

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
```

CONTAINER ID	IMAGE	COMMAND	CREATED NAMES	STATUS	PORTS
a93b223038f0	maskcam/maskcam-beta	" /usr/bin/entry.sh d..."	13 hours ago priceless_booth	Up 13 hours	0.0.0.0:1883->1883/tcp, :::1883->1883/tcp, 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:8554->8554/tcp, :::8554->8554/tcp

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

```
Opening in BLOCKING MODE
ERROR: [TRT]: INVALID_CONFIG: The engine plan file is generated on an incompatible device, expecting compute 7.2 got compute 5
      3, please rebuild
ERROR: [TRT]: engine.cpp [1546]: Serialization Error in deserialize: 0 (Core engine serialization failure)
ERROR: [TRT]: INVALID_STATE: std::exception
ERROR: [TRT]: INVALID_CONFIG: Deserialize the cuda engine failed.
ERROR: Deserialize engine failed from file: /opt/maskcam_1.0/yolo/maskcam_yt_1024_608_fp16.trt
0:02:08 INFO: [NvDsInferContextImpl] Warning from NvDsInferContextImpl::deserializeEngineBackendDl() <nvdslinfer_context_>.impl.cpp:1690> [UID = 1]: Warning from file :/opt/maskcam_1.0/yolo/maskcam_yt_1024_608_fp16.trt failed
0:02:09 532574014 100 0x3c852b00 WARN nvinfer gstnvinfer.cpp:616:gst_nvinfer_logger::primary-inference>
NvDsInferContextImpl[UID 1]: Warning from NvDsInferContextImpl::generateBackendContext() <nvdslinfer_context_>.impl.cpp:1797> [UID = 1]: Warning from file :/opt/maskcam_1.0/yolo/maskcam_yt_1024_608_fp16.trt failed, try rebuild
0:02:09 532574014 100 0x3c852b00 INFO nvinfer gstnvinfer.cpp:619:gst_nvinfer_logger::primary-inference>
NvDsInferContextImpl[UID 1]: Info from NvDsInferContextImpl::buildModel() <nvdslinfer_context_>.impl.cpp:1751> [UID = 1]: Trying to create engine from model files
Yolo type is not defined from config file name:
ERROR: Failed to create network using custom network creation function
ERROR: Failed to get cuda engine from custom library API
0:02:10 628839336 100 0x3c852b00 ERROR nvinfer gstnvinfer.cpp:613:gst_nvinfer_logger::primary-inference>
NvDsInferContextImpl[UID 1]: Error in NvDsInferContextImpl::buildModel() <nvdslinfer_context_>.impl.cpp:1735> [UID = 1]: build engine file failed
0:02:15 628839336 100 0x3c852b00 ERROR nvinfer gstnvinfer.cpp:613:gst_nvinfer_logger::primary-inference>
NvDsInferContextImpl[UID 1]: Error in NvDsInferContextImpl::generateBackendContext() <nvdslinfer_context_>.impl.cpp:1821> [UID = 1]: build backend context failed
0:02:16 628839336 100 0x3c852b00 ERROR nvinfer gstnvinfer.cpp:613:gst_nvinfer_logger::primary-inference>
NvDsInferContextImpl[UID 1]: Error in NvDsInferContextImpl::initialzel() <nvdslinfer_context_>.impl.cpp:1148> [UID = 1]: generate backend failed, check config file settings
0:02:16 375037370 100 0x3c852b00 WARN nvinfer gstnvinfer.cpp:809:gst_nvinfer_start::primary-inference>
ERROR: Failed to create NvDsInferContext instance
0:02:16 375037370 100 0x3c852b00 WARN nvinfer gstnvinfer.cpp:809:gst_nvinfer_start::primary-inference>
ERROR: Config file path: maskcam_config.txt, NvDsInfer_Error: NVDSINFER_CONFIG_FAILED
    ERROR inference | gst-resource-error-quark: Failed to create NvDsInferContext instance ('1'):
        prints.py:42
        /dvs/gst/dirty/git-master/linux/deepstream/sdl/src/gst-plugins/gst-nvinfer/gstnvinfer.cpp(809):
        Config file path: maskcam_config.txt, NvDsInfer_Error: NVDSINFER_CONFIG_FAILED
    Config file path: maskcam_config.txt, NvDsInfer_Error: NVDSINFER_CONFIG_FAILED
    prints.py:48
INFO   inference | TROUBLESHOOTING HELP
```

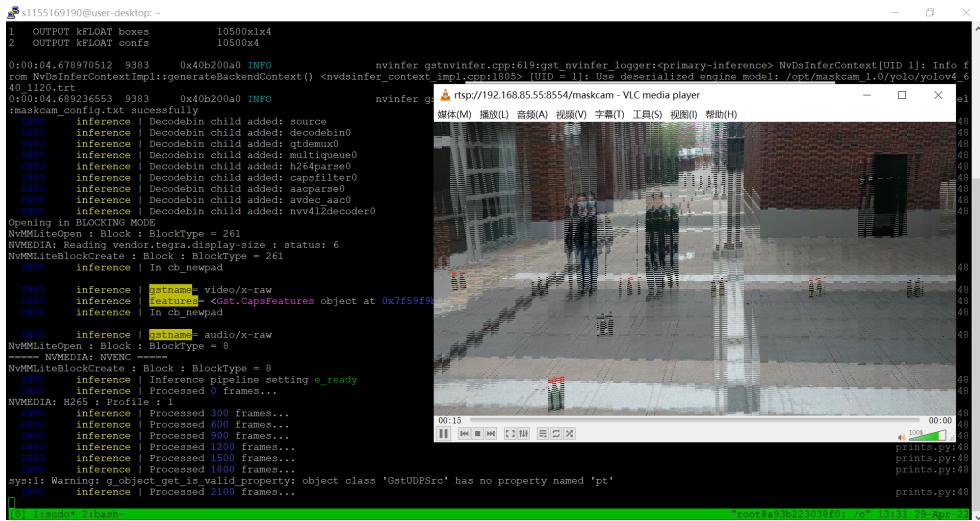
It shows that the GPU computing power is not match as the original model. Because this model is run through the trt file, our GPU computing power is not the same as the author. Therefore, we try to rebuild the darknet to create the onnx file. And using the onnx file to create the trt file.

```
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

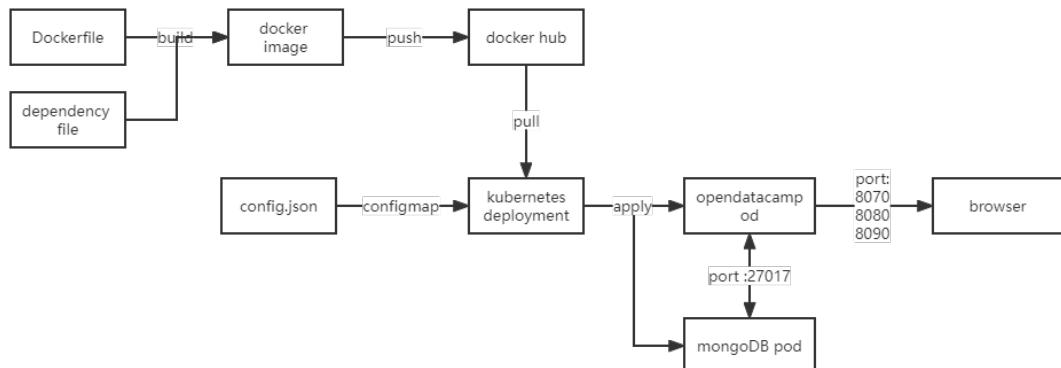
The disadvantages of this model are

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

2.2.4 Furthermore, we change the Kuberneete yaml file:

```
spec:
```

```
  containers:
```

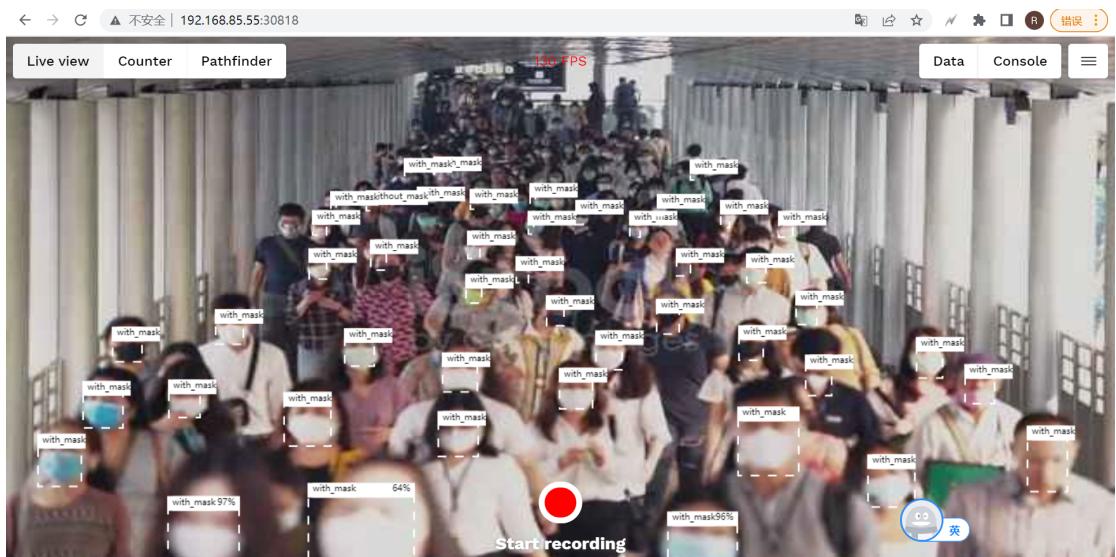
```
    - image: raphaelsun/mask_detection:v2
```

```
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 Kuberneates:

```
pi@raspberrypi: ~ $ kubectl get deploy/opendatacam  
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.