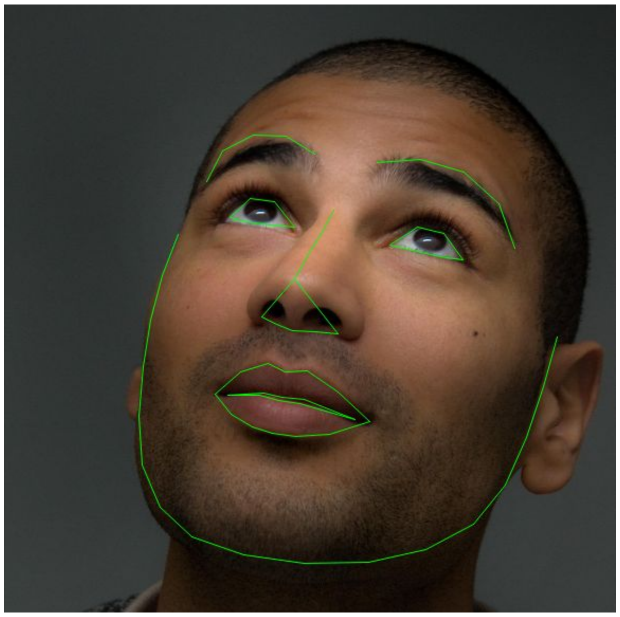
# FACE DETECTION BENCHMARK TESTING USING OPENCV.JS

*COMPILING OPENCV.JS WITH AND WITHOUT WEB_ASSEMBLY OPTION*



**SHAYANG ZANG, XIAOYAN QU**

12.13.2017
FALL 2017 CS243 HIGH PERFORMANCE ARCHITECTURE

## PROJECT CODEBASE

https://github.com/xiaoyanqu/opencv_face_detection

## INTRODUCTION

Emscripten is an LLVM-to-JavaScript compiler. It takes LLVM bitcode - which can be generated from C/C++ using clang, and compiles that into asm.js or WebAssembly that can execute directly inside the web browsers.

Asm.js is a highly optimizable, low-level subset of JavaScript. Asm.js enables ahead-of-time compilation and optimization in JavaScript engine that provide near-to-native execution speed.

WebAssembly is a new portable, size- and load-time-efficient binary format suitable for compilation to the web, which aims to execute at native speed. WebAssembly is currently being designed as an open standard by W3C.OpenCV.js is a JavaScript binding for selected subset of OpenCV functions for the web platform. It allows emerging web applications with multimedia processing to benefit from the wide variety of vision functions available in OpenCV.

OpenCV.js leverages Emscripten to compile OpenCV functions into asm.js or WebAssembly targets, and provides a JavaScript APIs for web application to access them. In this project, we were using OpenCV 3.3.1-dev version. The future versions of the library will take advantage of acceleration APIs that are available on the Web such as SIMD and multi-threaded execution.

In this quarter, we've been working on a face detection benchmarking project using compiled OpenCV.js with two different versions, i.e. Asm.js and WebAssembly. We anticipated there would be some improvement using WebAssembly, and compared the performance with the regular Asm.js option.

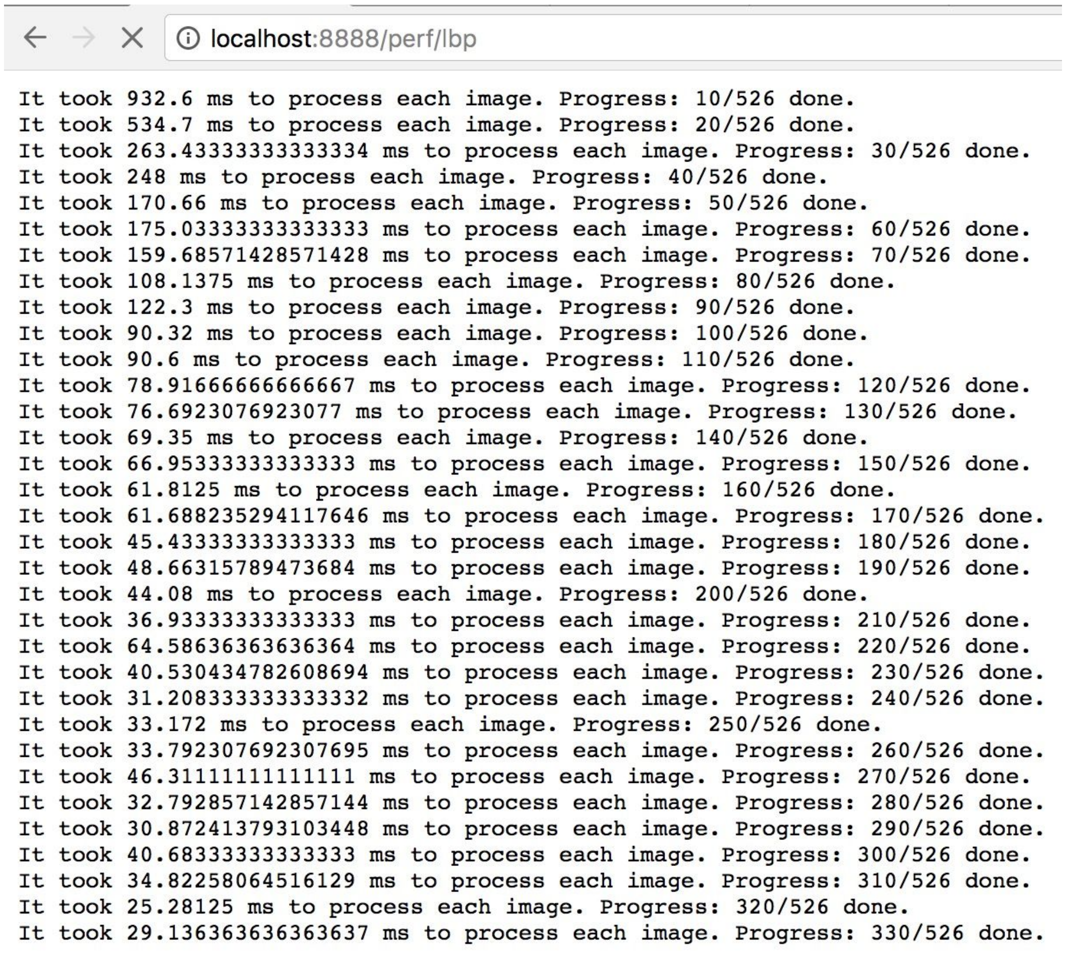
## METHODOLOGY & MATERIALS

1. Classifiers for face detection: LBP and HAAR
2. Image source: WIDER FACE: A Face Detection Benchmark by Multimedia Laboratory, Department of Information Engineering, The Chinese University of Hong Kong
3. Performance metrics: the average processing time for each image in batch size of

ten (10) images out of total of 526 images.

## RESULTS

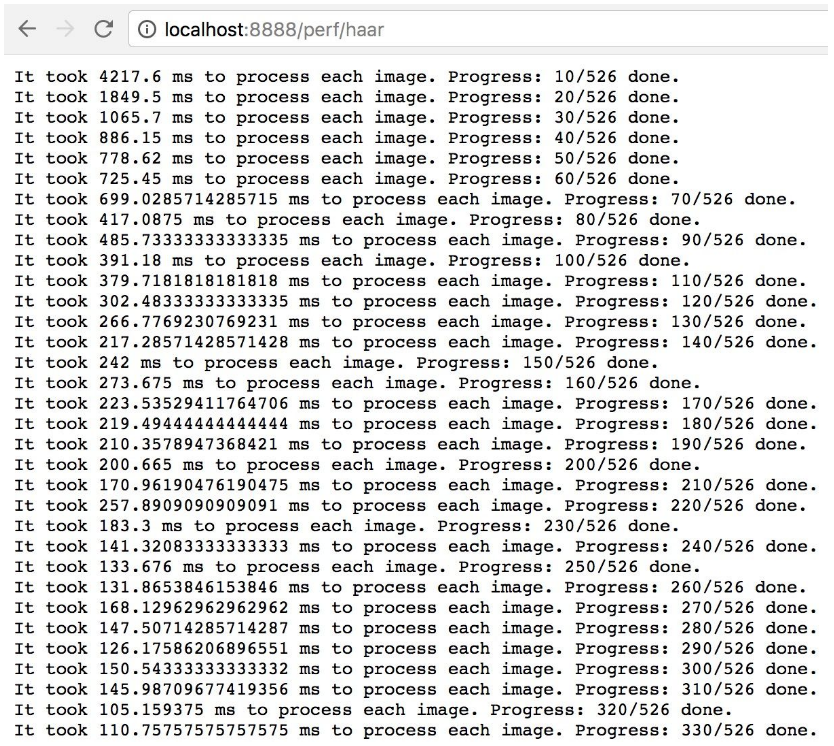
The following four figures are the screenshots of our results when processing the images.



localhost:8888/perf/lbp

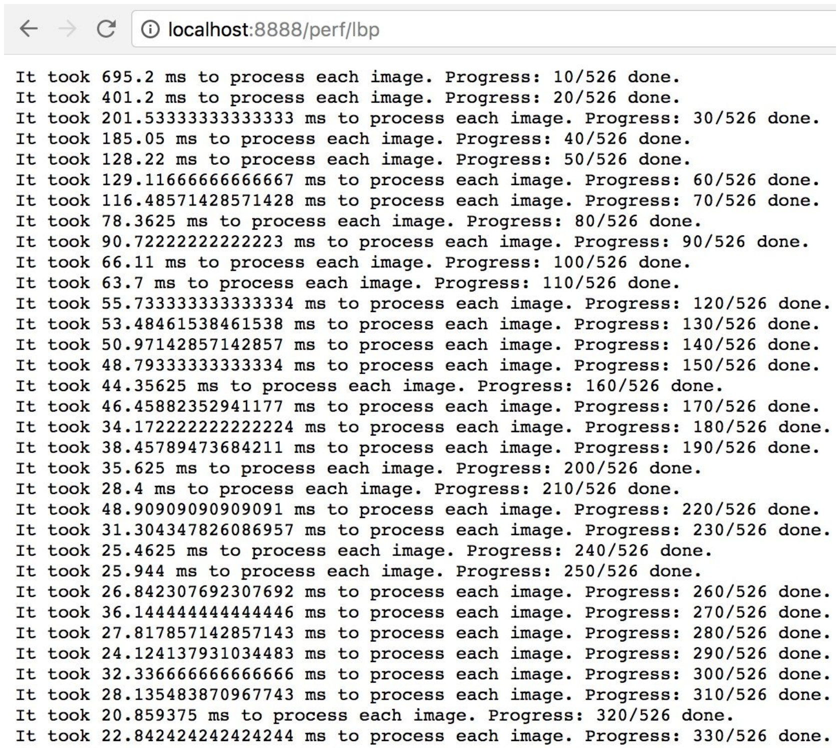
```
It took 932.6 ms to process each image. Progress: 10/526 done.
It took 534.7 ms to process each image. Progress: 20/526 done.
It took 263.43333333333334 ms to process each image. Progress: 30/526 done.
It took 248 ms to process each image. Progress: 40/526 done.
It took 170.66 ms to process each image. Progress: 50/526 done.
It took 175.03333333333333 ms to process each image. Progress: 60/526 done.
It took 159.68571428571428 ms to process each image. Progress: 70/526 done.
It took 108.1375 ms to process each image. Progress: 80/526 done.
It took 122.3 ms to process each image. Progress: 90/526 done.
It took 90.32 ms to process each image. Progress: 100/526 done.
It took 90.6 ms to process each image. Progress: 110/526 done.
It took 78.91666666666667 ms to process each image. Progress: 120/526 done.
It took 76.6923076923077 ms to process each image. Progress: 130/526 done.
It took 69.35 ms to process each image. Progress: 140/526 done.
It took 66.95333333333333 ms to process each image. Progress: 150/526 done.
It took 61.8125 ms to process each image. Progress: 160/526 done.
It took 61.688235294117646 ms to process each image. Progress: 170/526 done.
It took 45.43333333333333 ms to process each image. Progress: 180/526 done.
It took 48.66315789473684 ms to process each image. Progress: 190/526 done.
It took 44.08 ms to process each image. Progress: 200/526 done.
It took 36.93333333333333 ms to process each image. Progress: 210/526 done.
It took 64.58636363636364 ms to process each image. Progress: 220/526 done.
It took 40.530434782608694 ms to process each image. Progress: 230/526 done.
It took 31.208333333333332 ms to process each image. Progress: 240/526 done.
It took 33.172 ms to process each image. Progress: 250/526 done.
It took 33.792307692307695 ms to process each image. Progress: 260/526 done.
It took 46.31111111111111 ms to process each image. Progress: 270/526 done.
It took 32.792857142857144 ms to process each image. Progress: 280/526 done.
It took 30.872413793103448 ms to process each image. Progress: 290/526 done.
It took 40.68333333333333 ms to process each image. Progress: 300/526 done.
It took 34.82258064516129 ms to process each image. Progress: 310/526 done.
It took 25.28125 ms to process each image. Progress: 320/526 done.
It took 29.136363636363637 ms to process each image. Progress: 330/526 done.
```

Figure 1 - Processing Performance with LBP Classifier and Asm.js



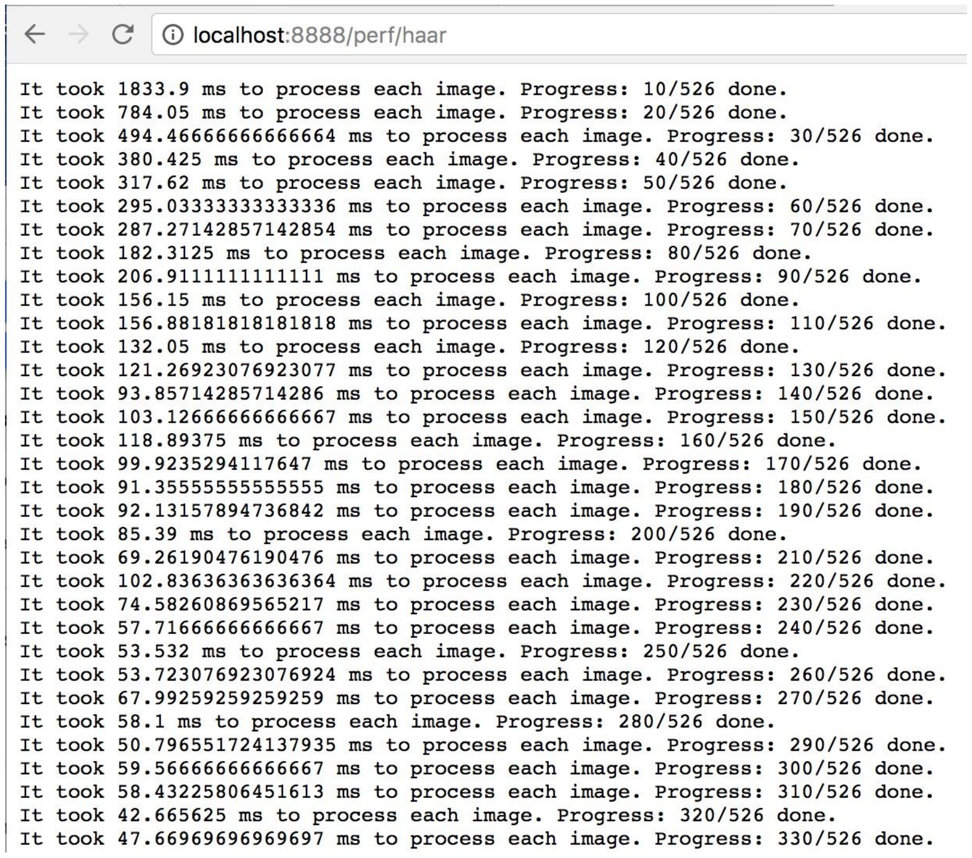
```
It took 4217.6 ms to process each image. Progress: 10/526 done.
It took 1849.5 ms to process each image. Progress: 20/526 done.
It took 1065.7 ms to process each image. Progress: 30/526 done.
It took 886.15 ms to process each image. Progress: 40/526 done.
It took 778.62 ms to process each image. Progress: 50/526 done.
It took 725.45 ms to process each image. Progress: 60/526 done.
It took 699.0285714285715 ms to process each image. Progress: 70/526 done.
It took 417.0875 ms to process each image. Progress: 80/526 done.
It took 485.73333333333335 ms to process each image. Progress: 90/526 done.
It took 391.18 ms to process each image. Progress: 100/526 done.
It took 379.7181818181818 ms to process each image. Progress: 110/526 done.
It took 302.48333333333335 ms to process each image. Progress: 120/526 done.
It took 266.7769230769231 ms to process each image. Progress: 130/526 done.
It took 217.28571428571428 ms to process each image. Progress: 140/526 done.
It took 242 ms to process each image. Progress: 150/526 done.
It took 273.675 ms to process each image. Progress: 160/526 done.
It took 223.53529411764706 ms to process each image. Progress: 170/526 done.
It took 219.49444444444444 ms to process each image. Progress: 180/526 done.
It took 210.3578947368421 ms to process each image. Progress: 190/526 done.
It took 200.665 ms to process each image. Progress: 200/526 done.
It took 170.96190476190475 ms to process each image. Progress: 210/526 done.
It took 257.8909090909091 ms to process each image. Progress: 220/526 done.
It took 183.3 ms to process each image. Progress: 230/526 done.
It took 141.32083333333333 ms to process each image. Progress: 240/526 done.
It took 133.676 ms to process each image. Progress: 250/526 done.
It took 131.8653846153846 ms to process each image. Progress: 260/526 done.
It took 168.12962962962962 ms to process each image. Progress: 270/526 done.
It took 147.50714285714287 ms to process each image. Progress: 280/526 done.
It took 126.17586206896551 ms to process each image. Progress: 290/526 done.
It took 150.54333333333332 ms to process each image. Progress: 300/526 done.
It took 145.98709677419356 ms to process each image. Progress: 310/526 done.
It took 105.159375 ms to process each image. Progress: 320/526 done.
It took 110.75757575757575 ms to process each image. Progress: 330/526 done.
```

Figure 2 - Processing Performance with HAAR Classifier and Asm.js



```
It took 695.2 ms to process each image. Progress: 10/526 done.
It took 401.2 ms to process each image. Progress: 20/526 done.
It took 201.53333333333333 ms to process each image. Progress: 30/526 done.
It took 185.05 ms to process each image. Progress: 40/526 done.
It took 128.22 ms to process each image. Progress: 50/526 done.
It took 129.11666666666667 ms to process each image. Progress: 60/526 done.
It took 116.48571428571428 ms to process each image. Progress: 70/526 done.
It took 78.3625 ms to process each image. Progress: 80/526 done.
It took 90.72222222222223 ms to process each image. Progress: 90/526 done.
It took 66.11 ms to process each image. Progress: 100/526 done.
It took 63.7 ms to process each image. Progress: 110/526 done.
It took 55.733333333333334 ms to process each image. Progress: 120/526 done.
It took 53.48461538461538 ms to process each image. Progress: 130/526 done.
It took 50.97142857142857 ms to process each image. Progress: 140/526 done.
It took 48.79333333333334 ms to process each image. Progress: 150/526 done.
It took 44.35625 ms to process each image. Progress: 160/526 done.
It took 46.45882352941177 ms to process each image. Progress: 170/526 done.
It took 34.172222222222224 ms to process each image. Progress: 180/526 done.
It took 38.45789473684211 ms to process each image. Progress: 190/526 done.
It took 35.625 ms to process each image. Progress: 200/526 done.
It took 28.4 ms to process each image. Progress: 210/526 done.
It took 48.90909090909091 ms to process each image. Progress: 220/526 done.
It took 31.304347826086957 ms to process each image. Progress: 230/526 done.
It took 25.4625 ms to process each image. Progress: 240/526 done.
It took 25.944 ms to process each image. Progress: 250/526 done.
It took 26.842307692307692 ms to process each image. Progress: 260/526 done.
It took 36.144444444444446 ms to process each image. Progress: 270/526 done.
It took 27.817857142857143 ms to process each image. Progress: 280/526 done.
It took 24.124137931034483 ms to process each image. Progress: 290/526 done.
It took 32.336666666666666 ms to process each image. Progress: 300/526 done.
It took 28.135483870967743 ms to process each image. Progress: 310/526 done.
It took 20.859375 ms to process each image. Progress: 320/526 done.
It took 22.842424242424244 ms to process each image. Progress: 330/526 done.
```

Figure 3 - Processing Performance with LBP Classifier and WebAssembly



```
It took 1833.9 ms to process each image. Progress: 10/526 done.
It took 784.05 ms to process each image. Progress: 20/526 done.
It took 494.46666666666664 ms to process each image. Progress: 30/526 done.
It took 380.425 ms to process each image. Progress: 40/526 done.
It took 317.62 ms to process each image. Progress: 50/526 done.
It took 295.03333333333336 ms to process each image. Progress: 60/526 done.
It took 287.27142857142854 ms to process each image. Progress: 70/526 done.
It took 182.3125 ms to process each image. Progress: 80/526 done.
It took 206.91111111111111 ms to process each image. Progress: 90/526 done.
It took 156.15 ms to process each image. Progress: 100/526 done.
It took 156.88181818181818 ms to process each image. Progress: 110/526 done.
It took 132.05 ms to process each image. Progress: 120/526 done.
It took 121.26923076923077 ms to process each image. Progress: 130/526 done.
It took 93.85714285714286 ms to process each image. Progress: 140/526 done.
It took 103.12666666666667 ms to process each image. Progress: 150/526 done.
It took 118.89375 ms to process each image. Progress: 160/526 done.
It took 99.9235294117647 ms to process each image. Progress: 170/526 done.
It took 91.35555555555555 ms to process each image. Progress: 180/526 done.
It took 92.13157894736842 ms to process each image. Progress: 190/526 done.
It took 85.39 ms to process each image. Progress: 200/526 done.
It took 69.26190476190476 ms to process each image. Progress: 210/526 done.
It took 102.83636363636364 ms to process each image. Progress: 220/526 done.
It took 74.58260869565217 ms to process each image. Progress: 230/526 done.
It took 57.71666666666667 ms to process each image. Progress: 240/526 done.
It took 53.532 ms to process each image. Progress: 250/526 done.
It took 53.723076923076924 ms to process each image. Progress: 260/526 done.
It took 67.99259259259259 ms to process each image. Progress: 270/526 done.
It took 58.1 ms to process each image. Progress: 280/526 done.
It took 50.796551724137935 ms to process each image. Progress: 290/526 done.
It took 59.56666666666667 ms to process each image. Progress: 300/526 done.
It took 58.43225806451613 ms to process each image. Progress: 310/526 done.
It took 42.665625 ms to process each image. Progress: 320/526 done.
It took 47.66969696969697 ms to process each image. Progress: 330/526 done.
```

Figure 4 - Processing Performance with HAAR Classifier and WebAssembly

## CONCLUSION

Based on the data we obtained from running the benchmarking tests, we found that webassembly built outperforms the regular asm.js built for both the LBP and the HAAR classifier. Please see Figure 5 - 8 the performance charts of our findings. All the code and detailed usage instructions can be found at:
https://github.com/xiaoyanqu/opencv_face_detection

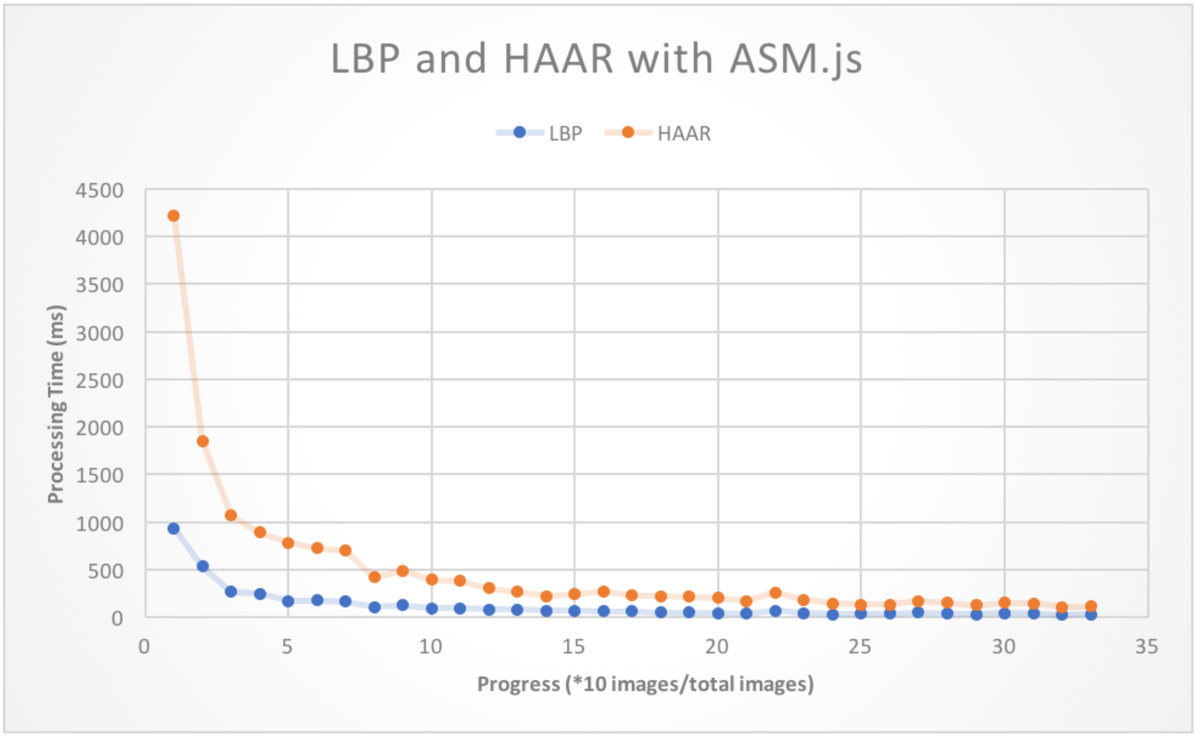


Figure 5 - Performance Chart with LBP and HAAR Classifier using ASM.js

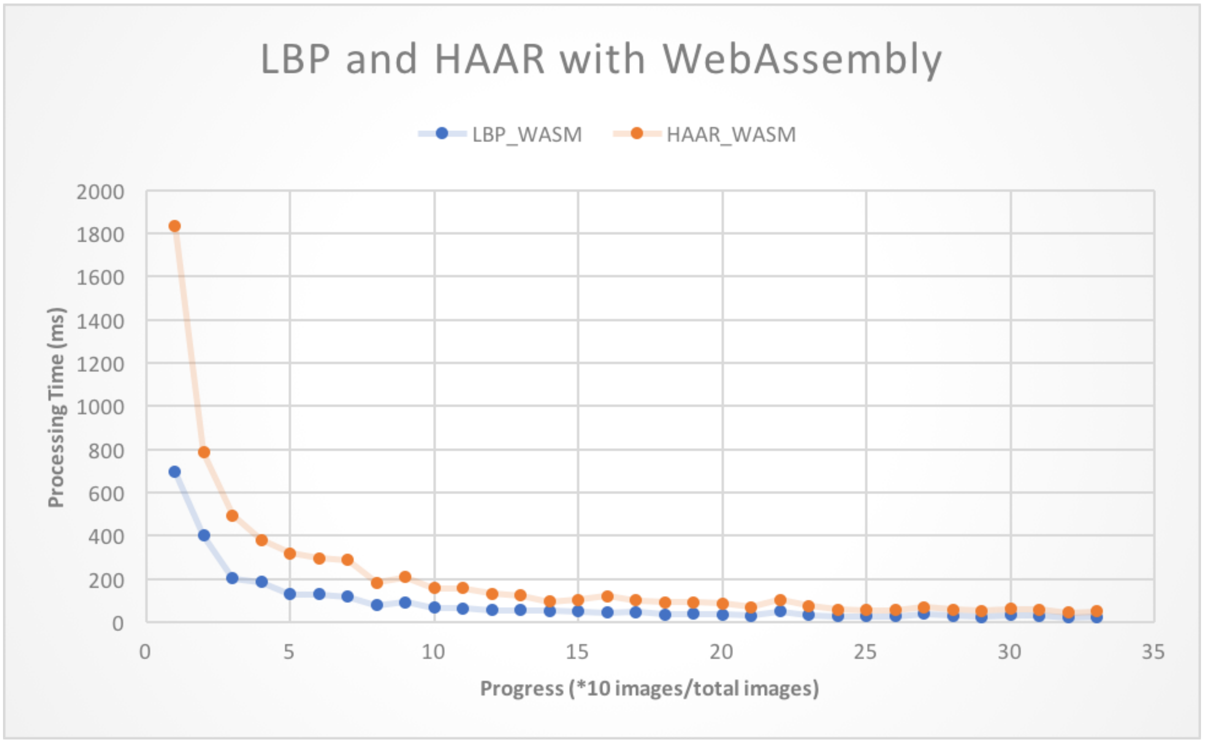


Figure 6 - Performance Chart with LBP and HAAR Classifier using WebAssembly

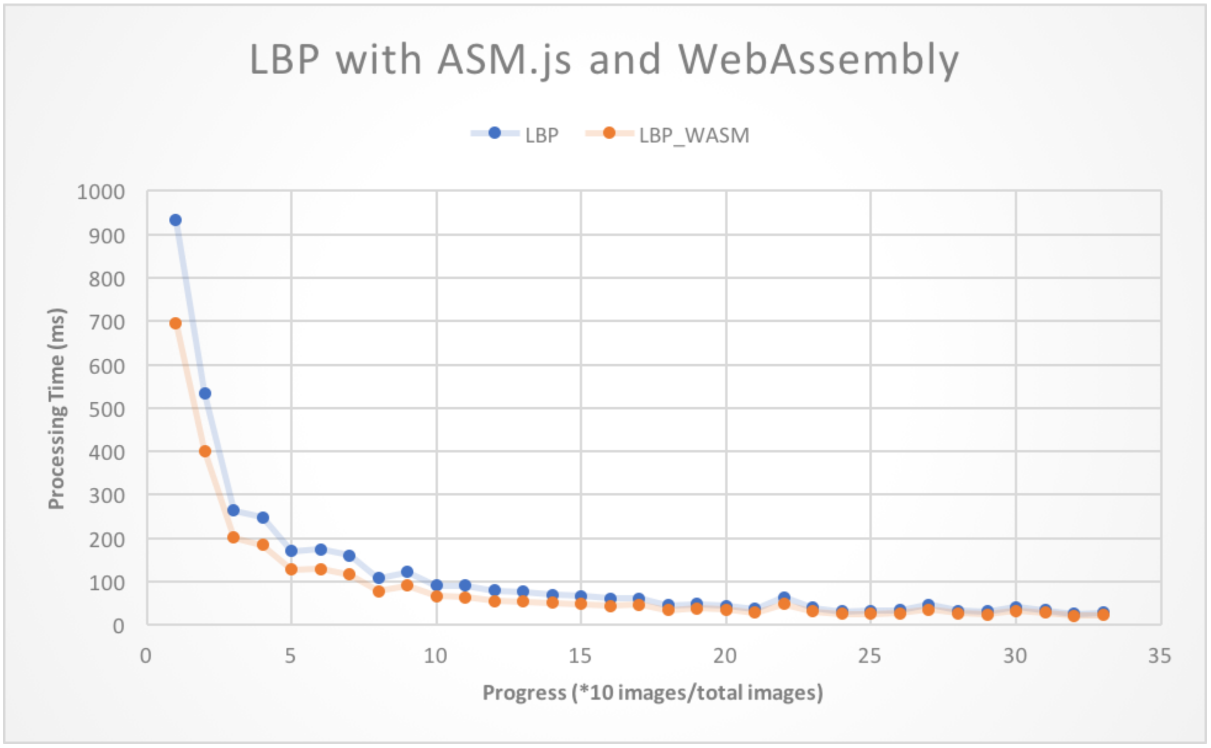


Figure 7 - Performance Chart with LBP Classifier between ASM.js and WebAssembly

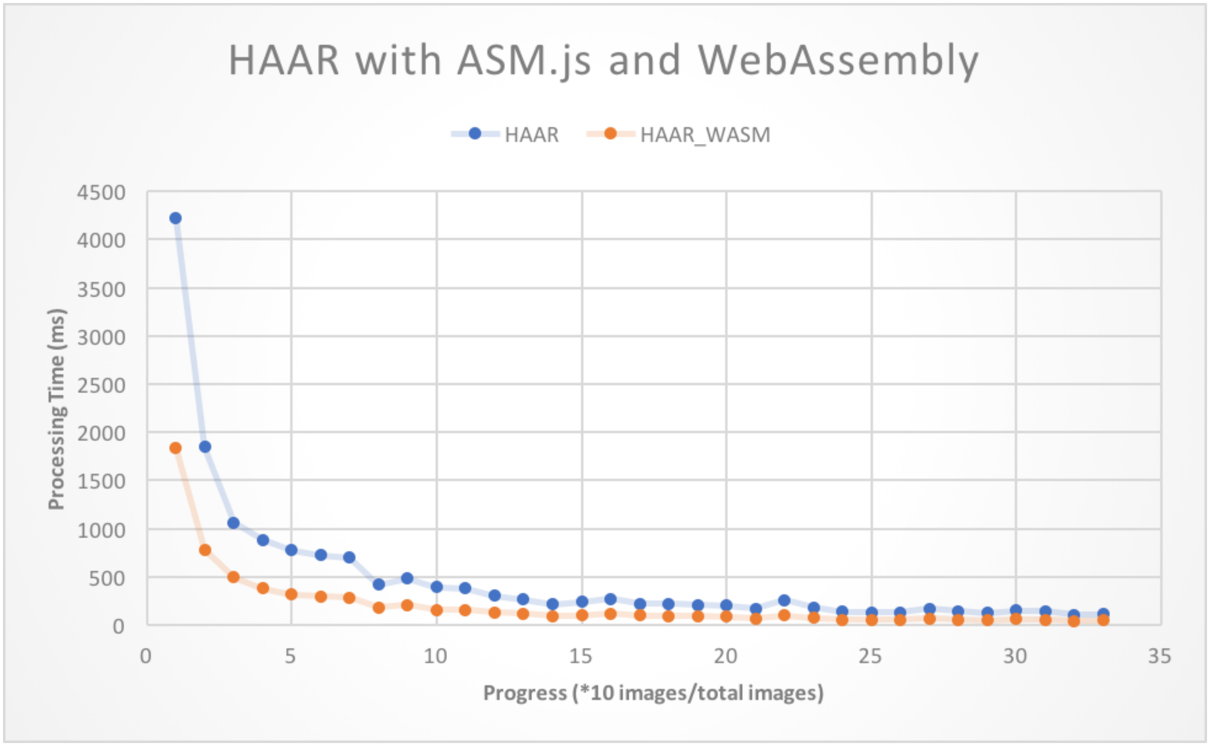


Figure 8 - Performance Chart with HAAR Classifier between ASM.js and WebAssembly

## REFERENCES

1. https://docs.opencv.org/master/df/df7/tutorial_js_table_of_contents_setup.html
2. http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/index.html
3. https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Yang_WIDER_FACE_A_CVPR_2016_paper.pdf

## IMAGE SOURCE:

1. Cover 1 (left): en.wikipedia.org

2. Cover 1 (right): blog.dlib.net

## ACKNOWLEDGEMENTS:

Thanks **Sajjad Taheri** for introducing us this great project, and his continuous professional support during the project!

Thanks **Prof. Alexandru Nicolau** for the guidance of the theories of high performance architecture and managing this course!