

# Getting Started with YOLO on Modalix

## Modalix에서 YOLOv8m 모델 시작하기

### 1. 문서 목적과 대상

이 가이드는 Ultralytics YOLOv8m 모델을 예시로 하여, SiMa.ai의 Modalix Edge AI 플랫폼에서 AI 추론 파이프라인을 직접 빌드하고 실행하는 고객용 단계별 실습 안내서(Hands-on Guide)입니다. 고객은 이 문서를 따라 하면서 SDK 환경 설정, 모델 변환, mpk 생성 및 Modalix 실행까지의 전체 과정을 한 번에 경험할 수 있습니다.

대상: Modalix 신규 사용자, 파트너 엔지니어, PoC 담당자 등

### 2. 준비물과 사전 요구사항

- Modalix SoM DevKit 또는 Engineering EVB
- Ubuntu 22.04 이상이 설치된 개발 PC
- Palette SDK 1.7 설치
- Modalix와 개발 PC 간 RJ45 유선 네트워크 연결 (SDK 사용 전, 시리얼 포트로 접속하여 Modalix IP를 수동으로 설정해야 함)
- ⚡ IP 주소 구성

개발 PC (Host): 192.168.1.10  
Modalix: 192.168.1.20

ffmpeg(호스트) → `rtsp://127.0.0.1:8554/mystream1`  
컨테이너에서 RTSP 접속 → 호스트 LAN IP(예: `192.168.1.10`)  
결과 스트림 → Modalix → 호스트 \*\*UDP:5000\*\*

- 이 IP 구성은 예제 명령어에서 사용되는 기본 설정이며, 실제 환경에 맞게 수정 가능합니다.
- ⚡ 참고: Docker, FFmpeg, GStreamer는 Palette SDK Docker 이미지에 이미 포함되어 있습니다. 별도의 설치는 필요하지 않습니다.

#### 2-1. 네트워크·포트 요약

- Network and Port

경로	프로토콜	포트	비고
ffmpeg → mediamtx	TCP	8554	-rtsp_transport tcp
컨테이너 → RTSP(호스트 IP)	TCP	8554	127.0.0.1 금지, LAN IP 사용

경로	프로토콜	포트	비고
Modalix → 호스트	UDP	5000	<code>gst udpsrc port=5000</code>

- 사전 체크리스트

모든 명령은 Docker 컨테이너 내부에서 실행합니다. (단, Step 11-14, 18-19는 Docker 외부)  
 RTSP 주소: 컨테이너 → 호스트 IP (예: 192.168.1.10) / 호스트 → 127.0.0.1  
 모델/해상도: imgsz=640, 입력 스트림 1920×1080

### 3. YOLOv8m 실습 가이드: 모델 변환부터 Modalix 실행까지

Ultralytics YOLOv8m 모델을 사용하여 Modalix용 mpk 파일을 생성하고 실제 장비에서 실행하는 전체 과정을 설명합니다.

#### Step 1. 터미널 열기 및 SDK Docker 실행

- Ubuntu에서 새 터미널을 엽니다. (Terminal 1)
- start.py를 실행하여 Palette SDK Docker 환경을 시작합니다.
- Palette SDK 설치 시 start.py가 함께 설치되며, Docker 실행의 권장 방법입니다.
- 예를 들어 만일 아래 폴더에 설치한 경우, 해당 폴더로 이동하여 실행합니다.

```
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\Dockers\Dockers 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$
```

- Palette SDK는 Docker 기반으로 제공되며, 위 명령을 실행하면 Docker가 시작됩니다.

#### Step 1-1. 호스트-컨테이너 공유 디렉터리 설정

- SDK를 사용할 때, 호스트와 Docker 컨테이너 간 파일을 손쉽게 공유할 수 있도록 로컬 워크스페이스 디렉터리를 컨테이너 내부 경로에 바인드 마운트해야 합니다.

예를 들어:  
 호스트: /home/howard-lee/workspace\_1.7.0  
 컨테이너: /home/docker/sima-cli

- 이 두 경로는 동일한 물리적 디렉터리를 가리키며, 한쪽의 변경이 즉시 다른 쪽에 반영됩니다. 이 구조를 통해 모델, 스크립트, 설정 파일을 별도의 복사 과정 없이 주고받을 수 있습니다.
- 참고: 폴더 이름과 경로는 SDK 설정 시 사용자가 자유롭게 지정할 수 있습니다.

## Step 2. sima-model-to-pipeline 설치 (Docker 내부, Terminal 1)

- 인터넷 접속이 가능한 경우는 Github를 이용하여 설치하고 실행이 가능합니다. 그렇지 않은 경우는 설치가 가능한 ZIP 파일을 이용하여 설치가 가능합니다.

### Step 2-1. Github를 이용한 설치와 실행 (인터넷 접속 가능)

- Docker로 이동합니다.

```
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
```

- 설치 명령을 실행합니다. 설치 성공이 나오면 이제 사용 가능합니다.

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

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

- yaml 파일을 편집하고 다음처럼 실행합니다. yaml 파일 예들은 samples 디렉터리에 있습니다.

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

- 완료 시에는 다음과 같은 테이블이 표시되며 모두 PASS인지 확인합니다.

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...

- 문제가 발생한 경우는 같은 위치에 logs 폴더에서 실패한 Step의 로그를 분석합니다.

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

- 문제 없이 모두 PASS인 경우는 같은 위치의 yolov8m\_simaaaisrc 디렉터리에 있는 project.mpk 파일을 확인합니다.

```
cd ./yolov8m_simaaaisrc
```

- 다음과 같은 파일들을 볼 수 있습니다.

```
.project
build
core
dependencies
plugins
resources
.hash
```

```
application.json
application_pcie.json
application_rtsp.json
project.mpk
yolov8m_simaaaisrc_plugin_version_index.json
```

- 생성된 project.mpk 파일을 바로 Modalix 보드에서 실행해 보기 위해서는 이제 바로 Step 11로 이동하시면 됩니다.
- 또한, 만일 yaml 파일을 사용하지 않고 각 단계씩 직접 실험해 보기 원하신다면 Step 4로 이동하시면 됩니다.

### Step 2-2. mpktool 패키지 디렉터리 이동 (인터넷 접속 불가 환경)

- Docker로 이동합니다.

```
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
```

- Docker 외부에서 내부 폴더로 mpktool\_package\_sdk1.7.zip 을 복사합니다

```
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
```

- Docker에서 mpktool\_package\_sdk1.7.zip 압축을 풉니다.

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

- 압축이 풀린 폴더로 이동합니다.

```
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. ZIP 파일을 이용한 sima-model-to-pipeline 설치 (Docker 내부, Terminal 1)

- 이제 sima-model-to-pipeline tool을 설치합니다.

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

- 설치 중 경고 메시지는 발생할 수 있으나 설치가 완료되면 정상입니다. 설치가 중단될 경우 로그를 첨부해 지원 채널로 문의하세요.

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

- 참고로 나중에 삭제가 필요하면 아래 2개 명령으로 합니다.

```
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
```

- 삭제 후에는 완전히 삭제되었나 확인합니다.

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

#### Step 4. Ultralytics 설치 (Docker 내부, Terminal 1)

- PyTorch(.pt) 모델을 ONNX 모델로 export하기 위하여 ultralytics 패키지를 설치합니다.

```
pip3 install ultralytics
```

설치가 되었나 yolo 라고 입력해 봅니다.

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

#### Step 5. YOLO 예제 디렉터리 생성 (Docker 내부, Terminal 1)

- 본 문서와 함께 2개 python 파일과 1개 mp4 파일을 받으셨습니다.

```
exam_surgery.py: Python으로 구현한 surgery 예
exam_compile.py: Python으로 구현한 Quantization/Compile 예
example.mp4: 예제 비디오 파일 1920x1080@30fps
```

- 실험을 진행할 디렉터리를 만듭니다.

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

- 2개 Python 파일(exam\_surgery.py, exam\_compile.py)만 yolo 폴더에 복사합니다.
- 복사 방법은 sima\_model\_to\_pipeline-sdk-1.7.tar.gz 파일을 docker로 복사한 것과 같습니다.
- example.mp4는 호스트 PC에서 ffmpeg로 RTSP 스트리밍할 때 사용하므로 Docker 컨테이너 밖에 둡니다.

#### Step 6. YOLOv8m 모델 다운로드 (Docker 내부, Terminal 1)

- ultralytics에서 예제인 YOLOv8m을 다운로드 받습니다.
- Google에서 'yolov8 ultralytics download'를 검색해서 다운로드해도 됩니다.

```
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. PyTorch(.pt) 모델을 ONNX 모델로 export (Docker 내부, Terminal 1)

- 다운로드 받은 모델은 yolov8m.pt 입니다.
- 이를 yolo 명령으로 ONNX 형식으로 export합니다.
- 아래 명령으로 같은 폴더에 yolov8m.onnx 생성됩니다.

```
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 (필요 시 모델 구조 수정) (Docker 내부, Terminal 1)

- Graph Surgery는 ONNX 그래프의 노드 구조를 직접 수정하거나, 불필요한 연산 노드를 제거·병합·변환하는 과정을 의미하며, AI 컴파일러 및 런타임 분야에서는 매우 널리 쓰이는 용어입니다.
- step surgery 옵션으로 graph surgery를 실행합니다.
- 사용된 입력영상 크기는 1920×1080 입니다.
- 아래 명령어로 같은 폴더에 yolov8m\_pipeline\_mod.onnx 생성됩니다.

```

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
```

- YOLOv8m 실행 후에는 host (192.168.1.10)의 port 5000으로 전송하도록 설정하고 있습니다.
- RTSP (Real Time Streaming Protocol)는 실시간 영상·음성 스트리밍을 제어하기 위한 네트워크 프로토콜입니다.
- 💡 컨테이너 내부에서 호스트 RTSP 서버에 접근하므로, 컨테이너 측 URL은 127.0.0.1이 아닌 호스트의 LAN IP(예: 192.168.1.10)를 사용해야 합니다. 이 설정은 Step 14에서 ffmpeg로 스트림을 송출할 때 사용하는 RTSP IP 주소(127.0.0.1)와 대응됩니다.
- 위 명령에서 device-type은 davinci 또는 modalix로 지정해 주셔야 합니다.

옵션	의미	적용 대상
--device-type davinci	Gen1	DaVinci EVB
--device-type modalix	Gen2	Modalix SoM/EVB

- detection-threshold, nms-iou-threshold, topk는 YOLO 성능 관련한 최적화 파라미터입니다

- 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. exam\_surgery.py 실행 시 출력

- sima-model-to-pipeline를 사용하지 않고 exam\_surgery.py를 실행했을 경우 출력은 다음과 같습니다.

	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
```

- yolov8m\_pipeline\_mod.onnx 생성을 확인합니다

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

## Step 9. Compile (Quantization 포함) (Docker 내부, Terminal 1)

- Graph Surgery 이후에는 Quantization과 Compile을 한번에 실시합니다.
- Quantization에 필요한 calibration image의 path를 입력합니다.
- Palette SDK 설치 시 calibration image가 Docker에 복사됩니다.
- Quantization에 필요한 precision을 정의하며 INT8, INT16, BF16이 존재합니다. 성능과 정확도를 둘 다 고려해 대부분의 경우 INT8 사용합니다.
- 이 문서에서도 INT8을 사용합니다. sima-model-to-pipeline은 INT8을 기본으로 지원합니다.
- 아래 명령어로 같은 폴더에 result/modalix/yolov8m\_pipeline.tar.gz 파일이 생성됩니다.
- 💡 Graph Surgery 단계에서 **yolov8m\_pipeline\_mod.onnx** 파일이 생성되더라도, sima-model-to-pipeline의 Step 9(compile)-Step 10(pipelinecreate) 입력 모델은 **yolov8m.onnx**입니다.

```
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. exam\_compile.py 실행 시 출력

- sima-model-to-pipeline를 사용하지 않고 exam\_compile.py를 실행했을 경우 출력

```

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
... (타일/메모리 배치 상세 로그 다수) ...
[OK] Compilation finished.
Requested artifact path: /home/docker/sima-
cli/mpktool_package/yolo/result/yolov8m_pipeline.tar.gz

```

- yolov8m\_pipeline.tar.gz 생성을 확인합니다

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

## Step 10. Pipeline 생성 (Docker 내부, Terminal 1)

- 이제 Modalix에서 실행할 pipeline을 만드는 순서입니다.
- edge device에서 pipeline은 모델 단위가 아니라, 모델을 포함한 전체 처리 흐름 단위를 지칭합니다.
- 즉, 입력 → 전처리 → 추론 → 후처리 → 출력 과정을 한 줄의 파이프로 묶은 개념입니다.
- Modalix에서 실행하기 위한 project.mpk 파일 생성에 필요한 입력 파일들을 만드는 단계입니다.
- 아래 명령어를 실행하면 같은 위치에 yolov8m\_pipeline\_simaaisrc 폴더가 생성 됩니다.

```

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

```

<input checked="" type="checkbox"/>	Completed : setup	: 0.10 sec
<input checked="" type="checkbox"/>	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 ..
-rwrxr-xr-x 1 howard-lee howard-lee     6700 Oct  4 13:16 exam_compile.py
-rwrxr-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\_simaaisrc에는 Modalix에서 실행 가능한 pipeline을 생성하기 위한 입력 파일들과 디렉터리들이 있습니다.
- JSON 파일 내용을 포함하여 상당히 복잡한 구조라 Python code 구현은 생략합니다.

```
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. 새 터미널 열기 (Docker 외부, Terminal 2)

- 이제 Docker 컨테이너 외부에서 RTSP 서버와 비디오 송출을 실행할 차례입니다.
- Ubuntu에서 새로운 터미널(Terminal 2)을 엽니다.

## Step 12. RTSP 서버 실행 (Docker 외부, Terminal 2)

- 다음의 명령으로 RTSP 서버를 실행시킵니다.

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

## Step 13. 새 터미널 열기 (Docker 외부, Terminal 3)

- 비디오 예제 스트림 송출을 위해 Ubuntu에서 또 다른 터미널(Terminal 3)을 엽니다.

## Step 14. 예제 비디오 스트림 송출 (Docker 외부, Terminal 3)

- 이제 예제 비디오 파일을 RTSP 서버로 스트림 송출이 필요합니다.
- 이때, example.mp4 파일이 해당 위치에 존재해야 합니다. 1920×1080 크기입니다.

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

- ☺ 참고: ffmpeg 명령에서는 RTSP 주소를 127.0.0.1로 사용하지만, Modalix Docker 컨테이너 내부에서는 192.168.1.10처럼 호스트의 IP로 접근해야 합니다.

## Step 15. Docker 내부로 복귀 (Docker 내부, Terminal 1)

- project.mpk 파일을 생성할 준비가 되었습니다. 이 파일이 Modalix 상에서 실행됩니다.
- SDK Docker 터미널(Terminal 1)로 다시 되돌아옵니다.

## Step 16. mpk 빌드 (Docker 내부, Terminal 1)

- 윗 단계에서 이미 생성된 yolov8m\_pipeline\_simaisrc 폴더에 필요한 파일들과 폴더들이 이미 생성되어 있습니다.
- 해당 폴더로 이동하여 아래와 같이 mpk create 명령을 실행시킵니다.
- 문제 없이 완료되면 같은 위치에 project.mpk 파일이 생성됩니다.

```
cd yolov8m_pipeline_simaisrc  
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. Modalix에 배포 및 실행 (Docker 내부, Terminal 1)

- 이제 Modalix에 생성된 project.mpk 파일을 deploy해야 합니다.
- Modalix 장비의 전원을 켭니다.
- 다음의 명령으로 Modalix의 부팅이 완료 되었나 확인합니다.

```
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
```

- ping 응답이 되면, 장비를 연결합니다. 이때 비밀번호가 필요할 수도 있는데 edgeai 입니다.

```
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.
```

- 연결이 성공했다는 메시지가 나오면 아래 명령으로 project.mpk를 deploy 합니다.

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

```
i Checking if App yolov8m_pipeline_simaaaisrc Plugin Version Index File
/home/docker/sima-
cli/mpktool_package/yolo/yolov8m_pipeline_simaaaisrc/yolov8m_pipeline_simaa-
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 'simaaaisrc': {'name': 'simaaaisrc', 'version': '1.7'} ->
Matched
    Match found for 'simaaiooverlay2': {'name': 'simaaiooverlay2', '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_simaaaisrc Vs Device Plugin Version Info
Comparison
✓ App yolov8m_pipeline_simaaaisrc 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.
```

- 소스/설정 변경 후에는 project.mpk 재생성 → Modalix 재부팅 → 재배포 순서로 진행하세요.
- 그 이후 아래 명령어를 통하여 안전하게 disconnect 하고 다시 connect 합니다. 그리고 deploy까지 합니다.

```
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. 새 터미널 열기 (Docker 외부, Terminal 4)

- YOLOv8m 처리된 결과 영상 스트림을 확인하기 위해 Ubuntu에서 새로운 터미널(Terminal 4)을 엽니다.

## Step 19. 결과 스트림 확인 (Docker 외부, Terminal 4)

- 새로운 4번째 창에서 아래 명령어를 실행하여 Modalix로부터 오는 영상 스트림을 받아서 화면에 보여줍니다.

```
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 ! fakesink sync=0
```

## 4. 성능 최적화 가이드

### 4-1. sima-model-to-pipeline 명령어를 통한 최적화

- detection\_threshold, nms\_iou\_threshold, topk 3개 주요 YOLO 파라미터를 변경해 보며 성능 최적화를 해봅니다.
- 파라미터 변경 시, mpk 빌드를 다시 해야 합니다. 그리고 Modalix도 Power Reset해야 합니다.

### 4-2. boxdecode.json 편집을 통한 최적화

- yolov8m\_pipeline\_simaaisrc/plugins/genericboxdecode/cfg/boxdecode.json 파일에서 detection\_threshold, nms\_iou\_threshold, topk를 찾을 수 있습니다. 해당 값을 수정한 후 저장하고 project.mpk 파일을 다시 생성하여 deploy하는 방법으로 YOLOv8m pipeline 최적화가 가능합니다.
- boxdecode.json 수정 후 저장하고,

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

- project.mpk 생성을 다시 합니다.

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

### 4-3. Python code를 통한 최적화

- sima-model-to-pipeline 함수들을 이용하여 Graph Surgery와 Compile등을 python code로 구현해 볼 수도 있습니다.
- yolo 폴더에 있던 2개 python 파일을 참조합니다.

exam\_surgery.py: model surgery를 어떻게 구현하는지에 대한 예제입니다.  
 exam\_compile.py: model Quantization 및 Compile을 어떻게 구현하는지에 대한 예제입니다.

- yolov8m.onnx 생성 이후에 exam\_surgery.py 실행합니다
- 아래 명령어로 같은 폴더에 yolov8m\_pipeline\_mod.onnx 생성됩니다.

```

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
  
```

- labels-file에는 ./coco\_80\_class\_names.txt 등의 label 파일을 제공하면 됩니다. YOLOv8m 학습에 사용된 COCO 데이터셋의 80개 클래스 이름은 coco\_80\_class\_names.txt 파일에 포함되어 있습니다. 각 행은 클래스 인덱스(0~79)에 대응하는 이름입니다. --labels-file 옵션을 제공하지 않으면 클래스 이름 대신 숫자 인덱스로 표시됩니다. 하지만 sima-model-to-pipeline이 적합한 label을 자동으로 제공함으로 여기서는 입력할 필요가 없습니다.
- --labels-file에는 ./coco\_80\_class\_names.txt 등의 라벨 파일을 제공하면 됩니다. 즉, YOLOv8m 학습에 사용된 COCO 데이터셋의 80개 클래스 이름이 coco\_80\_class\_names.txt 파일에 포함되어 있어야 합니다. 각 행은 클래스 인덱스(0~79)에 대응하는 이름을 가집니다. --labels-file 옵션을 지정하지 않으면 클래스 이름 대신 숫자 인덱스로 표시됩니다. 단, 본 문서에서 사용된 YOLOv8m 모델은 모델 내부에 이미 라벨 정보가 포함되어 있으므로 별도의 입력이 필요하지 않습니다.

```

>>> 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'}
```

- yolov8m\_pipeline\_mod.onnx 생성 이후에 exam\_compile.py 실행합니다.
- sima-model-to-pipeline 와는 다르게 새로 생성된 **yolov8m\_pipeline\_mod.onnx** 파일이 사용되도록 했습니다.
- 아래 명령어로 같은 폴더에 result/modalix/yolov8m\_pipeline.tar.gz 파일이 생성됩니다.

```
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. 코드 설명

- exam\_surgery.py

이 스크립트는 세 부분으로 구성됩니다.

##### A. 환경 설정 및 모듈 불러오기

SiMa.ai SDK(model\_to\_pipeline) 경로를 등록하고, YOLOv8 전용 `surgeon_yolov8` 모듈과 `StepSurgery` 클래스를 불러옵니다.

##### B. build\_args() 함수

명령행 인자를 SDK가 요구하는 내부 형식으로 변환합니다.

입력 ONNX 경로, 클래스 수, 해상도, Top-K 등을 설정하고

출력 파일명을 항상 `./yolov8m_pipeline_mod.onnx`으로 고정합니다.

##### C. main() 함수

인자를 파싱한 뒤 `StepSurgery().run()`을 호출하여 그래프 서저리를 수행합니다.

성공 시 출력 경로를 표시하고, 실패 시 종료합니다.

출력:

수정된 ONNX 파일은 항상 현재 작업 폴더(`./yolov8m_pipeline_mod.onnx`)에 생성되며, 다음 단계(`exam_compile.py`)의 입력으로 사용됩니다.

- `exam_compile.py`

이 스크립트는 세 부분으로 구성됩니다.

A. 환경 설정 및 모듈 불러오기

`SiMa.ai` SDK(`model_to_pipeline`) 경로를 등록하고, `CompileYoloGeneric` 클래스를 불러옵니다.

이 클래스는 수정된 ONNX 모델을 컴파일하여 `.tar.gz` 형태의 실행 아티팩트를 생성합니다.

B. `build_args()` 함수

명령행 인자를 SDK에서 요구하는 내부 구조로 변환합니다.

모델 정보, 정밀도(precision), 배치 크기, 보정(calibration) 설정 등을 포함하며 출력 폴더를 항상 `./result/modalix.`로, 파일명을 `yolov8m_pipeline.tar.gz`로 고정합니다.

C. `main()` 함수

인자를 파싱한 후 `CompileYoloGeneric().run()`을 실행하여 컴파일을 수행합니다.

완료 후 결과 파일을 지정된 경로로 이동 또는 이름을 맞춰 저장하며, 실패 시 오류 메시지와 함께 종료합니다.

출력:

최종 결과물은 항상 `./result/modalix/yolov8m_pipeline.tar.gz`로 생성되며, Modalix 장치에서 실행 가능한 패키지 형태로 제공됩니다.

## 5. 부록

### 용어집

- Modalix: SiMa.ai의 MLSoC 기반 Edge AI 모듈 (MLA, APU, CVU 통합)
- Palette SDK: Modalix용 모델 변환 및 컴파일 개발 환경 (Docker 기반)
- Graph Surgery: ONNX 그래프 수정 과정. 노드 병합, 제거, 구조 최적화 수행
- Quantization: 부동소수점 모델을 정수형으로 변환하여 성능 향상 및 메모리 절약
- Pipeline (mpk): Modalix에서 실행되는 단위. 모델·입출력·후처리 포함한 전체 데이터 흐름
- RTSP: Real-Time Streaming Protocol, 영상 스트리밍용 네트워크 프로토콜
- NMS (Non-Maximum Suppression): 겹치는 bounding box 제거 알고리즘
- IoU (Intersection over Union): 객체 탐지 성능 측정 지표로, 박스 간 겹침 정도를 의미
- Calibration Data: Quantization 시 정수 스케일링을 결정하기 위한 샘플 이미지 집합
- TopK: 후처리 단계에서 confidence가 높은 상위 K개의 detection만 유지

- Ultralytics YOLOv8m (기본 공개 모델, yolov8m.pt)은 COCO dataset(Common Objects in Context)으로 학습된 모델이며, 80개의 클래스를 가지고 있습니다. 0: person, 1: bicycle, 2: car, 3: motorcycle, ..., 79: toothbrush 순서입니다.

## 6. 참고자료

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