the 30 frame webcam experiment. Thus the MTCNN had a recall of 83% and 80%, and a precision of 100% for both trials. Meanwhile, the Haar Classifier obtained 72 faces in the 73 frames and 225 faces in 210 frames – once again indicating the presence of false-positives. The Haar classifier had a precision of 98.6% and 93.3%, and recall of 98.6% and 100% for each of the two trials.

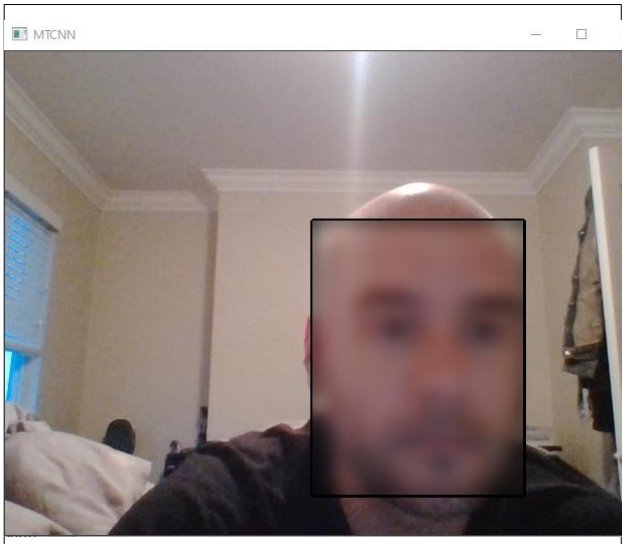

Figure 5: Results obtained from the MTCNN classifier

While performing face detection with MTCC, the frames per second rates were 1.28 and 1.22 FPS. While performing the face detection with the Haar classifier the frames per second rates were 18.3 and 17.1 FPS.

Table 2: Webcam1 Performance Metrics

	MTCNN	Haar
Frame Rate	1.28 FPS	18.3 FPS
Recall	0.83	0.986
Precision	1.00	0.986

5. Conclusion

While both the Haar Cascade Classifier and Multi-Task Convolutional Neural Network provide adequate results, their use cases are dissimilar. As the Haar Cascade classifier is less computationally complex, it is much better suited to real-time video capture. The Haar classifier still results in frame-rate deceleration butwith adequate hardware this could be minimized so to perform fast enough to be applied on live footage – this is not the case for the MTCNN.

The MTCNN, while obtaining more accurate results, was sufficiently complex that video and webcam footage ran at 1/7th of the speed it should have. While its results were better, it is

simply unfeasible to apply this to live video because it takes significantly too much computational complexity to run in parallel. Thus it is my recommendation that for live footage on uses the Haar classifier but the MTCNN for photographic databases as it has better precision.

Table 3: Webcam 2 Performance Metrics

	MTCNN	Haar
Frame Rate	1.22 FPS	17.1 FPS
Recall	0.80	1.00
Precision	1.00	0.930

6. Acknowledgements

Thanks to the professors and teaching assistants associated with Comp-478 for making this material so accessible.

Also, thanks to Nick for persuading me not to drop this class and avoiding that depression.

Also, thanks to the marker for going easy on me – I know this ain't great but hey, *at least I tried.* ^__^;

6. References

[1] Kaipeng Zhang et al, “Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks”, IEEE Signal Processing Letters, 23(10), 2016

[2] Viola and Jones, "Rapid object detection using a boosted cascade of simple features", Computer Vision and Pattern Recognition, 2001

[3] Brendan Tierney, “Understanding, Building, and Using Neural Network Machine Learning Models using Oracle 18C”, Oracle Ace, 2021, *https://developer.oracle.com/databases/neural-network-machine-learning.html*

[4] Jeremy Norman, “Woodrow Beldsoe Originates of Automated Facial Recognition”, 2021, historyofinformation.com

[5] Wassa Team, “Modern ace Detection based on Deep Learning using Python and Mxnet”, Medium, 2017

[6] Jason Brownless, “How to Develop a Face Recognition System Using FaceNet in Keras”, 2019, Machine Learning Mastery

[7] David Sandberg, “MTCNN FaceNet face Detection and Alignment”, https://github.com/davidsandberg/facenet/tree/master/src/align