Getting Started with YOLO on Modalix

Getting started with the YOLOv8m model on Modalix

1. Purpose and Audience

This guide is a step-by-step hands-on guide for customers to build and run an AI inference pipeline on SiMa.ai's Modalix Edge AI platform using the Ultralytics YOLOv8m model as an example. By following this document, customers will experience the entire process in one go: setting up the SDK environment, converting the model, creating an mpk, and running it on Modalix.

Audience: New Modalix users, partner engineers, PoC owners, etc.

2. Prerequisites and Requirements

- Modalix SoM DevKit or Engineering EVB
- A development PC running Ubuntu 22.04 or later
- Palette SDK 1.7 installed
- RJ45 wired network connection between Modalix and the development PC (before using the SDK, connect via the serial port and manually set the Modalix IP)
- 💡 IP address configuration

```
Development PC (Host): 192.168.1.10
Modalix: 192.168.1.20
```

```
ffmpeg (host) → `rtsp://127.0.0.1:8554/mystream1`
From container to RTSP → host LAN IP (e.g., `192.168.1.10`)
Result stream → Modalix → host **UDP:5000**
```

- This IP layout is the default used in the example commands and can be adjusted to fit your actual environment.
- 💡 Note: Docker, FFmpeg, and GStreamer are already included in the Palette SDK Docker image. No separate installation is required.

2-1. Network & Port Summary

- Network and Port

Path	Protocol	Port	Notes
ffmpeg → mediamtx	TCP	8554	`-rtsp_transport tcp`

Path	Protocol	Port	Notes
Container → RTSP (host IP)	TCP	8554	127.0.0.1 prohibited, use **LAN IP**
Modalix → Host	UDP	5000	`gst udpsrc port=5000`

- ☑ Pre-checklist

```
Run all commands inside the Docker container. (Except Steps 11–14, 18–19
which are outside Docker.)
RTSP address: container → host IP (e.g., 192.168.1.10) / host → 127.0.0.1
Model/resolution: imgsz=640, input stream 1920×1080
```

3. YOLOv8m Hands-on Guide: From Model Conversion to Running on Modalix

This section describes the end-to-end process to create an mpk for Modalix using the Ultralytics YOLOv8m model and to run it on the actual device.

Step 1. Open a Terminal and Start the SDK Docker

- Open a new terminal in Ubuntu. (Terminal 1)
- Run start.py to launch the Palette SDK Docker environment.
- start.py is installed with Palette SDK and is the recommended way to run Docker.
- For example, if you installed it in the folder below, move to that folder and run it.

```
cd ~/palette/1.7.0_Palette_SDK_master_B219/sima-cli
python3 start.py
```

```
/home/howard-lee/palette/1.7.0_Palette_SDK_master_B219/sima-
cli/start.py:111: SyntaxWarning: invalid escape sequence '\P'
docker_start_cmd = 'cmd.exe /c "start "" "C:\Program
Files\Docker\Docker\Docker Desktop.exe""'
Set no_proxy to localhost,127.0.0.0
Using port 49152 for the installation.
Checking if the container is already running...
==> Starting the stopped container: palettesdk_1_7_0_Palette_SDK_master_B219
palettesdk_1_7_0_Palette_SDK_master_B219
howard-lee@38bc47e7a4ec:/home$
```

- The Palette SDK is provided as a Docker-based environment; running the above command starts Docker.

Step 1-1. Set Host–Container Shared Directory

- When using the SDK, bind-mount a local workspace directory into a path inside the container so you can easily share files between the host and the Docker container.

```
For example:
Host: /home/howard-lee/workspace_1.7.0
Container: /home/docker/sima-cli
```

- These two paths refer to the same physical directory; changes on one side immediately reflect on the other. With this structure you can exchange models, scripts, and config files without extra copy steps.
- Note: Folder names and paths can be freely chosen during SDK setup.

Step 2. Install sima-model-to-pipeline (Inside Docker, Terminal 1)

- If Internet access is available, installation and execution can be done through GitHub. If not, installation can be done using the provided ZIP package.

Step 2-1. Installation and Execution via GitHub (Internet Access Required)

- Move into Docker.

```
howard-lee@38bc47e7a4ec:/home$ ls -al
```

```
drwxr-xr-x  1 root       root       4096 Aug 11 19:31 .
drwxr-xr-x  1 root       root       4096 Aug 11 19:31 ..
drwxrwxrwx  1 docker     docker     4096 Sep 12 23:12 docker
drwxr-xr-x 12 howard-lee howard-lee 4096 Oct  2 21:39 howard-lee
```

```
cd /home/docker/sima-cli
```

- Run the installation command. Once the message “Successfully installed” appears, the tool is ready to use.

```
sima-cli install gh:sima-ai/tool-model-to-pipeline
```

```
Successfully installed sima-model-to-pipeline-0.1.0
```

- Edit your YAML configuration file and run the following command. Example YAML files can be found in the `samples` directory.

```
sima-model-to-pipeline model-to-pipeline --config-yaml ./yolov8m.yaml
```

- When completed, the following summary table will appear. Verify that all steps show **PASS**.

```
    SiMa.ai Model to Pipeline Summary
┌─────────────────┬──────────────┬──────────────┐
│   Step Name     │ Elapsed Time │    Status    │
├─────────────────┼──────────────┼──────────────┤
│ downloadmodel   │     PASS     │  0.10 sec    │
│ surgery         │     PASS     │  3.90 sec    │
│ downloadcalib   │     PASS     │  0.11 sec    │
│ compile         │     PASS     │ 280.19 sec   │
│ pipelinecreate  │     PASS     │  1.31 sec    │
│ mpkcreate       │     PASS     │ 120.85 sec   │
└─────────────────┴──────────────┴──────────────┘
                  Summary

☑ Process completed, press any key to exit...
Shutting down Flask server...
```

- If an error occurs, check the `logs` folder for the failed step's log file.

```
downloadmodel_1759927083.1472435.log
surgery_1759927093.1744452.log
downloadcalib_1759927097.5104256.log
compile_1759927097.6145797.log
pipelinecreate_1759927375.0648186.log
mpkcreate_1759927376.373405.log
```

- If all steps pass successfully, verify that the generated `project.mpk` file is available under the `yolov8m_simaaisrc` directory.

```
cd ./yolov8m_simaaisrc
```

- The following files should be visible in the directory:

```
.project
build
core
dependencies
plugins
```

```
resources
.hash
application.json
application_pcie.json
application_rtsp.json
project.mpk
yolov8m_simaaisrc_plugin_version_index.json
```

- To run the generated `project.mpk` file on the Modalix board, go directly to **Step 11**.
- If you prefer to execute each stage manually instead of using a YAML file, continue to **Step 4**.

Step 2-2. Move to the mpktool Package Directory (Without Internet Connection)

- Move into Docker.

```
howard-lee@38bc47e7a4ec:/home$ ls -al
```

```
drwxr-xr-x  1 root       root       4096 Aug 11 19:31 .
drwxr-xr-x  1 root       root       4096 Aug 11 19:31 ..
drwxrwxrwx  1 docker     docker     4096 Sep 12 23:12 docker
drwxr-xr-x 12 howard-lee howard-lee 4096 Oct  2 21:39 howard-lee
```

```
cd /home/docker/sima-cli
```

- From outside Docker, copy `mpktool_package_sdk1.7.zip` into the internal folder

```
Downloads$ cp mpktool_package_sdk1.7.zip /home/howard-lee/workspace_1.7.0
```

```
ls -al mpktool_package_sdk1.7.zip
-rw-rw-r-- 1 howard-lee howard-lee 15948815 Oct  5 01:41
mpktool_package_sdk1.7.zip
```

- Unzip `mpktool_package_sdk1.7.zip` inside Docker.

```
unzip mpktool_package_sdk1.7.zip
```

- Move into the extracted folder.

```
cd ./mpktool_package
```

```
howard-lee@38bc47e7a4ec:/home/docker/sima-cli/mpktool_package$ ls -al
total 100
drwxrwxr-x 5 howard-lee howard-lee  4096 Aug 24 22:59 .
drwxrwxr-x 9 howard-lee howard-lee  4096 Oct  5 02:13 ..
drwxrwxr-x 3 howard-lee howard-lee  4096 Aug  4 08:33 code
drwxrwxr-x 5 howard-lee howard-lee  4096 Jul 28 03:01 cpp_detection_app
drwxrwxr-x 5 howard-lee howard-lee  4096 Aug 24 22:51 python_app
-rw-rw-r-- 1 howard-lee howard-lee  5766 Aug 24 22:58 readme.md
-rw-r--r-- 1 howard-lee howard-lee 73621 Aug 24 22:57
sima_model_to_pipeline-sdk-1.7.tar.gz
```

Step 3. Install sima-model-to-pipeline via ZIP File (Inside Docker, Terminal 1)

- Now install the sima-model-to-pipeline tool.

```
sudo pip3 install sima_model_to_pipeline-sdk-1.7.tar.gz
```

- You may see warning messages during installation; that's fine as long as installation completes. If the installation stops, attach the logs and contact the support channel.

```
Successfully installed sima-model-to-pipeline-0.1.0
```

- If you need to remove it later, use the two commands below.

```
pip uninstall -y sima-model-to-pipeline
```

```
Found existing installation: sima-model-to-pipeline 0.1.0
Uninstalling sima-model-to-pipeline-0.1.0:
Successfully uninstalled sima-model-to-pipeline-0.1.0
```

```
sudo rm /usr/local/bin/sima-model-to-pipeline
```

- After removal, confirm it's fully gone.

```

```

```
pip list | grep sima
which sima-model-to-pipeline
```

Step 4. Install Ultralytics (Inside Docker, Terminal 1)

- Install the `ultralytics` package to export a PyTorch (.pt) model to ONNX.

```
pip3 install ultralytics
```

```
To verify installation, type 'yolo'.
howard-lee@38bc47e7a4ec:/home/docker/sima-cli/mpktool_package$ yolo
Arguments received: ['yolo']. Ultralytics 'yolo' commands use the following
syntax:
```

Step 5. Create a YOLO Example Directory (Inside Docker, Terminal 1)

- You received two Python files and one mp4 file with this document.

```
exam_surgery.py: Graph surgery example implemented in Python
exam_compile.py: Quantization/Compile example implemented in Python
example.mp4: Example video file 1920×1080@30fps
```

- Create a directory for the experiment.

```
mkdir -p yolo
cd yolo
```

- Copy only the two Python files (`exam_surgery.py`, `exam_compile.py`) into the `yolo` folder.
- Copy them in the same way you copied `sima_model_to_pipeline-sdk-1.7.tar.gz` into Docker.
- Keep `example.mp4` outside the Docker container because it will be streamed via ffmpeg on the host PC.

Step 6. Download the YOLOv8m Model (Inside Docker, Terminal 1)

- Download the YOLOv8m example model from Ultralytics.
- You can also search Google for ‘yolov8 ultralytics download’ and download it.

```
wget
https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8m.pt
```

```
--2025-10-05 02:45:58--
https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8m.pt
Resolving github.com (github.com)... 20.200.245.247
Connecting to github.com (github.com)|20.200.245.247|:443... connected.
HTTP request sent, awaiting response... 302 Found
...
2025-10-05 02:46:07 (6.24 MB/s) - ‘yolov8m.pt’ saved [52136884/52136884]
```

```
howard-lee@38bc47e7a4ec:/home/docker/sima-cli/mpktool_package/yolo$ ls -al
total 50940
drwxr-xr-x 2 howard-lee howard-lee     4096 Oct  5 02:45 .
drwxrwxr-x 6 howard-lee howard-lee     4096 Oct  5 02:43 ..
-rwxr-xr-x 1 howard-lee howard-lee     6700 Oct  4 13:16 exam_compile.py
-rwxr-xr-x 1 howard-lee howard-lee     4982 Oct  4 13:16 exam_surgery.py
-rw-r--r-- 1 howard-lee howard-lee 52136884 Sep 29  2024 yolov8m.pt
```

Step 7. Export the PyTorch (.pt) Model to ONNX (Inside Docker, Terminal 1)

- The downloaded model is `yolov8m.pt`.
- Export it to ONNX format using the `yolo` command.
- The following command creates `yolov8m.onnx` in the same folder.

```
yolo export model=yolov8m.pt format=onnx imgsz=640 opset=13 dynamic=False
simplify=False nms=False
```

```
Ultralytics 8.3.145 🚀 Python-3.10.12 torch-2.8.0+cu128 CPU (Intel Core(TM)
Ultra 9 185H)
YOLOv8m summary (fused): 92 layers, 25,886,080 parameters, 0 gradients, 78.9
GFLOPs

PyTorch: starting from 'yolov8m.pt' with input shape (1, 3, 640, 640) BCHW
and output shape(s) (1, 84, 8400) (49.7 MB)

ONNX: starting export with onnx 1.17.0 opset 13...
ONNX: export success ☑ 1.0s, saved as 'yolov8m.onnx' (99.0 MB)

Export complete (1.7s)
```

```
yolo
├── yolov8m.pt
```

```
├── yolov8m.onnx
├── exam_surgery.py
└── exam_compile.py
```

Step 8. Graph Surgery (Modify Model Structure if Needed) (Inside Docker, Terminal 1)

- Graph Surgery refers to directly editing the node structure of an ONNX graph, and removing/merging/converting unnecessary ops; it's a widely used process in AI compilers and runtimes.
- Run graph surgery with the `step surgery` option.
- The input image size used here is 1920×1080.
- The following command creates `yolov8m_pipeline_mod.onnx` in the same folder.

```
sima-model-to-pipeline model-to-pipeline \
--model-path ./yolov8m.onnx \
--model-name yolov8 \
--pipeline-name yolov8m_pipeline \
--input-width 1920 \
--input-height 1080 \
--rtsp-src rtsp://192.168.1.10:8554/mystream1 \
--host-ip 192.168.1.10 \
--host-port 5000 \
--detection-threshold 0.1 \
--nms-iou-threshold 0.3 \
--topk 100 \
--device-type modalix \
--step surgery
```

```
☑ Completed : setup               : 0.10 sec
☑ Completed : surgery             : 4.29 sec


   SiMa.ai Model to Pipeline Summary
   ┌──────────────┬──────────────┬──────────┐
   │ Step Name    │ Elapsed Time │  Status  │
   ├──────────────┼──────────────┼──────────┤
   │  surgery     │     PASS     │ 4.29 sec │
   └──────────────┴──────────────┴──────────┘
                Summary
```

```
howard-lee@38bc47e7a4ec:/home/docker/sima-cli/mpktool_package/yolo$ ls -al
total 253624
drwxr-xr-x 3 howard-lee howard-lee      4096 Oct  5 02:55 .
drwxrwxr-x 6 howard-lee howard-lee      4096 Oct  5 02:43 ..
-rwxr-xr-x 1 howard-lee howard-lee      6700 Oct  4 13:16 exam_compile.py
```

```
-rwxr-xr-x 1 howard-lee howard-lee      4982 Oct  4 13:16 exam_surgery.py
drwxr-xr-x 2 howard-lee howard-lee      4096 Oct  5 02:54 logs
-rw-r--r-- 1 howard-lee howard-lee 103773663 Oct  5 02:49 yolov8m.onnx
-rw-r--r-- 1 howard-lee howard-lee 103762033 Oct  5 02:55
yolov8m_pipeline_mod.onnx
-rw-r--r-- 1 howard-lee howard-lee  52136884 Sep 29  2024 yolov8m.pt
```

- After running YOLOv8m, it is configured to send the result to host (192.168.1.10) port 5000.
- RTSP (Real Time Streaming Protocol) is a network protocol for controlling real-time audio/video streaming.
- 💡 Since the container accesses the host RTSP server, use the host's LAN IP (e.g., 192.168.1.10) for the URL inside the container instead of 127.0.0.1. This corresponds to the RTSP IP address (127.0.0.1) used by ffmpeg in Step 14 when streaming from the host.
- In the above command, `device-type` must be set to `davinci` or `modalix`.

Option	Meaning	Target
`--device-type davinci`	Gen1	DaVinci EVB
`--device-type modalix`	Gen2	Modalix SoM/EVB

- `detection-threshold`, `nms-iou-threshold`, and `topk` are YOLO performance-tuning parameters.

```
- detection_threshold: 0.1
Minimum confidence score a detection must have to be kept.
Example: if a bounding box has only 0.1 confidence, it gets discarded.
Higher = fewer false positives (but more missed detections). Lower = more
detections (but noisier).
```

```
- nms_iou_threshold: 0.3
Controls Non-Maximum Suppression (NMS).
After YOLO outputs many overlapping boxes, NMS keeps the strongest and
removes others if the overlap (IoU) is greater than 0.3.
Lower value (e.g., 0.1) → very aggressive suppression (may remove
neighboring objects).
Higher value (e.g., 0.7) → keeps more boxes, but may leave duplicates.
```

```
- topk: 100
Maximum number of detections to keep after NMS.
If the model finds 200 candidate boxes, only the top 100 (by confidence)
survive.
Important in dense scenes (traffic, crowds) so you don't lose valid objects
just because of a low cap.
```

Step 8-1. Output When Running exam_surgery.py

- If you run `exam_surgery.py` instead of using `sima-model-to-pipeline`, the output looks like this:

```
┌────────────┬──────────────────┬──────────────────────┐
│            │ Original Model   │ Simplified Model     │
└────────────┴──────────────────┴──────────────────────┘
┌────────────┬──────────────────┬──────────────────────┐
│ Add        │ 15               │ 15                   │
│ Concat     │ 16               │ 16                   │
│ Constant   │ 213              │ 196                  │
│ Conv       │ 107              │ 107                  │
│ MaxPool    │ 3                │ 3                    │
│ Mul        │ 77               │ 77                   │
│ Resize     │ 2                │ 2                    │
│ Sigmoid    │ 80               │ 80                   │
│ Softmax    │ 12               │ 12                   │
│ Split      │ 8                │ 8                    │
│ Model Size │ 99.1MiB          │ 99.0MiB              │
└────────────┴──────────────────┴──────────────────────┘
ONNX file saved to /home/docker/sima-
cli/mpktool_package/yolo/yolov8m_pipeline_mod.onnx
[OK] Surgery ONNX created: /home/docker/sima-
cli/mpktool_package/yolo/yolov8m_pipeline_mod.onnx
```

- Confirm that `yolov8m_pipeline_mod.onnx` is created.

```
ls -lh ./yolov8m_pipeline_mod.onnx
```

Step 9. Compile (Including Quantization) (Inside Docker, Terminal 1)

- After Graph Surgery, perform Quantization and Compile together.
- Provide the path to the calibration images required for quantization.
- When installing the Palette SDK, calibration images are copied into the Docker environment.
- Define the precision required for quantization—INT8, INT16, BF16 are available. Considering both performance and accuracy, INT8 is used in most cases.
- This document also uses INT8. `sima-model-to-pipeline` uses INT8 by default.
- The following command creates `result/modalix/yolov8m_pipeline.tar.gz` in the same folder.
- 💡 Even if `yolov8m_pipeline_mod.onnx` is created in the Graph Surgery stage, the input model for `sima-model-to-pipeline` in Step 9 (compile) and Step 10 (pipelinecreate) is `yolov8m.onnx`.

```
sima-model-to-pipeline model-to-pipeline \
--model-path yolov8m.onnx \
```

```
--model-name yolov8 \
--pipeline-name yolov8m_pipeline \
--input-width 1920 \
--input-height 1080 \
--rtsp-src rtsp://192.168.1.10:8554/mystream1 \
--host-ip 192.168.1.10 \
--host-port 5000 \
--detection-threshold 0.1 \
--nms-iou-threshold 0.3 \
--topk 100 \
--calibration-data-path /home/docker/calibration_images \
--device-type modalix \
--step compile
```

```
☑ Completed : setup               : 0.10 sec
☑ Completed : compile             : 307.06 sec


    SiMa.ai Model to Pipeline Summary
┌─────────────┬──────────────────┬──────────────┐
│ Step Name   │ Elapsed Time     │   Status     │
├─────────────┼──────────────────┼──────────────┤
│  compile    │      PASS        │ 307.07 sec   │
└─────────────┴──────────────────┴──────────────┘
                Summary
```

```
howard-lee@38bc47e7a4ec:/home/docker/sima-cli/mpktool_package/yolo$ ls -al
total 253628
drwxr-xr-x 4 howard-lee howard-lee      4096 Oct  5 03:19 .
drwxrwxr-x 6 howard-lee howard-lee      4096 Oct  5 02:43 ..
-rwxr-xr-x 1 howard-lee howard-lee      6700 Oct  4 13:16 exam_compile.py
-rwxr-xr-x 1 howard-lee howard-lee      4982 Oct  4 13:16 exam_surgery.py
drwxr-xr-x 2 howard-lee howard-lee      4096 Oct  5 03:17 logs
drwxr-xr-x 3 howard-lee howard-lee      4096 Oct  5 03:19 result
-rw-r--r-- 1 howard-lee howard-lee 103773663 Oct  5 02:49 yolov8m.onnx
-rw-r--r-- 1 howard-lee howard-lee 103762033 Oct  5 02:55
yolov8m_pipeline_mod.onnx
-rw-r--r-- 1 howard-lee howard-lee  52136884 Sep 29  2024 yolov8m.pt
```

```
howard-lee@38bc47e7a4ec:/home/docker/sima-cli/mpktool_package/yolo$ ls -al
result/modalix/yolov8m_pipeline.tar.gz
-rw-r--r-- 1 howard-lee howard-lee 30735879 Oct  5 03:22
result/modalix/yolov8m_pipeline.tar.gz
```

Step 9-1. Output When Running exam_compile.py

- Output when running `exam_compile.py` instead of `sima-model-to-pipeline`:

```
SiMa Model and Quantization Details
...
Calibration progress: completed 100 samples
Running Calibration ...DONE
Running quantization ...DONE
[INFO] Compiling quantized net "yolov8"
[INFO] Allocating memory for IFM/OFM tensors
... (many detailed logs for tile/memory placement) ...
[OK] Compilation finished.
Requested artifact path: /home/docker/sima-
cli/mpktool_package/yolo/result/yolov8m_pipeline.tar.gz
```

- Confirm that `yolov8m_pipeline.tar.gz` is created.

```
ls -lh ./result/modalix/yolov8m_pipeline.tar.gz
```

Step 10. Create the Pipeline (Inside Docker, Terminal 1)

- Now we create the pipeline to run on Modalix.
- On an edge device, a pipeline refers not just to the model, but to the entire processing flow that includes the model.
- In other words, it bundles the steps—input → pre-processing → inference → post-processing → output—into a single linear flow.
- This step generates the input files required to create `project.mpk` for execution on Modalix.
- Running the command below creates the folder `yolov8m_pipeline_simaaisrc` in the same location.

```
sima-model-to-pipeline model-to-pipeline \
--model-path yolov8m.onnx \
--model-name yolov8 \
--pipeline-name yolov8m_pipeline \
--input-width 1920 \
--input-height 1080 \
--rtsp-src rtsp://192.168.1.10:8554/mystream1 \
--host-ip 192.168.1.10 \
--host-port 5000 \
--detection-threshold 0.1 \
--nms-iou-threshold 0.3 \
--topk 100 \
--device-type modalix \
--step pipelinecreate
```

```
☑ Completed : setup               : 0.10 sec
☑ Completed : pipelinecreate      : 1.30 sec


     SiMa.ai Model to Pipeline Summary
┌──────────────────┬──────────────┬──────────┐
│   Step Name      │ Elapsed Time │  Status  │
├──────────────────┼──────────────┼──────────┤
│ pipelinecreate   │     PASS     │ 1.31 sec │
└──────────────────┴──────────────┴──────────┘
                  Summary
```

```
howard-lee@38bc47e7a4ec:/home/docker/sima-cli/mpktool_package/yolo$ ls -al
total 253632
drwxr-xr-x 5 howard-lee howard-lee      4096 Oct  5 03:32 .
drwxrwxr-x 6 howard-lee howard-lee      4096 Oct  5 02:43 ..
-rwxr-xr-x 1 howard-lee howard-lee      6700 Oct  4 13:16 exam_compile.py
-rwxr-xr-x 1 howard-lee howard-lee      4982 Oct  4 13:16 exam_surgery.py
drwxr-xr-x 2 howard-lee howard-lee      4096 Oct  5 03:32 logs
drwxr-xr-x 3 howard-lee howard-lee      4096 Oct  5 03:19 result
-rw-r--r-- 1 howard-lee howard-lee 103773663 Oct  5 02:49 yolov8m.onnx
-rw-r--r-- 1 howard-lee howard-lee 103762033 Oct  5 02:55
yolov8m_pipeline_mod.onnx
drwxr-xr-x 7 howard-lee howard-lee      4096 Oct  5 03:32
yolov8m_pipeline_simaaisrc
-rw-r--r-- 1 howard-lee howard-lee  52136884 Sep 29  2024 yolov8m.pt
```

- `yolov8m_pipeline_simaaisrc` contains the input files and directories needed to generate a pipeline executable on Modalix.
- The structure is quite complex, including JSON files, so the Python implementation is omitted here.

```
howard-lee@38bc47e7a4ec:/home/docker/sima-cli/mpktool_package/yolo$ ls -al
yolov8m_pipeline_simaaisrc/
total 56
drwxr-xr-x 7 howard-lee howard-lee 4096 Oct  5 03:32 .
drwxr-xr-x 5 howard-lee howard-lee 4096 Oct  5 03:32 ..
-rw-r--r-- 1 howard-lee howard-lee 5187 Oct  5 03:32 application.json
-rw-r--r-- 1 howard-lee howard-lee 4608 Oct  5 03:32 application_pcie.json
-rw-r--r-- 1 howard-lee howard-lee 5187 Oct  5 03:32 application_rtsp.json
drwxr-xr-x 9 howard-lee howard-lee 4096 Aug  6 18:03 core
drwxr-xr-x 3 howard-lee howard-lee 4096 Oct  5 03:32 dependencies
-rw-r--r-- 1 howard-lee howard-lee   32 Oct  5 03:32 .hash
drwxr-xr-x 8 howard-lee howard-lee 4096 Oct  5 03:32 plugins
drwxr-xr-x 2 howard-lee howard-lee 4096 Oct  5 03:32 .project
drwxr-xr-x 2 howard-lee howard-lee 4096 Oct  5 03:32 resources
```

Step 11. Open a New Terminal (Outside Docker, Terminal 2)

- Now it's time to run the RTSP server and video streaming outside the Docker container.
- Open a new terminal in Ubuntu (Terminal 2).

Step 12. Run the RTSP Server (Outside Docker, Terminal 2)

- Run the RTSP server with the following command.

```
docker run --name rtsp_server --rm -e MTX_PROTOCOLS=udp,tcp -p 8554:8554
bluenviron/mediamtx
```

Step 13. Open a New Terminal (Outside Docker, Terminal 3)

- Open another terminal (Terminal 3) in Ubuntu to transmit the example video stream.

Step 14. Transmit the Example Video Stream (Outside Docker, Terminal 3)

- Now you need to stream the example video file to the RTSP server.
- Ensure the `example.mp4` file exists at the given location. The size is 1920×1080.

```
ffmpeg -re -nostdin -stream_loop -1 -i ./example.mp4 -c:v copy -f rtsp -
rtsp_transport tcp rtsp://127.0.0.1:8554/mystream1
```

- 💡 Note: In the ffmpeg command, the RTSP address uses 127.0.0.1, but inside the Modalix Docker container you must access it via the host's IP (e.g., 192.168.1.10).

Step 15. Return to the Inside of Docker (Inside Docker, Terminal 1)

- You are now ready to create the `project.mpk` file. This file runs on Modalix.
- Return to the SDK Docker terminal (Terminal 1).

Step 16. Build the mpk (Inside Docker, Terminal 1)

- The folder `yolov8m_pipeline_simaaisrc` already contains the necessary files and folders generated in the previous step.
- Move to that folder and run the `mpk create` command as below.
- If finished without issues, `project.mpk` is created in the same location.

```
cd yolov8m_pipeline_simaaisrc
mpk create -s . -d . --clean --board-type modalix
```

```
ℹ Cleaning up build artifacts...
✔ Successfully cleaned up build artifacts.
ℹ No changes detected in the a65-apps source code. Skipping compilation.
ℹ Compiling Plugins...
✔ Plugins Compiled successfully.
ℹ Processing Plugins...
ℹ Checking plugins info before writing...
ℹ Getting plugins info from cmake files...
✔ App yolov8m_pipeline_simaaisrc: Plugin version info written to
'/home/docker/sima-
cli/mpktool_package/yolo/yolov8m_pipeline_simaaisrc/yolov8m_pipeline_simaais
rc_plugin_version_index.json'
Successfully.
✔ Processing Plugins is Successful.
ℹ Copying Resources...
✔ Resources Copied successfully.
ℹ Building Rpm...
✔ Rpm built successfully.
ℹ Creating mpk file...
✔ Mpk file created successfully at /home/docker/sima-
cli/mpktool_package/yolo/yolov8m_pipeline_simaaisrc/project.mpk .
✔ Total time taken is 67.772 seconds.
```

Step 17. Deploy to and Run on Modalix (Inside Docker, Terminal 1)

- Now you need to deploy the generated `project.mpk` file to Modalix.
- Power on the Modalix device.
- Check if Modalix has finished booting with the following command.

```
ping 192.168.1.20
```

```
howard-lee@38bc47e7a4ec:/home/docker/sima-
cli/mpktool_package/yolo/yolov8m_pipeline_simaaisrc$ ping 192.168.1.20
PING 192.168.1.20 (192.168.1.20) 56(84) bytes of data.
64 bytes from 192.168.1.20: icmp_seq=1 ttl=63 time=0.417 ms
64 bytes from 192.168.1.20: icmp_seq=2 ttl=63 time=0.307 ms
64 bytes from 192.168.1.20: icmp_seq=3 ttl=63 time=0.252 ms
64 bytes from 192.168.1.20: icmp_seq=4 ttl=63 time=0.305 ms
```

- When ping responds, connect to the device. A password may be required; it is `edgeai`.

```
mpk device connect -t sima@192.168.1.20
```

```
ℹ Connecting to sima@192.168.1.20...
🔗 Connection established to 192.168.1.20 .
ℹ Fetching Device Plugin Version data file from : 192.168.1.20 ...
✔ Successfully fetched and updated Plugin Version Data file from :
192.168.1.20.
```

- After the connection is successful, deploy `project.mpk` with the command below.

```
mpk deploy -f project.mpk
```

```
ℹ Checking if App yolov8m_pipeline_simaaisrc Plugin Version Index File
/home/docker/sima-
cli/mpktool_package/yolo/yolov8m_pipeline_simaaisrc/yolov8m_pipeline_simaais
rc_plugin_version_index.json exists...
ℹ File Exists, proceeding with Version Check...
  Match found for 'simaaiprocessmla': {'name': 'simaaiprocessmla',
'version': '1.7'} -> Matched
  Match found for 'simaaisrc': {'name': 'simaaisrc', 'version': '1.7'} ->
Matched
  Match found for 'simaaioverlay2': {'name': 'simaaioverlay2', 'version':
'1.7'} -> Matched
  Match found for 'simaaiprocesscvu': {'name': 'simaaiprocesscvu',
'version': '1.7'} -> Matched
  Match found for 'simaaiboxdecode': {'name': 'simaaiboxdecode', 'version':
'1.7'} -> Matched

ℹ Summary of App yolov8m_pipeline_simaaisrc Vs Device Plugin Version Info
Comparison
✔ App yolov8m_pipeline_simaaisrc Overall Match: TRUE - All the local
Version items found a corresponding match. Continuing with mpk
deploy ...

🚀 Sending MPK to 192.168.1.20...
Transfer Progress for project.mpk:  100.00%
🏁 MPK sent successfully!
✔ MPK Deployed! ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100%
✔ MPK Deployment is successful for project.mpk.
```

- After changing sources/configs, follow the sequence: recreate `project.mpk` → reboot Modalix → redeploy.

- Then safely disconnect and reconnect with the commands below, and deploy again.

```
mpk device disconnect -t sima@192.168.1.20
mpk device connect -t sima@192.168.1.20
mpk deploy -f project.mpk
```

Step 18. Open a New Terminal (Outside Docker, Terminal 4)

- Open a new terminal (Terminal 4) in Ubuntu to check the processed YOLOv8m result video stream.

Step 19. Check the Result Stream (Outside Docker, Terminal 4)

- In the new fourth window, run the command below to receive the video stream from Modalix and display it.

```
GST_DEBUG=0 gst-launch-1.0 udpsrc port=5000 ! application/x-rtp,encoding-
name=H264,payload=96 ! rtph264depay ! video/x-h264,stream-format=byte-
stream,alignment=au !  avdec_h264  ! autovideoconvert ! fpsdisplaysink
sync=0
```

4. Performance Tuning Guide

4-1. Tuning via sima-model-to-pipeline Command

- Adjust the three key YOLO parameters—`detection_threshold`, `nms_iou_threshold`, `topk`—to optimize performance.
- When parameters change, rebuild the mpk and perform a power reset on Modalix.

4-2. Tuning by Editing boxdecode.json

- You can find `detection_threshold`, `nms_iou_threshold`, and `topk` in `yolov8m_pipeline_simaaisrc/plugins/genericboxdecode/cfg/boxdecode.json`. Modify those values, save the file, regenerate `project.mpk`, and redeploy to optimize the YOLOv8m pipeline.
- After editing and saving `boxdecode.json`,

```
{
...
"detection_threshold": 0.1,
"nms_iou_threshold": 0.3,
"topk": 100,
...
}
```

- Recreate `project.mpk`.

```
mpk create -s . -d . --clean --board-type modalix
```

4-3. Tuning via Python Code

- You can implement Graph Surgery and Compile in Python using `sima-model-to-pipeline` functions.
- Refer to the two Python files in the `yolo` folder.

```
exam_surgery.py: an example showing how to implement model surgery.
exam_compile.py: an example showing how to implement model quantization and
compile.
```

- After creating `yolov8m.onnx`, run `exam_surgery.py`.
- The following command creates `yolov8m_pipeline_mod.onnx` in the same folder.

```
python3 exam_surgery.py \
    --model-path ./yolov8m.onnx \
    --model-name yolov8 \
    --pipeline-name yolov8m_pipeline \
    --num-classes 80 \
    --input-width 1920 \
    --input-height 1080 \
    --labels-file "" \
    --topk 300 \
    --device-type modalix
```

- Use the --labels-file option to specify a label file such as ./coco_80_class_names.txt. This file must contain the 80 class names from the COCO dataset used to train YOLOv8m, with each line corresponding to a class index (0–79). If the option is not specified, numeric indices are displayed instead of class names. For the YOLOv8m model used in this document, label information is embedded in the model itself, so no external label file is required.

```
>>> from ultralytics import YOLO
>>>
>>> model = YOLO("yolov8m.pt")
>>> print(model.names)
{0: 'person', 1: 'bicycle', 2: 'car', 3: 'motorcycle', 4: 'airplane', 5:
'bus', 6: 'train', 7: 'truck', 8: 'boat', 9: 'traffic light', 10: 'fire
hydrant', 11: 'stop sign', 12: 'parking meter', 13: 'bench', 14: 'bird', 15:
'cat', 16: 'dog', 17: 'horse', 18: 'sheep', 19: 'cow', 20: 'elephant', 21:
'bear', 22: 'zebra', 23: 'giraffe', 24: 'backpack', 25: 'umbrella', 26:
'handbag', 27: 'tie', 28: 'suitcase', 29: 'frisbee', 30: 'skis', 31:
'snowboard', 32: 'sports ball', 33: 'kite', 34: 'baseball bat', 35:
'baseball glove', 36: 'skateboard', 37: 'surfboard', 38: 'tennis racket',
39: 'bottle', 40: 'wine glass', 41: 'cup', 42: 'fork', 43: 'knife', 44:
'spoon', 45: 'bowl', 46: 'banana', 47: 'apple', 48: 'sandwich', 49:
'orange', 50: 'broccoli', 51: 'carrot', 52: 'hot dog', 53: 'pizza', 54:
'donut', 55: 'cake', 56: 'chair', 57: 'couch', 58: 'potted plant', 59:
'bed', 60: 'dining table', 61: 'toilet', 62: 'tv', 63: 'laptop', 64:
'mouse', 65: 'remote', 66: 'keyboard', 67: 'cell phone', 68: 'microwave',
69: 'oven', 70: 'toaster', 71: 'sink', 72: 'refrigerator', 73: 'book', 74:
```

```
'clock', 75: 'vase', 76: 'scissors', 77: 'teddy bear', 78: 'hair drier', 79:
'toothbrush'}
```

- After generating `yolov8m_pipeline_mod.onnx`, run `exam_compile.py`.
- Unlike `sima-model-to-pipeline`, it is set up to use the newly created `yolov8m_pipeline_mod.onnx`.
- The following command creates `result/modalix/yolov8m_pipeline.tar.gz` in the same folder.

```
python3 exam_compile.py \
    --surgery-onnx ./yolov8m_pipeline_mod.onnx \
    --precision int8 \
    --calib-dir /home/docker/calibration_images \
    --calib-count 128 \
    --calib-ext jpg \
    --calib-type minmax \
    --batch-size 1 \
    --device-type modalix
```

4-3-1. Code Explanation

- `exam_surgery.py`

```
This script consists of three parts.

A. Environment setup and module import
Registers the SiMa.ai SDK (model_to_pipeline) path, and imports the YOLOv8-
specific surgeon_yolov8 module and the StepSurgery class.

B. build_args() function
Converts CLI arguments into the internal format required by the SDK.
Sets input ONNX path, number of classes, resolution, Top-K, etc.
Fixes the output filename to ./yolov8m_pipeline_mod.onnx.

C. main() function
Parses arguments and calls StepSurgery().run() to perform graph surgery.
On success, prints the output path; on failure, exits.

Output:
The modified ONNX file is always created in the current working folder
(./yolov8m_pipeline_mod.onnx),
and is used as the input for the next step (exam_compile.py).
```

- `exam_compile.py`

```
This script consists of three parts.
```

```
A. Environment setup and module import
Registers the SiMa.ai SDK (model_to_pipeline) path and imports the
CompileYoloGeneric class.
This class compiles the modified ONNX model to produce a .tar.gz runtime
artifact.

B. build_args() function
Converts CLI arguments into the internal structure required by the SDK.
Includes model info, precision, batch size, and calibration settings.
Fixes the output folder to ./result/modalix and the filename to
yolov8m_pipeline.tar.gz.

C. main() function
Parses arguments and runs CompileYoloGeneric().run() to perform compilation.
After completion, moves or saves the result file to the specified path with
the expected name,
and on failure, exits with an error message.

Output:
The final artifact is always generated at
./result/modalix/yolov8m_pipeline.tar.gz,
provided as a package executable on the Modalix device.
```

5. Appendix

Glossary

- Modalix: SiMa.ai's MLSoC-based Edge AI module (integrating MLA, APU, CVU)
- Palette SDK: A Docker-based development environment for model conversion and compilation for Modalix
- Graph Surgery: ONNX graph modification process; performs node merge/removal and structural optimization
- Quantization: Converts a floating-point model to integer to improve performance and save memory
- Pipeline (mpk): The execution unit on Modalix; the entire data flow that includes model, I/O, and post-processing
- RTSP: Real-Time Streaming Protocol, a network protocol for video streaming
- NMS (Non-Maximum Suppression): Algorithm that removes overlapping bounding boxes
- IoU (Intersection over Union): Metric indicating overlap between boxes, used in object detection evaluation
- Calibration Data: A sample image set used to determine integer scaling during quantization
- TopK: Keeps only the top-K detections by confidence in the post-processing step

- Ultralytics YOLOv8m (public base model, `yolov8m.pt`) is trained on the COCO dataset (Common Objects in Context) and has 80 classes. Indices are in order 0: person, 1: bicycle, 2: car, 3: motorcycle, ..., 79: toothbrush.

6. References

- SiMa.ai Developer Guide
- SiMa.ai Community
- Welcome to SiMa.ai on GitHub
- Hugging Face: SiMa.ai
- Ultralytics YOLOv8 Docs
- bluenviron/mediamtx GitHub