Bitácora de trabajo

Martes 3 de setiembre: Investigación sobre DLStreamer basado en GStreamer. Intento de instalación pero da errores con ciertos paquetes, no encontré mucha documentación acerca de los problemas.
https://gstreamer.freedesktop.org/documentation/installing/on-linux.html?gi-language=c

Miércoles 4 de setiembre: Nuevamente investigación sobre la herramienta DLStreamer, aún presenta problemas en la instalación. De preferencia no se utilizará Dlstreamer (Gstreamer) por las dificultades presentadas. Búsqueda de información para utilizar OpenVINO y OpenCV, posiblemente utilizando YOLO.
https://dlstreamer.github.io/get_started/install/install_guide_ubuntu.html
Da problemas con las dependencias que se requieren instalar.

Miércoles 11 de setiembre: Investigación para generar imagen con Yocto para VirtualBox que contenga un hola mundo en Python. Pasos a seguir:

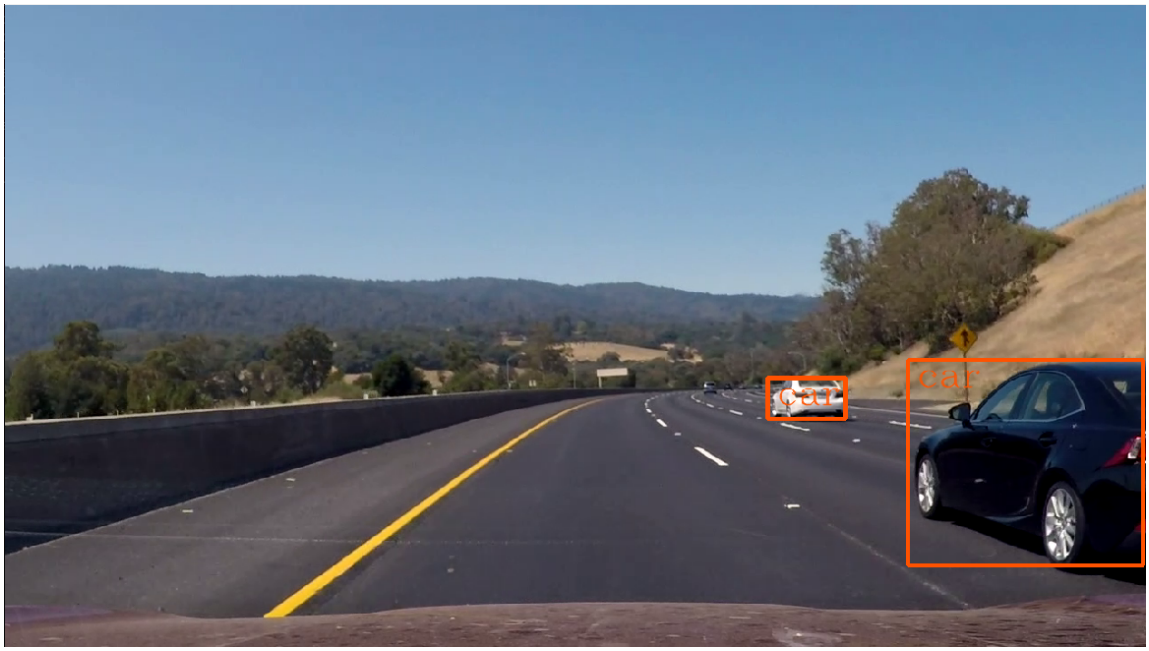
1) Clonar repositorio, utilizar Kirkstone e inicializar el ambiente
2) Abrir el archivo poky/build/conf/local.conf y agregar las líneas:
   IMAGE_INSTALL:append = " python3 python3-pip"
   IMAGE_INSTALL:append = " python-hello"
   IMAGE_FSTYPES += "iso"
3) Abrir el archivo poky/build/conf/bblayers.conf y agregar las líneas:
   BBFILES += "/home/tobi/poky/meta-misc/recipes-example/*/*.bb"
4) Generar carpeta donde estén los metas que se desean agregar:
   ```
   meta-misc/
     └── recipes-example/
       └── python-hello/
         ├── python-hello_1.0.bb
         └── files/
           └── hello.py
   ```
5) En el archivo python-hello_1.0.bb insertar lo siguiente:
   ```
   DESCRIPTION = "Hola Mundo en Python"
   LICENSE = "MIT"
   LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT"

   SRC_URI = "file://hello.py"

   S = "${WORKDIR}"

   do_install() {
      install -d ${D}${bindir}
      install -m 0755 ${WORKDIR}/hello.py ${D}${bindir}/hello.py
   }

   RDEPENDS_${PN} = "python3-core"
   ```
6) En el archivo hello.py agregar lo siguiente:
   ```
   #!/usr/bin/env python3
   print("Hola Mundo")
   ```

7) Finalmente, ejecutar bitbake core-image-minimal

El resultado es una imagen que imprime Hola mundo en la consola

Sábado 14 de setiembre:
Inicio de documentación del documento tutorial siguiendo la rúbrica brindada
Revisión de ejemplos de la página de OPENVino.
Mediante un ambiente de micromamba se ejecuta un ejemplo de detección de carros. El ambiente se configura con los paquetes necesarios:
$ micromamba install numpy
$ micromamba install -c conda-forge opencv
$ micromamba install -c conda-forge ocl-icd-system
$ micromamba install openvino=2024.3

se corre el archivo y da como resultado la siguiente salida:



Tareas por realizar:

- Investigar entrenamiento del modelo para detección de las rocas
- Una vez entrenado, ejecutar el script
- Si funciona, buscar metas necesarios para imagen de Yocto
- Cambiar archivos local.conf y bblayers.conf para incluir los metas de la app

Miércoles 18 de setiembre: Actividades realizadas
Entrenamiento de modelo de detección de rocas
De la página Roboflow descargo el dataset para entrenar el modelo deseado
https://universe.roboflow.com/nasarockyard/nasa-rockyard/dataset/5
En este caso se descarga Yolov8

Se entrena y exporta a OpenVINO con el siguiente código:

```
from ultralytics import YOLO
```

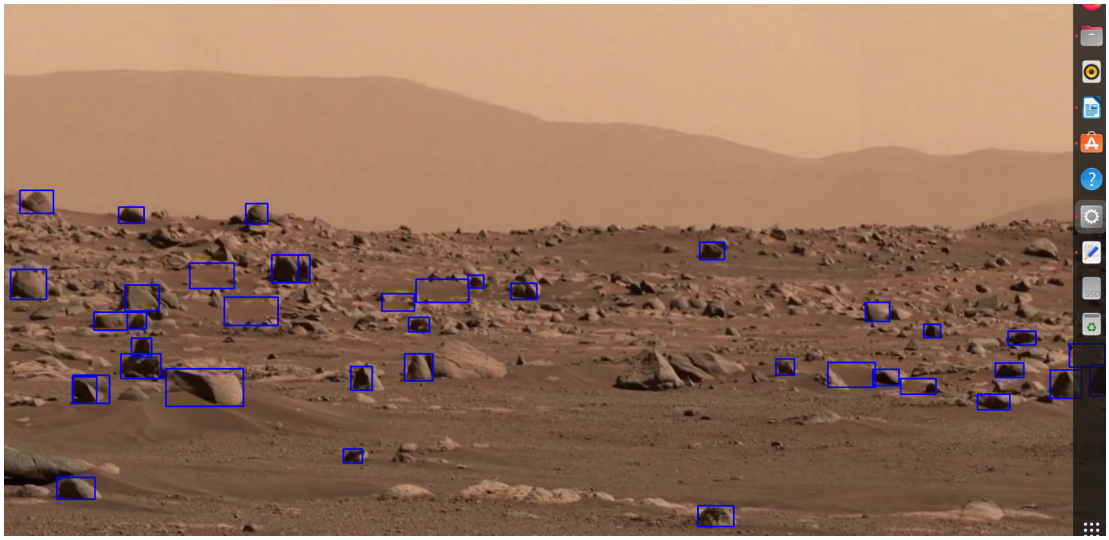
```
# Cargar el modelo preentrenado
model = YOLO("yolov8n.pt")

# Entrenar el modelo, especificando la carpeta de guardado
results = model.train(data="/home/tobi/Embebidos/Training/Rocks/data.yaml",
                      epochs=20)

# Evaluar el rendimiento del modelo
results = model.val()

# Realizar detección de objetos en una imagen
results = model("/home/tobi/Embebidos/Training/Rocks/test.jpg")

# Exportar el modelo a formato openvino y guardarlo en la carpeta especificada
success = model.export(format="openvino")
```

Del cual se obtienen un archivo .xml y otro .bin

Al terminar el entrenamiento, se intenta ejecutar un programa de detección y que dibuje bounding boxes pero el resultado no es el esperado. Hay ciertos problemas con las dimensiones del modelo y el Tensor. El código requiere que el tamaño sea de 300x300 pero el modelo entrenado utiliza 640x640, se intenta realizar las transformaciones necesarias pero sin resultados satisfactorios.

Conversando con el profesor, existe la opción de ejecutar otra app que sirva para continuar con el proceso de la creación de imagen, rebajando los puntos correspondientes por no utilizar una aplicación acorde a lo solicitado.

Sábado 21 de setiembre: nuevamente se entrena el modelo 30 épocas pero esta vez es exportado a formato ONNX y se utiliza el código extraído del repositorio de ultralytics. Da buen resultado:



Al buscar los archivos necesarios para integrar el programa en la imagen de Yocto, se obtienen metas de OpenEmbedded. Los metas necesarios específicamente para Ultralytics son:

- openembedded-core
- meta-python
- meta-oe
- meta-clang-revival

Al intentar la creación de la imagen, dos problemas resultaron: el primero fue la cantidad de almacenamiento que requería la imagen excedió la capacidad disponible en el disco, por lo que no se pudo seguir construyendo. El segundo, fue errores en los bblayers, los cuales indicaban que no había una dependencia cuando sí estaba incluída dentro de los archivos de configuración.

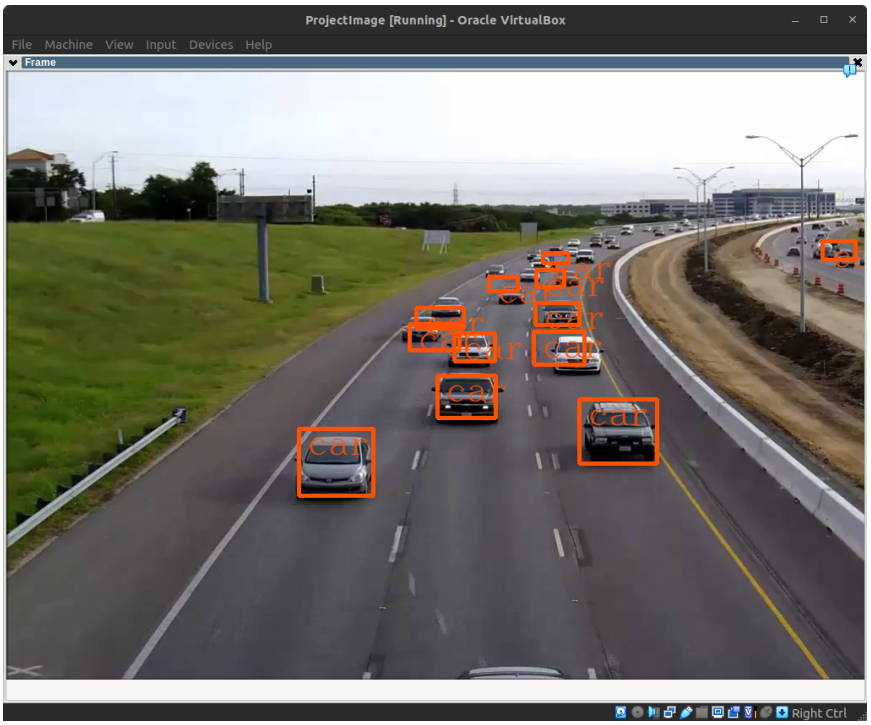
Por tanto, procedí a construir una imagen con la detección de carros dado el poco tiempo disponible antes de la entrega. Esta imagen cumple con la preferencia de utilizar OpenVINO y OpenCV, además usa Gstreamer por debajo.
Los metas utilizados son:
media/tobi/Yocto/poky/meta \
/media/tobi/Yocto/poky/meta−poky \
/media/tobi/Yocto/poky/meta−yocto−bsp \
/media/tobi/Yocto/poky/meta−openembedded/meta−oe \
/media/tobi/Yocto/poky/meta−openembedded/meta−python \
/media/tobi/Yocto/poky/meta−openembedded/meta−multimedia \
/media/tobi/Yocto/poky/meta−intel \
/media/tobi/Yocto/poky/meta−mylayer \

En meta-mylayer se incluyen los archivos para la aplicación, incluyendo scripts de Python y video de prueba.

Además, para evitar exceder el almacenamiento, se opta por utilizar un disco externo con suficiente memoria disponible. Por lo tanto, el proceso de construcción se lleva acabo en media/tobi/Yocto, donde Yocto es el nombre del disco. Se agregan las dependencias en local.conf y el tipo de imagen “wic.vmdk” para la imagen en VirtualBox. La construcción tomó hasta el día domingo para finalizarse, incluyendo varios errores de código 1, por lo que basta con volver a ejecutar el comando de bitbake para que siga la construcción.

https://docs.ultralytics.com/integrations/onnx/#supported-deployment-options
https://colab.research.google.com/github/openvinotoolkit/openvino_notebooks/blob/main/notebooks/401-object-detection-webcam/401-object-detection.ipynb#Table-of-contents:

Domingo 22 de setiembre:
Al finalizar, la construcción pesa alrededor de 192 GB. Se procede a crear la máquina virtual para probar la imagen, dando resultados satisfactorios.



Por último, se procede a la creación del documento tutorial siguiendo las especificaciones indicadas en la rúbrica.