

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

```
blairjones — root@824204f7bece: /src — ssh bjonesneu@10.0.0.9 — 95x36
root@824204f7bece:/src# python3 facedetectapp.py
Running on device: cuda:0
Setting up connection to local broker
Connected to local broker with rc: 0
[ WARN:0] global /home/nvidia/host/build_opencv/nv_opencv/modules/videoio/src/cap_gstreamer.cpp
(933) open OpenCV | GStreamer warning: Cannot query video position: status=0, value=-1, durati
on=-1
cnt= 0
Face detected at 585 583 828 861 , probability: 0.9992 , broker connected: True
Face detected at 238 648 461 897 , probability: 1.0 , broker connected: True
Face detected at 1439 240 1646 486 , probability: 0.9979 , broker connected: True
Face detected at 359 75 546 307 , probability: 0.9999 , broker connected: True
Face detected at 836 -23 1015 212 , probability: 0.9995 , broker connected: True
libpng warning: Image height is zero in IHDR
libpng error: Invalid IHDR data
Face detected at 1153 199 1341 421 , probability: 1.0 , broker connected: True
cnt= 1
Face detected at 588 588 824 860 , probability: 0.999 , broker connected: True
Face detected at 237 646 463 900 , probability: 1.0 , broker connected: True
Face detected at 1437 241 1637 481 , probability: 0.9982 , broker connected: True
Face detected at 1152 191 1347 425 , probability: 1.0 , broker connected: True
Face detected at 837 -27 1017 212 , probability: 0.9991 , broker connected: True
libpng warning: Image height is zero in IHDR
libpng error: Invalid IHDR data
Face detected at 357 79 534 300 , probability: 1.0 , broker connected: True
Face detected at 75 692 93 713 , probability: 0.8223 , broker connected: True
cnt= 2
Face detected at 587 585 825 860 , probability: 0.999 , broker connected: True
Face detected at 237 645 462 899 , probability: 1.0 , broker connected: True
Face detected at 1435 240 1637 483 , probability: 0.9987 , broker connected: True
Face detected at 1152 199 1342 421 , probability: 1.0 , broker connected: True
Face detected at 836 -23 1014 210 , probability: 0.9996 , broker connected: True
libpng warning: Image height is zero in IHDR
libpng error: Invalid IHDR data
Face detected at 357 81 533 299 , probability: 1.0 , broker connected: True
root@824204f7bece:/src#
```

Figure 2: Detected faces

