W251 Homework - Blair Jones

Questions

- Describe your solution in detail. What neural network did you use? What dataset was it trained on? What accuracy does it achieve?
- Does it achieve reasonable accuracy in your empirical tests? Would you use this solution to develop a robust, production-grade system?
- What framerate does this method achieve on the Jetson? Where is the bottleneck?
- Which is a better quality detector: the OpenCV or the neural one?

Responses

For the base solution I simply re-used the code from HW3. To save time, I did not run the full end-to-end code including AWS. I just stored the output images to local storage.

I used Facenet for the neural network as suggested by the homework guide. It was extremely easy to integrate into the face detection client app that I had written for HW3. I merely had to import a couple of additional libraries and swap out a couple of lines of code. Very easy.

Accuracy was very high, consistently near 100%. I pointed the camera at a picture with multiple faces and it detected and saved all faces in sub-second time. See Figures 1 and 2 below.

Perceived runtime was fast. I did not measure the performance. I would say that Facenet was on par with HW3 using OpenCV. Both OpenCV and Facenet detected faces very quickly.

For a production system I would choose the solution most appropriate to meet several criteria:

- Dependability (is it prone to crash)
- Size (memory, cpu)
- Inference Speed
- Accuracy
- Flexibility (how well does it work on rotated / partial view) - here Facenet should perform much better

The choice would depend on the requirements.

It is unfortunate that both MTCNN and Facenet use the same name for the model, MTCNN. This is confusing because they are two different solutions with drastically different performance, according to research.

Figure 1: Facenet accuracy

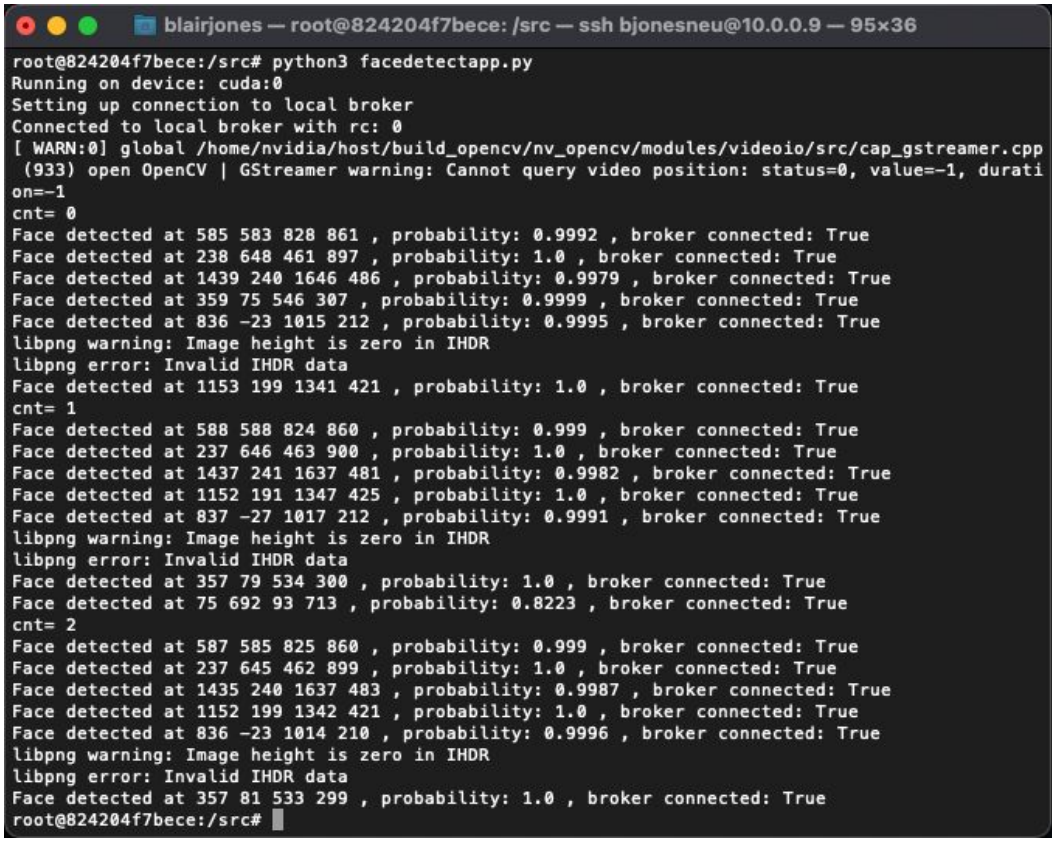

Figure 2: Detected faces