1. Introduction

Our team focus on the mask detection application. In China, there are a lot of places still restrict people to wearing masks. Many places will also use a lot of manpower and resources to control people wearing masks at any place. We want to achieve a real time mask detection camera to detect whether the people wear a mask or not. It will save a lot of resource. Therefore, we want to build up this application.

2. The process

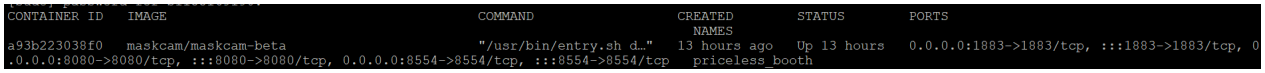
2.1 The failure case:

At first we use the guideline from: https://collabnix.com/building-a-real-time-crowd-face-mask-detection-system-on-nvidia-jetson-nano/

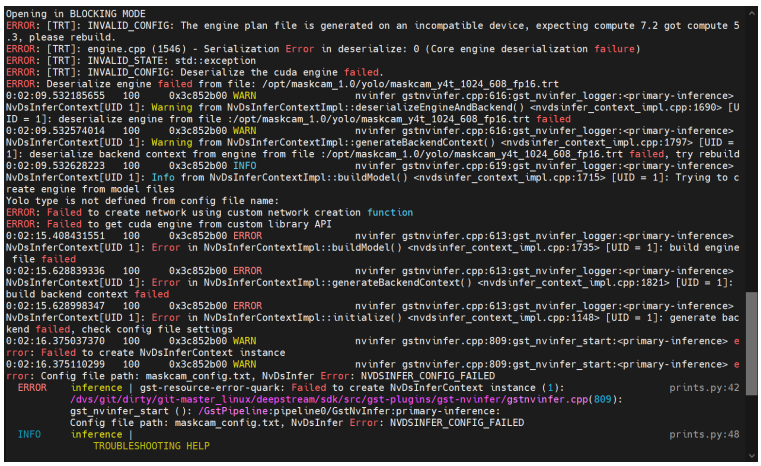
After we download the image from the guideline, log in as the developer.

sudo docker pull maskcam/maskcam-beta

docker run --runtime nvidia --privileged --rm -it --env DEV_MODE=1 -p 1883:1883 -p 8080:8080 -p 8554:8554 maskcam/maskcam-beta



And then we try to run the application. However, we cannot run it right. There is a error happen.



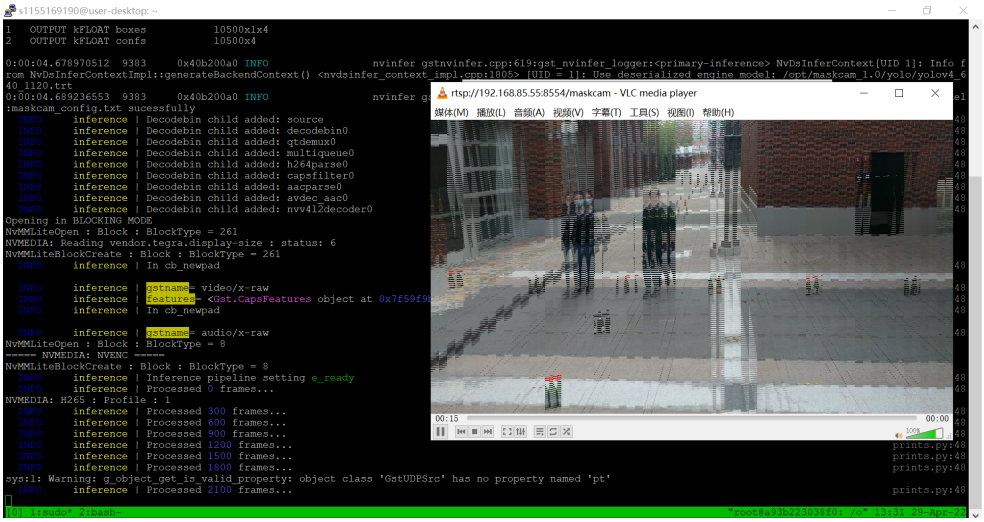
It shows that the GPU computing power is not match as the original model. Because this model is run through the trt file, our GPU computing power is not the same as the author. Therefore, we try to rebuild the darknet to create the onnx file. And using the onnx file to create the trt file.

PYTHONPATH='pytorch-YOLOv4:$PYTHONPATH' python3 pytorch-

YOLOv4/tool/darknet2onnx.py yolo/facemask-yolov4-tiny.cfg yolo/facemask-yolov4-tiny_best.weights --batch_size=1

Finally, we run the model through: *python3 maskcam_run.py file:////opt/maskcam_1.0/videos/inputs/bdti_exterior.mp4*

We get the result:



The advantage of this model is:

- It can detect the mask

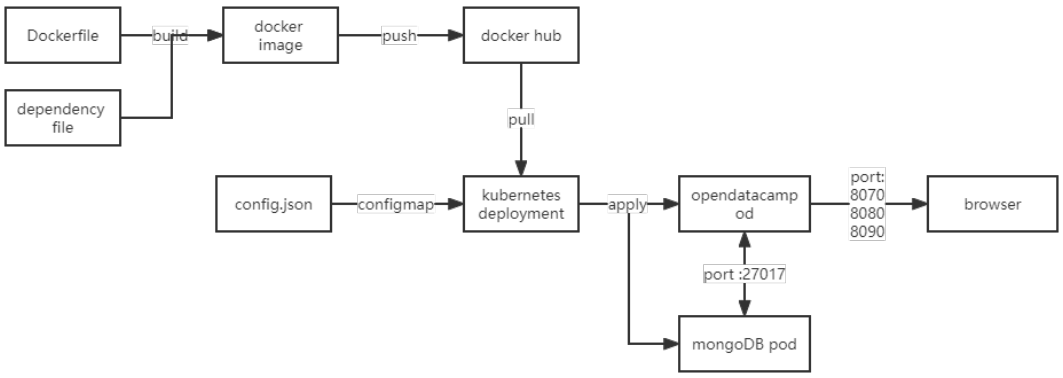
The disadvantages of this model are

- Image ghosting
- Cannot watch in a clearly way

Therefore, we want to improve this model:

2.2 Successful case

We do some improvement in the previous model:



2.2.1 First of all: we edit the config.json file:

```
NEURAL_NETWORK_PARAMS": {
    "yolov4": {
        "data": "mask/mask.data",
        "cfg": "mask/yolov4-tiny-masks.cfg",
        "weights": "mask/yolov4-tiny-mask.weights"      },
    "yolov4-tiny": {
        "data": "mask/mask.data",
        "cfg": "mask/yolov4-tiny-masks.cfg",
        "weights": "mask/yolov4-tiny-mask.weights"      }
 }
```

2.2.2 After that, we build up the new docker image- Dockerfile and upload this image to the Docker Hub.

```
FROM opendatacam/opendatacam:v3.0.2-xavier
WORKDIR /var/local/darknet
RUN mkdir mask
COPY mask/mask.data mask/
COPY mask/mask.names mask/
COPY mask/yolov4-tiny-mask.weights mask/
COPY mask/yolov4-tiny-masks.cfg mask/
COPY demo1.mp4 /var/local/darknet/opendatacam_videos/
COPY demo2.mp4 /var/local/darknet/opendatacam_videos/
# Include demo.mp4 file
WORKDIR /var/local/opendatacam
COPY config.json config.json
CMD ["./launch.sh"]
```

2.2.3 After that, we build the new docker image through downloading the image from the Docker Hub which is provided by us previous.

```
sudo docker build -t mask_detection:v2
sudo docker tag mask_detection :v2 raphaelsun/mask_detection:v2
```

sudo docker push raphaelsun/mask_detection:v2

```
REPOSITORY                                TAG
    IMAGE ID       CREATED         SIZE
mask_detection                            v1
    599849381elf   5 hours ago     5.93GB
mask_detection                            v2
    599849381elf   5 hours ago     5.93GB
raphaelsun/mask_detection                 v2
    599849381elf   5 hours ago     5.93GB
```

2.2.4 Furthermore, we change the Kubernete yaml file:

spec:

containers:

- image: raphaelsun/mask_detection:v2

```
  spec:
    containers:
    - image: raphaelsun/mask_detection:v2
      command: ["/bin/bash"]
      args: ["-c", "/var/local/opendatacam/launch.sh"]
      name: opendatacam
      ports:
      - containerPort: 8080
      - containerPort: 8070
      - containerPort: 8090
```

2.2.5 Finally, we deploy the Kubernates:

```
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
opendatacam-mongo   1/1     1            1           6h6m
opendatacam         1/1     1            1           6h6m
```

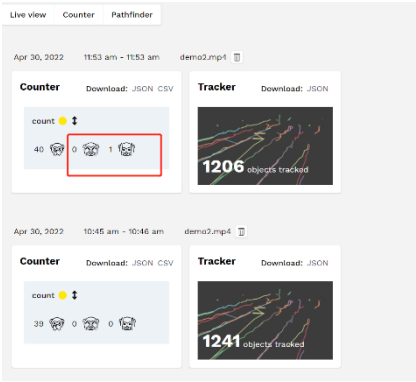
And getting the result:

In this result:

- we can successfully detect whether the people wear mask or not.
- The image not ghosting at all.

2.2.6 Further function

- Can make a warnings system for people do not wear mask
- The second icon represent the people do not wear the mask in a right way.
- The third icon represents no mask.
- When we got information from the second icon and the third icon, we can provide a warning.



3. Conclusion

Mask system can be widely use in China which can help us to reduce the manpower and resource to control the people for wearing the masks.