IEMS 5709
Emerging Topics in Information Engineering

Topic: IoT & Edge Computing

Group 11

Project Report: Deepstram application for Fall Detection

Instructor: Prof. ZAO Kar-kin, John

Eddie Christopher FOX III 1155160788
Ling JIN 1155162648
Lei XIA 1155169110

May 2022

For our project, we decided to look into Fall Detection because of its large potential applications in helping with the automated management / healthcare of individuals who may have diseases that make them prone to falling such as Parkinson's. It is estimated that between 45-68% of people with Parkinson's disease will fall each year, and between 50-86% of those that do will fall reccurrently [1], which is a danger to their health, especially as they get older and both the risk and damage of falling increases [1]. Assuming there is already a video camera monitoring each hospital room, the techniques used in this project could be deployed to detect two action states, normal and fall, where normal is anything but falling. Afterwards, an alert can be sent to inform staff to help the fallen patient.

Because this is a non-trivial project that does not involve merely modifying some opendatacam YAML files, and given our 1 week timeframe to complete the project, we are mainly concerned with fall detection from static video files rather than a livestream from an IP camera. Additionally, sending an alert is beyond the scope of this project.

As we do not have time to train a model, we wanted to use one that was already proven to work on Nvidia Jetson Edge devices and had good documentation. We went with the Action Recognition Net model [2], which is in Nvidia's NGC catalog. The model can be trained further, but we went with a pretrained version that is deployable on via Deepstream, which we have some experience with from Lab 2.

The model was trained on the HMDB51 dataset and there is 5 action classes, including fall_floor, for detecting when someone falls to the floor, which is what we are interested in. The videos the model was trained on were varied in visible body parts, quality, number of people involved, camera motion, and viewpoint, so it should be able to handle our imagined use case in a medical room.Reported accuracy is 90%, which is sufficient for this project. Because we are using this model, it will detect falling, walking, running, pushing, falling and riding a bike rather than just fall and normal, we don't have time to filter all actions but falling and to introduce a neutral state.

We decided to use the same dataset, HMDB [3] because the accuracy should be greater if we use the same videos that the model was trained on. We download the dataset onto the edge node using wget on [4]. About 2GB of videos with 7,000 clips distributed in 51 action classes.

名称	原位置	删除日期	大小	项目类型
walk.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	173,363 KB	RAR Other File
wave.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	28,750 KB	RAR Other File
talk.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	51,117 KB	RAR Other File
throw.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	32,695 KB	RAR Other File
turn.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	52,875 KB	RAR Other File
swing_baseball.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	56,942 KB	RAR Other File
sword.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	34,723 KB	RAR Other File
sword_exercise.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	37,699 KB	RAR Other File
smoke.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	42,027 KB	RAR Other File
somersault.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	34,784 KB	RAR Other File
stand.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	37,953 KB	RAR Other File
shoot_bow.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	65,710 KB	RAR Other File
shoot_gun.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	31,614 KB	RAR Other File
sit.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	40,062 KB	RAR Other File
situp.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	27,104 KB	RAR Other File
smile.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	17,543 KB	RAR Other File
run.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	60,805 KB	RAR Other File
shake_hands.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	49,794 KB	RAR Other File
shoot_ball.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	37,313 KB	RAR Other File
push.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	34,588 KB	RAR Other File
pushup.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	31,000 KB	RAR Other File
ride_bike.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	39,589 KB	RAR Other File
ride_horse.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	58,914 KB	RAR Other File
laugh.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	68,103 KB	RAR Other File
pick.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	33,643 KB	RAR Other File
pour.rar	C:\Users\dell\Downloads\hmdb51_org	28/4/2022 3:10 pm	46,268 KB	RAR Other File

The dataset comes in RAR files for each action. We delete the RAR for every action except for fall_floor and extract this folder into a dataset folder.

```
s1155169110@user-desktop:~$ cd ..
s1155169110@user-desktop:/home$ ll
total 40
drwxr-xr-x  8 root        root         4096 Apr 28 15:35 ./
drwxr-xr-x 22 root        root         4096 Aug  4  2021 ../
drwxr-xr-x  2 root        root        12288 Apr 28 15:36 dataset/
drwxr-xr-x  4 s11551      s11551       4096 Feb 20 17:10 s11551/
drwxr-xr-x 14 s1155160788 s1155160788  4096 Apr 23 17:00 s1155160788/
drwxr-xr-x 14 s1155162648 s1155162648  4096 Apr  6 16:37 s1155162648/
drwxr-xr-x 13 s1155169110 s1155169110  4096 Apr 28 14:09 s1155169110/
drwxr-xr-x 20 user        user         4096 Apr 27 14:33 user/
s1155169110@user-desktop:/home$ cd dataset/
s1155169110@user-desktop:/home/dataset$ ll
total 29832
drwxr-xr-x 2 root root  12288 Apr 28 15:36 ./
drwxr-xr-x 8 root root   4096 Apr 28 15:35 ../
-rw-r--r-- 1 root root 187904 Dec  7  2010 20060723sfjfffightcoldplay_fall_floor_f_cm_np1_fr_med_0.avi
-rw-r--r-- 1 root root 250368 Dec  7  2010 50_FIRST_DATES_fall_floor_f_cm_np1_fr_med_14.avi
-rw-r--r-- 1 root root  78336 Dec  7  2010 American_History_X_fall_floor_f_cm_np1_ba_med_3.avi
-rw-r--r-- 1 root root 289792 Dec  7  2010 American_History_X_fall_floor_f_cm_np1_le_bad_1.avi
-rw-r--r-- 1 root root 164352 Dec  7  2010 American_History_X_fall_floor_f_nm_np1_ba_bad_10.avi
-rw-r--r-- 1 root root 313856 Dec  7  2010 APOCALYPTO_fall_floor_f_nm_np1_ba_med_5.avi
-rw-r--r-- 1 root root 238080 Dec  7  2010 BATMAN_BEGINS_fall_floor_f_cm_np1_ba_med_11.avi
-rw-r--r-- 1 root root 193024 Dec  7  2010 BATMAN_BEGINS_fall_floor_f_cm_np1_ba_med_3.avi
-rw-r--r-- 1 root root 181248 Dec  7  2010 BATMAN_BEGINS_fall_floor_f_cm_np1_ba_med_5.avi
-rw-r--r-- 1 root root 221696 Dec  7  2010 BATMAN_BEGINS_fall_floor_h_cm_np1_fr_bad_2.avi
-rw-r--r-- 1 root root 320512 Dec  7  2010 BIG_FISH_fall_floor_f_cm_np1_fr_med_24.avi
-rw-r--r-- 1 root root 168960 Dec  7  2010 BIG_FISH_fall_floor_f_nm_np1_fr_bad_21.avi
-rw-r--r-- 1 root root 302080 Dec  7  2010 BIG_FISH_fall_floor_f_nm_np1_fr_med_11.avi
-rw-r--r-- 1 root root 271360 Dec  7  2010 BIG_FISH_fall_floor_f_nm_np1_fr_med_26.avi
-rw-r--r-- 1 root root 170496 Dec  7  2010 BLACK_HAWK_DOWN_fall_floor_f_cm_np1_fr_bad_34.avi
-rw-r--r-- 1 root root 260096 Dec  7  2010 BLACK_HAWK_DOWN_fall_floor_f_cm_np1_ri_med_13.avi
-rw-r--r-- 1 root root 306176 Dec  7  2010 BLACK_HAWK_DOWN_fall_floor_u_cm_np1_fr_med_0.avi
-rw-r--r-- 1 root root 182272 Dec  8  2010 Blade_Vs_Deacon_Frost_Sword_Fight_Scene_fall_floor_f_cm_np2_fr_med_3.avi
-rw-r--r-- 1 root root 306688 Dec  7  2010 CasinoRoyale_fall_floor_f_cm_np1_fr_med_7.avi
-rw-r--r-- 1 root root 154624 Dec  7  2010 CastAway1_fall_floor_u_nm_np1_fr_med_11.avi
-rw-r--r-- 1 root root 317952 Dec  7  2010 Catch_Me_If_You_Can_fall_floor_f_cm_np1_fr_med_0.avi
-rw-r--r-- 1 root root 205312 Dec  7  2010 Crash_fall_floor_u_cm_np1_fr_med_1.avi
-rw-r--r-- 1 root root 183296 Dec  7  2010 EVOLUTION_fall_floor_f_cm_np1_fr_med_21.avi
-rw-r--r-- 1 root root 248320 Dec  7  2010 EVOLUTION_fall_floor_f_nm_np1_ba_med_5.avi
-rw-r--r-- 1 root root 232960 Dec  7  2010 Fellowship_3_fall_floor_f_nm_np1_fr_med_3.avi
-rw-r--r-- 1 root root 373248 Dec  7  2010 Fellowship_4_fall_floor_u_cm_np1_fr_med_5.avi
-rw-r--r-- 1 root root 179200 Dec  7  2010 Fellowship_5_fall_floor_u_cm_np1_fr_med_8.avi
-rw-r--r-- 1 root root 110080 Dec  7  2010 Fellowship_6_fall_floor_f_nm_np1_fr_med_9.avi
-rw-r--r-- 1 root root 260608 Dec  7  2010 Fellowship_7_fall_floor_f_cm_np1_ba_med_2.avi
-rw-r--r-- 1 root root 260096 Dec  7  2010 Fellowship_7_fall_floor_f_cm_np1_ba_med_3.avi
-rw-r--r-- 1 root root 306176 Dec  7  2010 Finding_Forrester_3_fall_floor_f_cm_np2_ri_med_1.avi
-rw-r--r-- 1 root root 201728 Dec  7  2010 Glory_fall_floor_f_cm_np1_ba_med_46.avi
-rw-r--r-- 1 root root 218624 Dec  7  2010 Glory_fall_floor_f_cm_np1_fr_med_39.avi
-rw-r--r-- 1 root root 609792 Dec  7  2010 Glory_fall_floor_u_cm_np1_fr_med_55.avi
-rw-r--r-- 1 root root 180224 Dec  7  2010 HP_PRISONER_OF_AZKABAN_fall_floor_f_cm_np1_ri_med_5.avi
-rw-r--r-- 1 root root 248832 Dec  7  2010 HP_PRISONER_OF_AZKABAN_fall_floor_u_cm_np1_fr_med_14.avi
-rw-r--r-- 1 root root 253440 Dec  7  2010 KUNG_FU_HUSTLE_fall_floor_f_cm_np1_ba_bad_20.avi
-rw-r--r-- 1 root root 198656 Dec  7  2010 KUNG_FU_HUSTLE_fall_floor_f_cm_np1_ba_bad_21.avi
-rw-r--r-- 1 root root 136192 Dec  7  2010 KUNG_FU_HUSTLE_fall_floor_f_cm_np1_ba_bad_33.avi
-rw-r--r-- 1 root root 147456 Dec  7  2010 KUNG_FU_HUSTLE_fall_floor_f_cm_np1_ba_bad_49.avi
-rw-r--r-- 1 root root 118272 Dec  7  2010 KUNG_FU_HUSTLE_fall_floor_f_cm_np1_ba_med_35.avi
```

```
-rw-r--r-- 1 root root 212480 Dec  7  2010 THE_PROTECTOR_fall_floor_f_nm_np1_ba_bad_94.avi
-rw-r--r-- 1 root root 425984 Dec  7  2010 THE_PROTECTOR_fall_floor_f_nm_np1_fr_bad_100.avi
-rw-r--r-- 1 root root 174592 Dec  7  2010 THE_PROTECTOR_fall_floor_f_nm_np1_fr_bad_93.avi
-rw-r--r-- 1 root root 223744 Dec  7  2010 THE_PROTECTOR_fall_floor_f_nm_np1_fr_goo_10.avi
-rw-r--r-- 1 root root 300032 Dec  7  2010 THE_PROTECTOR_fall_floor_f_nm_np1_fr_med_18.avi
-rw-r--r-- 1 root root 364032 Dec  7  2010 THE_PROTECTOR_fall_floor_f_nm_np1_fr_med_61.avi
-rw-r--r-- 1 root root 179200 Dec  7  2010 THE_PROTECTOR_fall_floor_f_nm_np1_fr_med_84.avi
-rw-r--r-- 1 root root 115200 Dec  7  2010 THE_PROTECTOR_fall_floor_f_nm_np1_le_bad_2.avi
-rw-r--r-- 1 root root 229888 Dec  7  2010 THE_PROTECTOR_fall_floor_f_nm_np1_le_bad_67.avi
-rw-r--r-- 1 root root 137728 Dec  7  2010 THE_PROTECTOR_fall_floor_f_nm_np1_ri_bad_92.avi
-rw-r--r-- 1 root root 210944 Dec  7  2010 THE_PROTECTOR_fall_floor_u_cm_np1_ba_med_86.avi
-rw-r--r-- 1 root root 206848 Dec  7  2010 THE_PROTECTOR_fall_floor_u_cm_np1_fr_med_56.avi
-rw-r--r-- 1 root root 308736 Dec  7  2010 Two_Towers_3_fall_floor_u_cm_np1_fr_med_6.avi
-rw-r--r-- 1 root root 331264 Dec  7  2010 Veoh_Alpha_Dog_1_fall_floor_f_cm_np1_fr_med_35.avi
-rw-r--r-- 1 root root 203264 Dec  7  2010 Veoh_Alpha_Dog_1_fall_floor_u_cm_np1_fr_med_46.avi
s1155169110@user-desktop:/home/dataset$ ls | wc -l
136
s1155169110@user-desktop:/home/dataset$
```

There are 136 videos total, all in AVI format, and varying in video quality, signified by med for medium, goo for good, and bad for bad.

We created a “Fall.mp4” file by combining all the .avi files within the falling dataset. The end result is a 4.5 minute video. Many of the individual video clips were only several seconds long, and we discovered that unless the video is sufficiently long, the RTSP video stream doesn’t have time to start before the edge node has already finished inferencing all frames within the video and thus terminates the RTSP video pipeline.

During the programming / configuration of the container, we refer to [5] for documentation on Action Recognition Net, [6] for a guide on Deepstream 3d action recognition apps, [7] for a walkthrough blog of using Action Recognition Net with Deepstream, and [8] for documentation on the deepstream-l4t 6.0.1 containers.

We looked extensively, but the included 3d action recognition Deepstream app only supports outputting to a GUI application rather than allowing for output to a video file or RTSP video stream. The normal way to support GUI output on the local computer would be to install Xming server on the local server, then enable X11 forwarding in Putty. We tried this, but unfortunately, documentation specifically notes “On Jetson, running a DeepStream application over SSH (via putty) with X11 forwarding does not work” [9].

Because of this, we either had to change to another project, likely an opendatacam one given the limited time we had left by this point or find a way to pipe the output of the Deepstream 3d action recognition apps to a RTSP video stream instead. There was no easy way to do this, but someone on the internet wrote 941 lines of code to allow for this functionality [10].

We first pull the Deepstream L4T 6.0 Triton image using: “sudo docker pull nvcr.io/nvidia/deepstream-l4t:6.0-triton”. Triton has all of the base content of Deepstream L4T, along with all samples and developer libraries.

Screenshot on next page.

```
sl155160788@user-desktop:/home/FinalProject$ sudo docker pull nvcr.io/nvidia/deepstream-l4t:6.0-triton
6.0-triton: Pulling from nvidia/deepstream-l4t
b9bb7af7248f: Already exists
334a570c08f5: Already exists
c8bd4a89e71c: Already exists
0a8ce4f08307: Already exists
2e163bfa50fc: Already exists
8d6d8c86a148: Already exists
b483bad8509f: Already exists
8da3c6b29858: Already exists
a3e94453e320: Already exists
40ad1f92affd: Already exists
8ac6e2f9e2a0: Already exists
cc932e62bbab: Already exists
d77841654f58: Already exists
2bb42f0b6424: Already exists
954e18eab77a: Already exists
facc6f8bcc9b: Already exists
6f68af23302a: Already exists
5a459fbc656a: Pull complete
292622a2b35f: Pull complete
4bf8252ad73b: Pull complete
3a0b3a8a3356: Pull complete
388e90b3803a: Pull complete
Digest: sha256:98051f15f17df6d22b141e83330629e050632cbb3438cfd080107eb82e92b39e
Status: Downloaded newer image for nvcr.io/nvidia/deepstream-l4t:6.0-triton
nvcr.io/nvidia/deepstream-l4t:6.0-triton
sl155160788@user-desktop:/home/FinalProject$ ls
dataset  deepstreamtest3zyh.tar.gz
sl155160788@user-desktop:/home/FinalProject$
```

As suggested by Gary, we enter the image, figure out what commands need to be run for our application to succeed, and create a dockerfile based on this. Before doing so, we create a FinalProject folder on the edge node that contains our combined Fall.mp4 as well as [11], a packaging of the code from [10] that has all the necessary code, makefiles, the action recognition net model, and the text configuration files. All of this is put into a folder called "deepstream-3d-action-recognition-zyh", which we include in our FinalProject folder.

By combining instructions from [10] and Lab 2, we come up with the following run command:

"sudo docker run -it --gpus all --rm --name=test --privileged=true -p 8554:8554 -v /home/FinalProject/:/home/ -w /home/deepstream-3d-action-recognition-zyh nvcr.io/nvidia/deepstream-l4t:6.0-triton /bin/bash"

This command runs the Deepstream 6.0.1 Triton image with all GPU's and privileged access and exposes the 8554 port in the container for access from our local computer via GlobalProtect. The -v command binds the FinalProject folder under home outside the container to be our home folder inside the container. The -w specifies the work directory to be the "deepstream-3d-action-recognition-zyh" folder where the packaged code is.

We were originally encountering some warnings / errors about nvidia specific libraries but were able to resolve this by running "apt-get update && apt-get -y upgrade" within the container to upgrade any possible deepstream dependency to the latest version. We did not need to install anything new because Triton already has all the developer libraries and deepstream functionalities.

After upgrading, we run "make clean" and "make -j4" to create the "deepstream-3d-action-recognition" application. Again, this is the same as the 3d action recognition Deepstream program provided by Nvidia with the deepstream container, but with code to allow for RTSP video output.

After running these make commands, the created application is within the bound FinalProject folder, so the make commands don't need to be in the Dockerfile, it is a one-time action.

```
root@a946ec292f6c:/home/deepstream-3d-action-recognition-zyh# ls
Fall.mp4  README                    config_infer_primary_3d_action.txt  config_preprocess_3d_custom.txt  deepstream_3d_action_recognition.cpp  deepstream_action_config_parse.cpp     labels.txt                                  resnet18_3d_rgb_hmdb5_32.etlt_b4_gpu0_fp16.engine
Makefile  config_infer_primary_2d_action.txt  config_preprocess_2d_custom.txt  custom_sequence_preprocess        deepstream_action.h                   deepstream_action_recognition_config.txt  resnet18_3d_rgb_hmdb5_32.etlt
root@a946ec292f6c:/home/deepstream-3d-action-recognition-zyh# make -j4
g++ -c -o deepstream_action_config_parse.o -fvisibility=hidden -DPLATFORM_TEGRA -I /opt/nvidia/deepstream/deepstream/sources/includes -I /opt/nvidia/deepstream/deepstream/sources/apps/apps-common/includes -I /opt/nvidia/deepstream/deepstream/sources/gst-plugins/gst-nvdspreprocess/include -I /usr/local/cuda/include -
FPIC -std=c++14 -pthread -I/usr/include/gstreamer-1.0 -I/usr/include/glib-2.0 -I/usr/lib/aarch64-linux-gnu/glib-2.0/include deepstream_action_config_parse.cpp
g++ -c -o deepstream_3d_action_recognition.o -fvisibility=hidden -DPLATFORM_TEGRA -I /opt/nvidia/deepstream/deepstream/sources/includes -I /opt/nvidia/deepstream/deepstream/sources/apps/apps-common/includes -I /opt/nvidia/deepstream/deepstream/sources/gst-plugins/gst-nvdspreprocess/include -I /usr/local/cuda/include
-FPIC -std=c++14 -pthread -I/usr/include/gstreamer-1.0 -I/usr/include/glib-2.0 -I/usr/lib/aarch64-linux-gnu/glib-2.0/include deepstream_3d_action_recognition.cpp
make -C custom_sequence_preprocess
make[1]: Entering directory '/home/deepstream-3d-action-recognition-zyh/custom_sequence_preprocess'
g++ -c -o sequence_image_process.o -fvisibility=hidden -Wall -Werror -DPLATFORM_TEGRA -I /opt/nvidia/deepstream/deepstream/sources/includes -I /opt/nvidia/deepstream/deepstream/sources/gst-plugins/gst-nvdspreprocess/include -I /usr/local/cuda/include -FPIC -std=c++14 -pthread -I/usr/incl
ude/glib-2.0 -I/usr/lib/aarch64-linux-gnu/glib-2.0/include sequence_image_process.cpp
-fvisibility=hidden -Wall -Werror -DPLATFORM_TEGRA -I /opt/nvidia/deepstream/deepstream/sources/includes -I /opt/nvidia/deepstream/deepstream/sources/gst-plugins/gst-nvdspreprocess/include -I /usr/local/cuda/include -FPIC -std=c++14 -pthread -I/usr/include/gstreamer-1.0 -I/usr/include/glib-2.0 -I/usr/lib/aarch64-lin
ux-gnu/glib-2.0/include
/usr/local/cuda/bin/nvcc -c -o sequence_preprocess_kernel.o --compiler-options '-fPIC' sequence_preprocess_kernel.cu
g++ -o deepstream-3d-action-recognition deepstream_action_config_parse.o deepstream_3d_action_recognition.o -lgstreamer-1.0 -lgobject-2.0 -lglib-2.0 -L/usr/local/cuda/lib64/ -lcudart -lnvdsgst_helper -lm -lgstrtspserver-1.0 -L/opt/nvidia/deepstream/deepstream/lib/ -lnvdsgst_meta -lnvds_meta -lcuda -Wl,-rpath,/opt/nv
idia/deepstream/deepstream/lib/
g++ -o libnvds_custom_sequence_preprocess.so ./sequence_image_process.o ./sequence_preprocess_kernel.o -shared -Wl,--start-group -lgstreamer-1.0 -lgobject-2.0 -lglib-2.0 -L/usr/local/cuda/lib64/ -lcudart -lnvdsgst_helper -lm -L/opt/nvidia/deepstream/deepstream/lib/ -lnvdsgst_meta -lnvds_meta -lcuda -Wl,-rpath,/opt/n
vidia/deepstream/deepstream/lib/ -Wl,--end-group
make[1]: Leaving directory '/home/deepstream-3d-action-recognition-zyh/custom_sequence_preprocess'
root@a946ec292f6c:/home/deepstream-3d-action-recognition-zyh# ls
Fall.mp4  README                    config_infer_primary_3d_action.txt  config_preprocess_3d_custom.txt  deepstream-3d-action-recognition      deepstream_3d_action_recognition.o  deepstream_action_config_parse.cpp  deepstream_action_recognition_config.txt  resnet18_3d_rgb_hmdb5_32.etlt
Makefile  config_infer_primary_2d_action.txt  config_preprocess_2d_custom.txt  custom_sequence_preprocess        deepstream_3d_action_recognition.cpp  deepstream_action.h                 deepstream_action_config_parse.o    labels.txt                                resnet18_3d_rgb_hmdb5_32.etlt_b4_gpu0_fp16.engine
root@a946ec292f6c:/home/deepstream-3d-action-recognition-zyh#
```

Following [7] and [10], we make some modifications to the text files to fit our use case.

Screenshot on next page.

```
# deepstream action recognition config settings.
# run:
# $ deepstream-3d-action-recognition -c deepstream_action_recognition_config.txt

[action-recognition]

# stream/file source list
uri-list=file:///home/deepstream-3d-action-recognition-zyh/Fall.mp4

# eglglessink settings
display-sync=1


# <preprocess-config> is the config file path for nvdspreprocess plugin
# <infer-config> is the config file path for nvinfer plugin

# Enable 3D preprocess and inference
preprocess-config=config_preprocess_3d_custom.txt
infer-config=config_infer_primary_3d_action.txt

# Uncomment to enable 2D preprocess and inference
#preprocess-config=config_preprocess_2d_custom.txt
#infer-config=config_infer_primary_2d_action.txt

# nvstreammux settings
muxer-height=720
muxer-width=1280

# nvstreammux batched push timeout in usec
muxer-batch-timeout=40000


# nvmultistreamtiler settings
tiler-height=720
tiler-width=1280

# Log debug level. 0: disabled. 1: debug. 2: verbose.
debug=1

# Enable fps print on screen. 0: disable. 1: enable
enable-fps=1
```

deepstream_action_recognition_config.txt is the main text file that is run in the final command that launches the deepstream-3d-action-recognition app. We change the uri-list to direct to the Fall.mp4 file we have inside the working directory. Another change we make is setting debug=1 so that we can tell when inferencing actually begins.

There is the option for either inference based on a 3d action recognition model or a 2d one, determined by which text file lines are commented within the main text file.

model	dataset	center accuracy	conv accuracy
resnet18_2d_rgb_hmdb5_32	HMDB5	84.69%	82.88%
resnet18_3d_rgb_hmdb5_32	HMDB5	84.69%	85.59%

Model Type	Device	precision	batch_size	FPS
3D-RGB	Jetson Nano	FP16	1	0.57
3D-RGB	Jetson NX	FP16	4	4.9
3D-RGB	Jetson Xavier	FP16	4	33
3D-RGB	T4	FP16	4	137
2D-RGB	Jetson Nano	FP16	1	30
2D-RGB	Jetson NX	FP16	16	250
2D-RGB	Jetson Xavier	FP16	16	490
2D-RGB	T4	FP16	16	1818

Based on the Action Recognition Net documentation [2], the 2d model seems more compelling. The accuracy drops from 86 to 83%, but the frames per second goes up from 5 to 250 on the Jetson Xavier NX. Unfortunately, the custom code that allows for RTSP video output using these actions only support the 3d model.

Screenshot on next page.

root@a846ec292f6c: /home/deepstream-3d-action-recognition-zyh

```
################################################################################
# Copyright (c) 2021, NVIDIA CORPORATION. All rights reserved.
#
# Permission is hereby granted, free of charge, to any person obtaining a
# copy of this software and associated documentation files (the "Software"),
# to deal in the Software without restriction, including without limitation
# the rights to use, copy, modify, merge, publish, distribute, sublicense,
# and/or sell copies of the Software, and to permit persons to whom the
# Software is furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  IN NO EVENT SHALL
# THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
# FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
# DEALINGS IN THE SOFTWARE.
################################################################################

# The values in the config file are overridden by values set through GObject
# properties.

[property]
enable=1
target-unique-ids=1

    # network-input-shape: batch, channel, sequence, height, width
# 3D sequence of 64 images
#network-input-shape= 1;3;64;224;224

# 3D sequence of 32 images
network-input-shape= 1;3;32;224;224

    # 0=RGB, 1=BGR, 2=GRAY
network-color-format=0
    # 0=NCHW, 1=NHWC, 2=CUSTOM
network-input-order=2
    # 0=FP32, 1=UINT8, 2=INT8, 3=UINT32, 4=INT32, 5=FP16
tensor-data-type=0
tensor-name=input_rgb

processing-width=224
processing-height=224

    # 0=NVBUF_MEM_DEFAULT 1=NVBUF_MEM_CUDA_PINNED 2=NVBUF_MEM_CUDA_DEVICE
    # 3=NVBUF_MEM_CUDA_UNIFIED  4=NVBUF_MEM_SURFACE_ARRAY(Jetson)
scaling-pool-memory-type=0

    # 0=NvBufSurfTransformCompute_Default 1=NvBufSurfTransformCompute_GPU
    # 2=NvBufSurfTransformCompute_VIC(Jetson)
scaling-pool-compute-hw=0

    # Scaling Interpolation method
    # 0=NvBufSurfTransformInter_Nearest 1=NvBufSurfTransformInter_Bilinear 2=NvBufSurfTransformInter_Algo1
    # 3=NvBufSurfTransformInter_Algo2 4=NvBufSurfTransformInter_Algo3 5=NvBufSurfTransformInter_Algo4
    # 6=NvBufSurfTransformInter_Default
scaling-filter=0

    # model input tensor pool size
tensor-buf-pool-size=8

custom-lib-path=/opt/nvidia/deepstream/deepstream-6.0/lib/libnvds_custom_sequence_preprocess.so
#custom-lib-path=./custom_sequence_preprocess/libnvds_custom_sequence_preprocess.so
custom-tensor-preparation-function=CustomSequenceTensorPreparation

# 3D conv custom params
[user-configs]
channel-scale-factors=0.007843137;0.007843137;0.007843137
channel-mean-offsets=127.5;127.5;127.5
stride=1
subsample=0

[group-0]
src-ids=0
process-on-roi=1
roi-params-src-0=0;0;1280;720
#roi-params-src-1=0;0;1280;720
#roi-params-src-2=0;0;1280;720
#roi-params-src-3=0;0;1280;720

~
~
"config_preprocess_3d_custom.txt" 83L, 3102C
```

Above is the “config_preprocess_3d_custom.txt” file that handles pre-processing. This is necessary because the action recognition net model can’t actually process video, rather it processes rgb frames, optical flows, or both combined. We make some changes to the network as suggested in [7] to input shape, src-ids, and roi-params-src lines to account for only having 1 video input, down from the default of 4.

Screenshot on next page of the config_infer_primary_3d_action.txt file.

```
# Following properties are mandatory when engine files are not specified:
#   int8-calib-file(Only in INT8)
#   Caffemodel mandatory properties: model-file, proto-file, output-blob-names
#   UFF: uff-file, input-dims, uff-input-blob-name, output-blob-names
#   ONNX: onnx-file
#
# Mandatory properties for detectors:
#   num-detected-classes
#
# Optional properties for detectors:
#   cluster-mode(Default=Group Rectangles), interval(Primary mode only, Default=0)
#   custom-lib-path,
#   parse-bbox-func-name
#
# Mandatory properties for classifiers:
#   classifier-threshold, is-classifier
#
# Optional properties for classifiers:
#   classifier-async-mode(Secondary mode only, Default=false)
#
# Optional properties in secondary mode:
#   operate-on-gie-id(Default=0), operate-on-class-ids(Defaults to all classes),
#   input-object-min-width, input-object-min-height, input-object-max-width,
#   input-object-max-height
#
# Following properties are always recommended:
#   batch-size(Default=1)
#
# Other optional properties:
#   net-scale-factor(Default=1), network-mode(Default=0 i.e FP32),
#   model-color-format(Default=0 i.e. RGB) model-engine-file, labelfile-path,
#   mean-file, gie-unique-id(Default=0), offsets, process-mode (Default=1 i.e. primary),
#   custom-lib-path, network-mode(Default=0 i.e FP32)
#
# The values in the config file are overridden by values set through GObject
# properties.

[property]
gpu-id=0

tlt-encoded-model=resnet18_3d_rgb_hmdb5_32.etlt
tlt-model-key=nvidia_tao
model-engine-file=resnet18_3d_rgb_hmdb5_32.etlt_b4_gpu0_fp16.engine

labelfile-path=labels.txt
batch-size=1
process-mode=1

# requries preprocess metadata input
input-tensor-from-meta=1

## 0=FP32, 1=INT8, 2=FP16 mode
network-mode=2
gie-unique-id=1

# 1: classifier, 100: custom
network-type=1

# Let application to parse the inference tensor output
output-tensor-meta=1
tensor-meta-pool-size=8
~
```

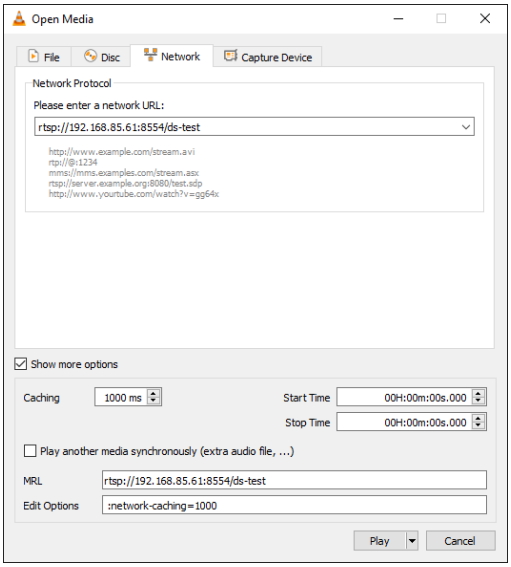
In config_infer_primary_3d_action.txt, we also change the batch-size from 4 to 1 to account for the number of video inputs being only one. This text file specifies the deployable version of the Action Recognition Net model as the target model to be used for inference.

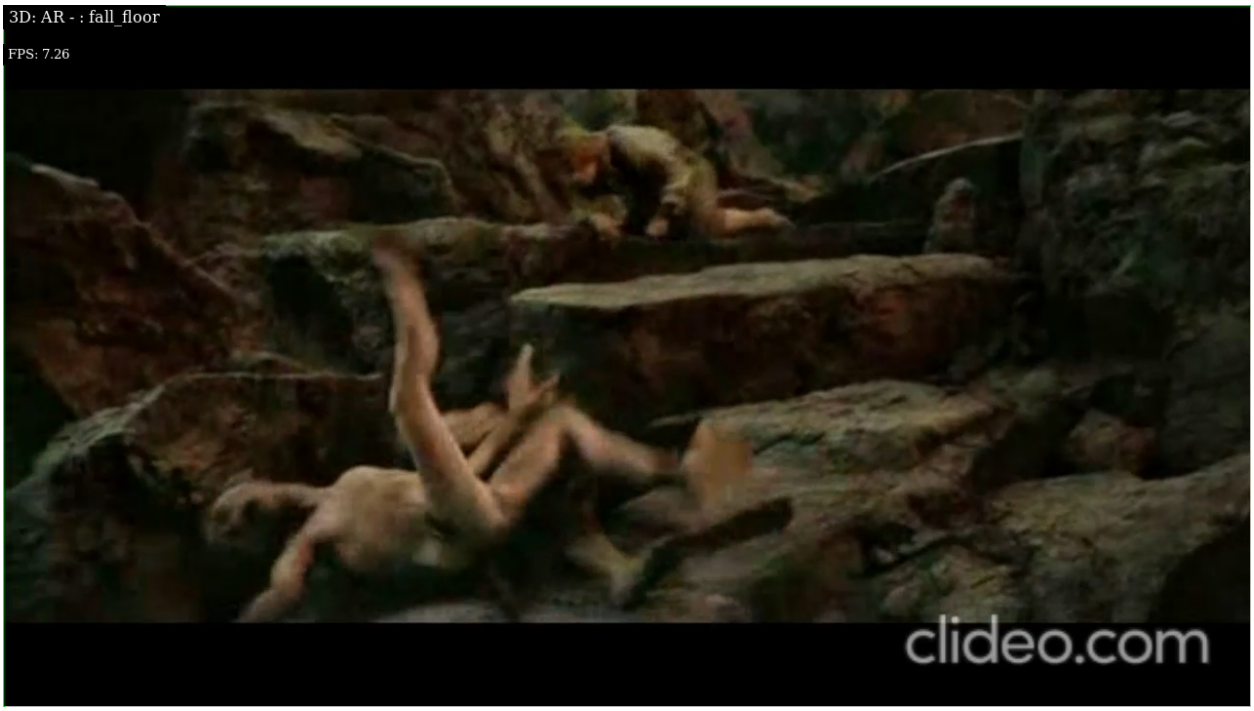
Almost everything above is one time setup that doesn't have to happen again, since we are making changes to text files in the bound folder.

We execute the application by running "./deepstream-3d-action-recognition -c deepstream_action_recognition_config.txt". There is only a change to deepstream_action_recognition_config.txt if we want to perform inferencing on a different source file or RTSP stream, and everything else remains untouched.

After execution begins and we see inferencing taking place in the debug log, as above, we use VLC Media Player to open a "rtsp://192.168.85.61:8554/ds-test" network stream.

Screenshot on next page.





Every text line in the debug log is a frame of the video that has been analyzed by the application and has its action classified. The classified action is overlaid as text on the top left of the video, along with the frame rate. Although 7 frames per second seems lackluster, it is fairly difficult to perform 3d model action recognition over time as opposed to just classifying and tracking an object / person. This is actually somehow above the average documented performance on a Xavier NX of 5 frames per second, perhaps because we maxed the wattage, cores, and clocks. Among some

limitations, it is only capable of tracking one region of interest / person, so if there are multiple people, it may track someone who is standing and classify the actions as walking, even if someone is on the ground. This would need to be accounted for in a hospital setting to be able to track more than one person and to send an alert if even one person is detected on the floor. There may appear to be more inaccuracies in classification then there actually are according to the application debug logs because VLC has issues playing a 7 fps stream in real time. It will show playback in bursts and need to choose frames to throw away, which sometimes includes ones where the action is classified as fall_floor.

Dockerfile:

We start from the one provided in Lab 3.

Dockerfile ×

C: > Users > Eddie > Desktop > IEMS 5709 - Emerging Topics in Information Engineering > Week 15 > Dockerfile > ...

```
FROM nvcr.io/nvidia/deepstream-l4t:6.0.1-triton

RUN apt-get update && apt-get -y upgrade

WORKDIR /opt/nvidia/deepstream/deepstream-6.0

EXPOSE 8554

ENV NVIDIA_VISIBLE_DEVICES all
ENV NVIDIA_DRIVER_CAPABILITIES all
CMD ["/bin/bash"]
#EOF
```

Since we handle almost all the processing in the bound folder, we don't need to do much when building our image. We run apt-get update and apt-get upgrade, expose the port, and use the /bin/bash command. When testing with Kubernetes, we later needed to add the two ENV lines in order to give the pods access to the full capabilities of the GPU through the nvidia container runtime, referring to [12].

Docker Build:

We use "sudo docker build -t finalproject:v1 ." to build the docker file.

Screenshot on next page.

```
sl155160788@user-desktop:/home/FinalProject$ sudo docker build -t finalproject:v1 .
Sending build context to Docker daemon  285.7MB
Step 1/7 : FROM nvcr.io/nvidia/deepstream-l4t:6.0.1-triton
 ---> d3984db2b6b1
Step 2/7 : RUN apt-get update
 ---> Running in 25dcebe14bad
Get:1 http://ports.ubuntu.com/ubuntu-ports bionic InRelease [242 kB]
Get:2 http://ports.ubuntu.com/ubuntu-ports bionic-updates InRelease [88.7 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports bionic-backports InRelease [74.6 kB]
Get:4 http://ports.ubuntu.com/ubuntu-ports bionic-security InRelease [88.7 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports bionic/multiverse arm64 Packages [153 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports bionic/main arm64 Packages [1285 kB]
Get:7 http://ports.ubuntu.com/ubuntu-ports bionic/universe arm64 Packages [11.0 MB]
Get:8 http://ports.ubuntu.com/ubuntu-ports bionic/restricted arm64 Packages [572 B]
Get:9 http://ports.ubuntu.com/ubuntu-ports bionic-updates/multiverse arm64 Packages [5351 B]
Get:10 http://ports.ubuntu.com/ubuntu-ports bionic-updates/main arm64 Packages [1888 kB]
Get:11 http://ports.ubuntu.com/ubuntu-ports bionic-updates/universe arm64 Packages [2027 kB]
Get:12 http://ports.ubuntu.com/ubuntu-ports bionic-updates/restricted arm64 Packages [5042 B]
Get:13 http://ports.ubuntu.com/ubuntu-ports bionic-backports/universe arm64 Packages [12.5 kB]
Get:14 http://ports.ubuntu.com/ubuntu-ports bionic-backports/main arm64 Packages [12.2 kB]
Get:15 http://ports.ubuntu.com/ubuntu-ports bionic-security/main arm64 Packages [1506 kB]
Get:16 http://ports.ubuntu.com/ubuntu-ports bionic-security/universe arm64 Packages [1331 kB]
Get:17 http://ports.ubuntu.com/ubuntu-ports bionic-security/multiverse arm64 Packages [2759 B]
Get:18 http://ports.ubuntu.com/ubuntu-ports bionic-security/restricted arm64 Packages [4372 B]
Fetched 19.7 MB in 6s (3290 kB/s)
Reading package lists... Done
Removing intermediate container 25dcebe14bad
 ---> 7f9db3aedf66
Step 3/7 : WORKDIR /opt/nvidia/deepstream/deepstream-6.0
 ---> Running in 9545c5f983c4
Removing intermediate container 9545c5f983c4
 ---> 467bf4952273
Step 4/7 : EXPOSE 8554
 ---> Running in 5bd5b30f37a2
Removing intermediate container 5bd5b30f37a2
 ---> 9b1e4883ab8a
Step 5/7 : ENV NVIDIA_VISIBLE_DEVICES all
 ---> Running in 6d6080a79b5f
Removing intermediate container 6d6080a79b5f
 ---> a3d7392eccff
Step 6/7 : ENV NVIDIA_DRIVER_CAPABILITIES all
 ---> Running in 47d58a2cb5f3
Removing intermediate container 47d58a2cb5f3
 ---> ec238f3d0897
Step 7/7 : CMD ["/bin/bash"]
 ---> Running in f61fe56a049a
Removing intermediate container f61fe56a049a
 ---> 4fd2dbd4228e
Successfully built 4fd2dbd4228e
Successfully tagged finalproject:v1
sl155160788@user-desktop:/home/FinalProject$
```

We can see the finalproject final image here.

We confirm that the commands still work with this image we built with a Dockerfile. For the run command, we reference Lab 3's run command, but add the bound folder and work directory from Lab 2's run command as well as -it since we are going into the container to execute the command. The number and letter combination at the end corresponds to the image ID of our built image.

Run command: "sudo docker run -it --runtime nvidia --gpus all --publish 8554:8554 -v /home/FinalProject/:/home/ -w /home/deepstream-3d-action-recognition-zyh 4fd2dbd4228e"

Kubernetes Deployment:

AR-Deployment.yaml

AR-Deployment.yaml

C: > Users > Eddie > Desktop > IEMS 5709 - Emerging Topics in Information Engineering > Week 15 > AR-Deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: ar
  name: ar
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ar
      tier: frontend
  template:
    metadata:
      labels:
        app: ar
        tier: frontend
    spec:
      containers:
      - image: eddiecfox/iems5709-final-project:v1
        command: ["/bin/bash", "-c", "--"]
        args: ["while true; do sleep 30; done;"]
        name: ar
        ports:
        - containerPort: 8554
        resources: {}
        securityContext:
          privileged: true
        volumeMounts:
        - mountPath: /home
          name: finalproject
      restartPolicy: Always
      volumes:
      - name: finalproject
        hostPath:
          path: /home/FinalProject
          type: Directory
```

AR-Service.yaml

Screenshot on next page.

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: ar
  name: ar
spec:
  ports:
  - name: "8554"
    port: 8554
  selector:
    app: ar
    tier: frontend
  type: LoadBalancer
```

We start with Lab 4's opendatacam Kubernetes code and make modifications to fit our use case. We were also inspired by [13]. Specifically, we only need one port exposed this time, 8554. We create a deployment to handle managing the containerized image and a service to expose the port. We change the application name to "ar" to reflect the purpose. We refer to [14] to create a command that will keep the Kubernetes pod running forever when deployed instead of exiting and to [15] for the syntax of creating a volume to mount the /home/FinalProject from the local node to the /home folder of the container, like we do with the "docker run -v" parameter in the containerized run of the application.

Lens IDE screenshots:

Pods 2 items Namespace: default Search Pods...

Name	Names...	Contain...	Restarts	Controll...	Node	QoS	A...	Stat...
ar-577c6864c6-ch69x	default		0	ReplicaSet	user-desktop	BestEffort	6h9m	Running
svclb-ar-zjj62	default		0	DaemonSet	user-desktop	BestEffort	14h	Running

Deployments 1 item Namespace: default Search Deployments...

Name	Namespace	Pods	Replicas	Age	Conditions
ar	default	1/1	1	6h9m	Available Progressing

Services 2 items Namespace: default Search Services...

Name	Namespace	Type	Cluster IP	Ports	External IP	Selector	Age	Status
ar	default	LoadBalancer	10.43.24.154	8554:32277/TCP	192.168.85.61	app=ar tier=frontend	14h	Active
kubernetes	default	ClusterIP	10.43.0.1	443:6443/TCP	-		29d	Active

Further Docker / Kubectl commands with Kubernetes:

```
s1155160788@user-desktop:/home/FinalProject$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: eddiecfox
Password:
WARNING! Your password will be stored unencrypted in /home/s1155160788/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

```
s1155160788@user-desktop:/home/FinalProject$ sudo docker tag finalproject:v1 eddiecfox/iems5709-final-project:v1
s1155160788@user-desktop:/home/FinalProject$ sudo docker image ls
REPOSITORY                                TAG                                 IMAGE ID       CREATED          SIZE
eddiecfox/iems5709-final-project          v1                                  43d0b051b2c0   3 minutes ago    4.01GB
finalproject                              v1                                  43d0b051b2c0   3 minutes ago    4.01GB
```

Like in Lab 3, we first need to log into docker and tag our image to match a created repository on Dockerhub.

```
s1155160788@user-desktop:/home/FinalProject$ sudo docker push eddiecfox/iems5709-final-project:v1
The push refers to repository [docker.io/eddiecfox/iems5709-final-project]
2ac025c6be0b: Pushed
b6bdb8c5b230: Pushed
d7bd04ab5f74: Layer already exists
0d3d7cd4b62b: Layer already exists
f1c58ee612be: Layer already exists
005d5e82999e: Layer already exists
1a21c1f7f460: Layer already exists
99b4412f35ff: Layer already exists
79ab9312bc97: Layer already exists
3ddbf7951ae8: Layer already exists
a79de234596b: Layer already exists
7bfcc1617253: Layer already exists
c7d52123c964: Layer already exists
d7735866217a: Layer already exists
126d1729cda6: Layer already exists
254d0c54e606: Layer already exists
7b67c523d9a7: Layer already exists
f5525955d20e: Layer already exists
8903b96a915b: Layer already exists
100e08d60496: Layer already exists
7d15b4fcff42: Layer already exists
9bae17f1b6fc: Layer already exists
7e2de379d5e2: Layer already exists
0588f50388e8: Layer already exists
v1: digest: sha256:2e22e80c9cdc01c957578f417b84e3883839485e04ce2404985cdcefd03da86a size: 5352
s1155160788@user-desktop:/home/FinalProject$
```

We then push this tagged image to Dockerhub. This needs to be done so that when we apply the yaml file later, Kubernetes can pull from the repository.

```
s1155160788@user-desktop:/home/FinalProject$ sudo kubectl apply -f AR-Deployment.yaml
deployment.apps/ar created
s1155160788@user-desktop:/home/FinalProject$ sudo kubectl apply -f AR-Service.yaml
service/ar created
s1155160788@user-desktop:/home/FinalProject$
```

After applying the two YAML files, the deployment and service are created in Kubernetes, and we took the Lens IDE screenshots shown above.

```
s1155160788@user-desktop:~$ sudo kubectl get pods
[sudo] password for s1155160788:
NAME                    READY   STATUS    RESTARTS   AGE
svclb-ar-zjj62          1/1     Running   0          15h
ar-577c6864c6-ch69x     1/1     Running   0          6h47m
s1155160788@user-desktop:~$
```

```
s1155160788@user-desktop:/home/FinalProject$ sudo kubectl exec -it ar-577c6864c6-5lp5x -- /bin/bash
root@ar-577c6864c6-5lp5x:/opt/nvidia/deepstream/deepstream-6.0# cd /home
root@ar-577c6864c6-5lp5x:/home# ls
AR-Deployment.yaml  Dockerfile  dataset                                deepstream_deployment.yaml  deepstream_service.yaml
AR-Service.yaml     conf.json   deepstream-3d-action-recognition-zyh  deepstream_pod.yaml
root@ar-577c6864c6-5lp5x:/home# cd deepstream-3d-action-recognition-zyh/
root@ar-577c6864c6-5lp5x:/home/deepstream-3d-action-recognition-zyh# ls
Fall.mp4                          config_preprocess_3d_custom.txt      deepstream_action_config_parse.cpp
Makefile                          custom_sequence_preprocess           deepstream_action_config_parse.o
README                            deepstream-3d-action-recognition     deepstream_action_recognition_config.txt
config_infer_primary_2d_action.txt deepstream_3d_action_recognition.cpp labels.txt
config_infer_primary_3d_action.txt deepstream_3d_action_recognition.o  resnet18_3d_rgb_hmdb5_32.etlt
config_preprocess_2d_custom.txt   deepstream_action.h                  resnet18_3d_rgb_hmdb5_32.etlt_b4_gpu0_fp16.engine
root@ar-577c6864c6-5lp5x:/home/deepstream-3d-action-recognition-zyh#
```

We exec into the pod created by the deployment and navigate to the bound folder, demonstrating proper volume binding / folder mounting. Unfortunately, we encountered a number of library errors when trying to execute the deepstream-3d-action-recognition app during Kubernetes. We are fairly confident that our Kubernetes commands are correct. The issue is that while troubleshooting Kubernetes earlier, we ran a mass upgrade through "apt-get upgrade" on the edge node which likely broke some of the links to the nvidia libraries due to version mismatches.

After doing this, we were unable to execute the deepstream-app anymore even on the containerized non Kubernetes version. The container never had all the libraries on their own. Rather, the necessary libraries are mounted through the nvidia container runtime, but we probably upgraded one of the components to a bugged version. If you are using this edge node, 192.168.85.61 for inferencing in the future, I would recommend flashing the board and reinstalling Jetpack, k3s, etc from scratch. Unfortunately, we do not have this level of access to the edge node.

When trying to run deepstream-app, even on the unmodified deepstream-l4t 6.0.1 triton image without Kubernetes, we now receive the same error Gary did in Lab 3: "Error when loading shared libraries: libnvinfer.so.8: cannot open shared object file: No such file or directory."

We hope that we have demonstrated enough understanding of containerization and Kubernetes. In this report, we were able to successfully execute the containerized version and create a Kubernetes deployment that should have the correct commands, stopped at the last step of execution due to library issues with the edge node.

References

[1] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6616496/

[2] https://catalog.ngc.nvidia.com/orgs/nvidia/teams/tao/models/actionrecognitionnet

[3] https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/

[4] http://serre-lab.clps.brown.edu/wp-content/uploads/2013/10/hmdb51_org.rar

[5] https://docs.nvidia.com/tao/tao-toolkit/text/action_recognition_net.html

[6] https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_3D_Action.html

[7] https://developer.nvidia.com/blog/developing-and-deploying-your-custom-action-recognition-application-without-any-ai-expertise-using-tao-and-deepstream/

[8] https://catalog.ngc.nvidia.com/orgs/nvidia/containers/deepstream-l4t

[9] https://docs.nvidia.com/metropolis/deepstream/DeepStream_4.0.1_Release_Notes.pdf

[10] https://blog.csdn.net/zhoubiaodi/article/details/122841928

[11] https://download.csdn.net/download/zhoubiaodi/79855297

[12] https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/user-guide.html

[13] https://www.suse.com/c/ai-at-the-edge-with-k3s-nvidia-jetson-nano-object-detection-real-time-video-analytics-src/

[14] https://stackoverflow.com/questions/31870222/how-can-i-keep-a-container-running-on-kubernetes

[15] https://stackoverflow.com/questions/49313809/kubernetes-docker-volume-mounting-option