HAILO

Empowering Intelligence

Hailo AI SW Suite User Guide

Release 2024-10

30 September 2024

Table of Contents

Disclaimer and Proprietary Information Notice

Copyright

This version of the document supersedes all previous versions.

General Notice

Hailo, to the fullest extent permitted by law, provides this document "as-is" and disclaims all warranties, either express or implied, statutory or otherwise, including but not limited to the implied warranties of merchantability, non-infringement of third parties' rights, and fitness for particular purpose.

Although Hailo used reasonable efforts to ensure the accuracy of the content of this document, it is possible that this document may contain technical inaccuracies or other errors. Hailo assumes no liability for any error in this document, and for damages, whether direct, indirect, incidental, consequential or otherwise, that may result from such error, including, but not limited to loss of data or profits.

The content in this document is subject to change without prior notice and Hailo reserves the right to make changes to content of this document without providing a notification to its users.

1. Getting Started Guide

Hailo SW products are set of frameworks and tools that enable you to compile, run and evaluate neural networks on Hailo devices:

1. Dataflow Compiler (Model conversion and compilation to Hailo binary format)
2. HailoRT (Runtime environment and driver for running networks and interacting with Hailo devices)
3. Model Zoo (Pre-trained models to run and evaluate on Hailo devices)
4. TAPPAS (Deployment framework, examples and multi-network pipelines)

Although you can install each product separately, Hailo releases a quarterly software suite in which all the product versions are aligned. Therefore, using Hailo AI SW Suites ensures the best compatibility.

For information about the latest suite version (recommended), see *Latest SW Suite Features*. For installation guide, see *Suite Installation*. For compatibility between Hailo's products (and suites) versions see *Version Compatibility Table*.

1.1. Suite Components

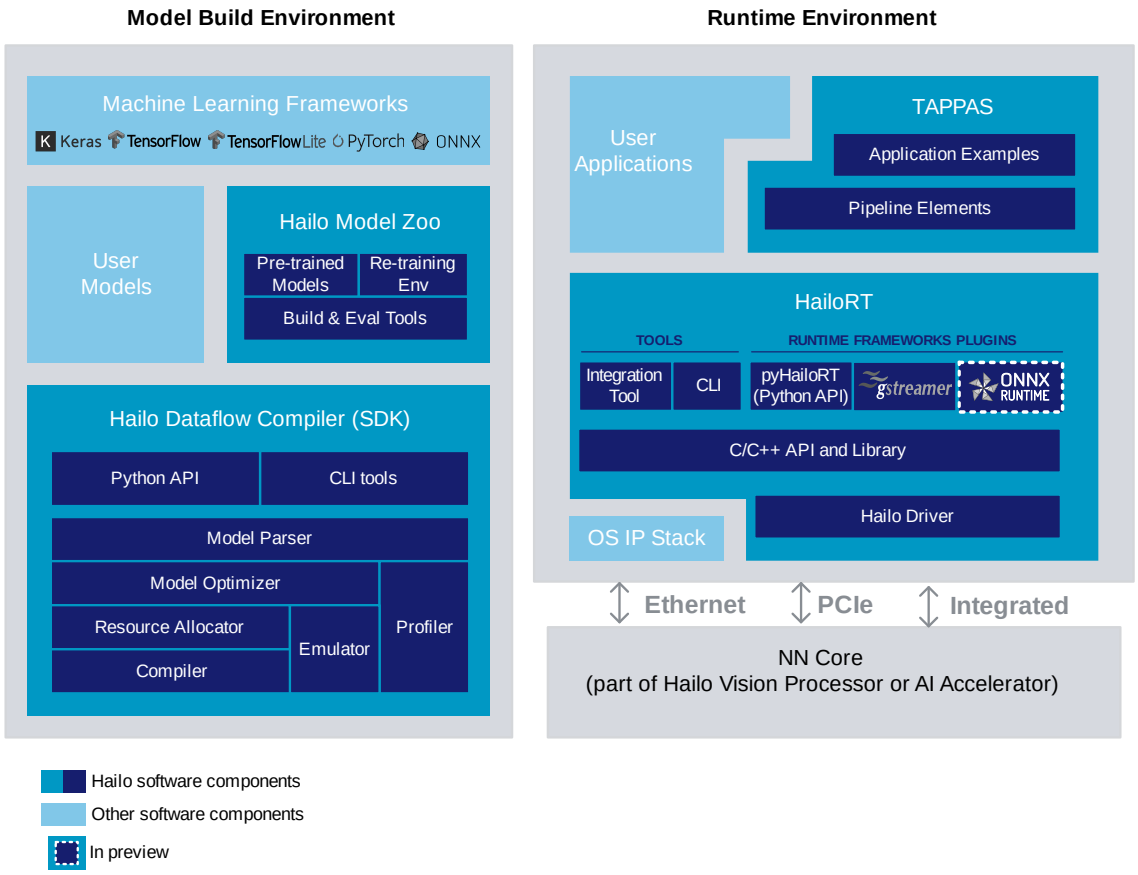

Figure 1. Detailed block diagram of Hailo software packages

Hailo SW components are used in this manner:

- **On the Model build environment:**
  - Hailo Dataflow Compiler is used to compile a trained model to run on Hailo devices.

  - Hailo Model Zoo contains a large database of pre-trained models that are validated to work with best performance on Hailo devices.

    It also contains a retraining environment.

- **On the Runtime environment:**
  - HailoRT is used to load the compiled model to Hailo device and interact with it (using the PCIe driver).
  - TAPPAS includes complete examples and demos of using HailoRT to create full pipelines on top of Hailo devices.

1.1.1. Dataflow Compiler

The Dataflow Compiler API is used for compiling models to Hailo binaries.

The input of the Dataflow Compiler is a trained Deep Learning model.

The output is a binary file which is loaded to the Hailo device.

1.1.2. HailoRT

The HailoRT API is used for deploying the built model on the target device. This library is used by the runtime applications.

It implements a userspace C/C++ API that is called from the user's applications. It allows both to control the Hailo device and to send and receive data from it. It supports both the PCIe interface and the Ethernet interface.

The HailoRT Python package wraps the C/C++ API and exposes a Python interface that allows to load models to the device and send and receive data from it.

It also includes a PCIe driver is required when working via the PCIe interface. It links the HailoRT library and the device. It also loads the device's firmware when working through this interface.

Finally, Hailo's Yocto layer allows the user to integrate Hailo's software into an existing Yocto environment. It includes recipes for the HailoRT library, Python package and the PCIe driver.

1.1.3. Hailo Model Zoo

Hailo Model Zoo provides pre-trained models for high-performance deep learning applications.

Using the Hailo Model Zoo you can measure the full precision accuracy of each model, the optimized accuracy using the Hailo Emulator and measure the accuracy on the Hailo-8 device.

Finally, you will be able to generate the Hailo Executable Format (HEF) binary file to speed-up development and generate high quality applications accelerated with Hailo-8.

The models are optimized for high accuracy on public datasets and can be used to benchmark the Hailo model optimization scheme.

1.1.4. TAPPAS

TAPPAS is Hailo's set of full application examples, implementing pipeline elements and pre-trained AI tasks.

Demonstrating Hailo's system integration scenario of specific use cases on predefined systems (software and Hardware platforms). It can be used for evaluations, reference code and demos:

- Accelerating time to market by reducing development time and deployment effort
- Simplifying integration with Hailo's runtime SW stack
- Providing a starting point for customers to fine-tune their applications

1.2. Where to begin

Consider that the Hailo AI Suite is **intended for the model build platform**, which is a host that meets the *system requirements*. The host could also serve as the runtime platform, but not necessarily.

1. In case an Hailo accelerator is used, install it on the PC. If you are using the M.2 module, refer to the Hailo-8 M.2 hardware installation guide.

   Note: In case the runtime platform is different from the build platform, for example when using the Hailo-15H VPU (or an Hailo accelerator that is installed on a different platform), the models that are compiled using the Model Zoo or the Dataflow Compiler should be transferred to the device, and run using the on-board HailoRT package.

2. Install the latest suite: *Suite Installation*.
3. If an Hailo accelerator is connected, Hailo recommends to begin with TAPPAS usage guide to see various real-time demos in action using your PC's camera.

   Note: TAPPAS also comes pre-installed in the Hailo-15 Vision Processor Software Package.

   Note: Behind the scenes, TAPPAS uses HailoRT to interact with the device. It also uses the GStreamer framework.

4. It is then recommended to try our **Pre-Trained Models**, with the Hailo Model Zoo. With an easy-to-use CLI interface, it allows you to convert, retrain and compile models to your Hailo device. If an accelerator device is connected, it allows you to run and evaluate the performance and accuracy on the Hailo hardware.

   Note: Behind the scenes, the Hailo Model Zoo uses the Dataflow Compiler for compilation, and HailoRT Python API to interact with the device.

5. If a Hailo Model Zoo model is a good fit for your application, you can use your own dataset for retraining using Hailo Retraining Docker Containers.
6. For your **Custom Model**, use the Dataflow Compiler for optimization and conversion to Hailo binary format. Refer to the Dataflow Compiler tutorials section.

   Note: Use the Dataflow Compiler CLI tools or Python APIs for your custom models, not the Model Zoo CLI.

7. For building realtime applications, read the HailoRT Tutorials section and API Reference.

   Note: You can also use the TAPPAS examples code as a reference (for GStreamer framework integration or C/CPP API).

2. 2024-10 Hailo AI SW Suite - New Features

2.1. Dataflow Compiler

Parser

- Added the *input-format* flag to the parser CLI API (preview)
- Einsum operator support in ONNX was expanded with another equation, `bmchw,bnmc -> bmhwn`, which represents a group convolution.

Model Optimization

- Added format conversion support for YCbCr to RGB conversions
- Added a model script command to allow splitting of a fused activation from its layer, making it a standalone activation

Tools

- The redesigned **DFC Studio** now supports viewing multiple models within the same project
- Added a new feature in the DFC Studio - mapping corresponding layers between the Original and Hailo's graph in the Parsing stage

2.2. Hailo Model Zoo

- Using `jit_compile` reduces dramatically the emulation inference time of the Hailo Model Zoo models
- New tasks:
  - BEV: Multi-View 3D Object Detection
    - Added support for NuScenes dataset
    - Added PETRv2
- Added New Models:
  - CAS-ViT - S, M, T - Convolutional-Attention based classification model
  - YOLOv10 - base, x-large - Latest YOLO detectors
  - CLIP Text Encoders - ResNet50x4, ViT-Large
- Added a new retraining Docker container for PETR - Multi-View 3D Object Detection
- Added a new flag for `hailomz` - `--ap-per-class` for measuring average-precision per-class (relevant for object detection and instance segmentation tasks)

2.3. HailoRT

HailoNet

- Added HailoNet Windows support

Multi-Process Service

- Performance improvements when using InferModel API

HailoRT Profiler

- HailoRT Profiler is now released
- Added environment variables to control when will the HailoRT Profiler generate the report

LibHailoRT

- Added performance optimizations for Hailo-15 use cases
- Added the option to use exceptions

InferModel Examples

- Changed the Async API inference examples to utilize exceptions
- Updated the advanced Async API inference example to use pre-mapping API

PyHailoRT

- Updated package dependency to support more `numpy` versions

Note: Starting this version, Hailo-8 Ethernet-based platforms support is deprecated, so communication between the host and Hailo-8 is supported only over PCIe. HailoRT 4.18.0 was the last version to support Hailo-8's Ethernet interface.

2.4. TAPPAS

- TAPPAS is now released separately for Hailo-8, for Hailo-15 please refer to https://github.com/hailo-ai/tappas/tree/master-vpu
- Various bug fixes and stability Improvements

2.5. Previous Suite Versions

2.5.1. 2024-07.1 Hailo AI SW suite

TAPPAS

Hailo-8

- Updated infrastructure to better support the Hailo Raspberry Pi 5 Examples
- Added an option to control YOLO (Detection) Hailort post-process parameters via a JSON configuration
- Semantic segmentation post-process now extracts the argmax tensor using Regular Expressions

2.5.2. 2024-07 Hailo AI SW Suite

Dataflow Compiler

General

- Hailo Dataflow Compiler now supports Hailo-10H

Compiler

- Added an additional value to the compiler flag `performance_param` - for allocating models, which returns the first feasible partition it finds, allowing to reduce compilation time at the expense of the solution's quality

Model Optimization

- Added a new flag to optimization CLI, `--compilation-only-har`, allowing to save a reduced size har, containing only compilation related data

- Deprecate Auto MO model script

Tools

- Added preview version of “Dataflow Compile Studio”, a new graphical tool for Hailo’s toolchain
- The current release includes only the parsing step
- Added a new CLI tool `dfc-studio` which launches the Studio

Hailo Model Zoo

- Updated to use Dataflow Compiler v3.28.0
- Updated to use HailoRT 4.18.0
- Target hardware now supports Hailo-10H device
- New Models:
  - Original ViT models - tiny, small, base - Transformer based classification models
  - DeiT models - tiny, small, base - Transformer based classification models
  - DETR (resnet50) - Transformer based object detection model
  - fastvit_sa12 - Fast transformer based classification model
  - levit256 - Transformer based classification model
  - YOLOv10 - nano, small - Latest YOLO detectors
  - RepGhostNet1.0x, RepGhostNet2.0x - Hardware-Efficient classification models
- New postprocessing support on NN Core:
  - yolov6 tag 0.2.1
- Added support for person attribute visualization

HailoRT

Hailo-10H

- This version supports Hailo-10H (preview)

Async API

- Added support for Python Async API for VDevice (preview)

HailoRT post-processing support

- Added Yolov8 bbox only support

HailoRT Profiler (preview)

- Added a new HailoRT profiler visualization tool, which creates an html report for the HailoRT Model Scheduler behavior

General

- Added PyHailoRT Python 3.11 support

TAPPAS

Hailo-15

- Updated all Hailo-15 example applications to use the latest API
- Added a new AI example application - which is a C++ based example application that demonstrates the use of the Hailo-15 API
- This release is aligned with the Hailo-15 Vision Processor Software Package 2024-07

Hailo-8

- Added a new example application which demonstrates x86 hardware-accelerated multi-stream detection
- Various bug fixes and stability improvements for Raspberry Pi 5

General

- Fixed various stability issues across apps and platforms

2.5.3. 2024-04 Hailo AI SW Suite

Dataflow Compiler

Parser

- Added information to logger after model translation is completed:
  - Mapping input layers to original input node names, to ease creation of feed dict for native inference
  - Listing output layers by their original names, in the same order specified by the user (or as the original model, if not specified)

Post Processing

- **Added support for NanoDet meta-arch based on** YOLOv8 post-processing
- **Added support with post-processing of bbox decoding only in YOLOv5 by** using *bbox_decoding_only=True*
- **yolov5_seg NMS for instance segmentation task is supported in all stages of emulation and** compilation with *engine=cpu* (preview)

Emulator

- Added emulation support for NV21 and i420 input conversions.

Compiler

- Achieved up to 10% enhancement in FPS in Transformer models running on H-8 hardware
- Achieved up to 15% enhancement in FPS in Transformer models running on H-15 hardware
- Extending Performance Mode to multi-context networks, for pushing resource utilization limits to achieve higher FPS
- Achieved an average of 50% improvement in multi-context networks ran with Performance mode in all hardware architectures

Hailo Model Zoo

New Models

- FastSAM-s - Zero-shot Instance Segmentation

CLI changes

- Introduced new flags for `hailomz` CLI:
  - –start-node-names and –end-node-names for customizing parsing behavior
  - –classes for adjusting the number of classes in post-processing configuration
  - These flags simplify the process of compiling models generated from Model Zoo's retraining Docker

HailoRT

Async API

- C++ support for VDevice is now released
- Added support when working with multi-process service
- Updated HailoNet to use Async API by default (no API changes needed)
- Updated and added new tutorials
- Added support for passing a DMA buffer (preview)

HailoRT post-processing support

- Added Yolov5 bbox only support

CLI

- Renamed run modes in `hailortcli run2`:
  - `full` mode is replaced by `full_sync` and will keep being the default run mode
  - `raw` is replaced by `raw_sync`

> **Note:** Default run mode will be changed to `full_async` in a future version

Pre-Mapping Buffers for Async API

- Added a new API for pre-mapping buffers, used with async API (which improves performance when the same buffer is used multiple times)

TAPPAS

- TAPPAS was updated with a revised list of supported platform and apps
- Added `yolov8` (as default) to Detection application examples
- Fixed various stability issues across apps and platforms

Note: Ubuntu 20.04 and Python 3.8 will no longer be supported in future versions.

2.5.4. 2024-01 Hailo AI SW Suite

Dataflow Compiler

General

- Hailo Dataflow Compiler now supports Hailo-15M device

Model Optimization

- Resolution reduction support for multiple input models

Compiler

- Improve Compilation time for large models in all hardware architectures for multi-context and single context networks

Post Processing

- YOLOv8 NMS is supported in all stages of emulation and compilation with engine=cpu

Parser

- Added support for LSTM bidirectional layers (PyTorch and ONNX only)

Hailo Model Zoo

Profiler Changes

- Removal of `--mode` flag from `hailomz profile` command, which generates a report according to provided HAR state
- Support KITTI Stereo Dataset

Models

- LaneAF-ERFNet - lane detection
- vit_pose_small - encoder based transformer with layernorm for pose estimation
- segformer_b0_bn - encoder based transformer with batchnorm for semantic segmentation

HailoRT

Hailo-15

- This version is supported on Hailo-15M

HailoRT post-processing support

- Added Yolov8 support

Async API

- Added C++ Async API support for VDevice (preview)
- Added Zero Copy optimization (when using Async API) which improves memory consumption for certain input formats

Multi-Process Service

- Fixed various stabilization issues - the service now supports wider and more various scenarios

API

- Added support for multi-planar input frames (based on V4L format)

TAPPAS

Example Applications

- Updated the Hailo-15 applications to use the updated Media Library implementation:
  - Basic Security Camera (streaming)
  - Detection
  - Single Stream OSD (On-Screen Display)
- Added a folder for external host scripts and added the UDP Stream Display script

---

Note: TAPPAS supports both Hailo-15 and Hailo-8. Temporarily, in this version, only the following Hailo-8 based example applications are supported:

- Detection
  - yolov5
  - yolov4
  - yolov3
  - mobilenet_ssd
- Multi-Stream Detection
  - Multi-Stream Detection
  - MultiStream Detection with Stream Multiplexer
- License Plate Recognition

These applications are supported under the general folder (x86-based platforms).

---

3. Suite Installation

The suite can be installed in one of three ways:

- Using all-in-one docker file (see: *Docker installation*)
- Using all-in-one self extracted executable (see: *Self Extracted Executable*)
- Manual installation of the packages in a defined order (see: *Manual installation*)

For interaction with Hailo Devices, a physical connection is required (although not required for installation):

- PCIe interface or Gigabit Ethernet (802.3) interface for the evaluation board
- M.2 connector for the M.2 board
- mPCIe connector for the mPCIe board

For the full compatibility table, see *Release versions compatibility*.

3.1. Docker installation

Note: For more information about Docker containers, see *Working With Docker Containers*.

3.1.1. System requirements

Note: Ubuntu 20.04 and Python 3.8 will no longer be supported in future versions.

In case of suite installation as a Docker file, the following are required:

1. Ubuntu 20.04/22.04, 64 bit
2. 16+ GB RAM (32+ GB recommended)
3. Docker package, either docker.io 20.10.07 (from Ubuntu repo), or docker-ce 20.10.6 (from Docker website).

Add the user to the docker group with the following steps:

1. Add your user to the docker group:

```
sudo usermod -aG docker ${USER}
```

2. Log out and log in.

In case you want to use Nvidia GPUs for hardware emulation and quantization, you also need to install:

1. Nvidia's Pascal/Turing/Ampere GPU architecture (such as Titan X Pascal, GTX 1080 Ti, RTX 2080 Ti, or RTX A4000)
2. GPU driver version 525
3. nvidia-docker2. It can be installed by running the following commands:


```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
&& curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-
↪key add - \
&& curl -s -L \
https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.
↪list \
| sudo tee /etc/apt/sources.list.d/nvidia-docker.list

sudo apt-get update
sudo apt-get install -y nvidia-docker2
sudo systemctl restart docker
```

3.1.2. Running the docker file

1. Download the suite from Developer Zone
2. Download PCIe driver from Developer Zone
3. Install PCIe driver by running the following command:

   ```
   sudo dpkg -i <pcie_driver>.deb
   ```

4. Reboot the computer
5. Extract the suite archive, then run the script - which opens a new container, and attaches to it ("gets inside it"):

   ```
   unzip hailo_ai_sw_suite_<version>.zip
   ./hailo_ai_sw_suite_docker_run.sh
   ```

   It is possible to enable/disable HailoRT features inside the Suite Docker. To see available flags, run:

   ```
   ./hailo_ai_sw_suite_docker_run.sh --help
   ```

   For example, to enable the HailoRT **multi-process service**, run:

   ```
   ./hailo_ai_sw_suite_docker_run.sh --hailort-enable-service
   ```

6. Run *pip list | grep hailo* to see the Hailo packages that are installed in the activated virtual environment
7. Run *hailo -h* to see a list of the available CLI commands of Hailo Dataflow Compiler and HailoRT

You can exit the docker by using the *exit* command, and then attach to it using the commands shown on *Working With Docker Containers*.

Another option is to use the script's additional options:

- **Resume** – go back to the existing container:

  ```
  ./hailo_ai_sw_suite_docker_run.sh --resume
  ```

- **Override** – delete the existing container and create new one:

  ```
  ./hailo_ai_sw_suite_docker_run.sh --override
  ```

Note: We have created a shared folder between the host system and the docker system. The path inside the docker is `/local/shared_with_docker/`, and the path outside is `./shared_with_docker` folder that is created on the same directory where hailo_ai_sw_suite_docker_run.sh is run.

Note: To validate that the PCIe driver was installed successfully, run *lspci | grep Co-processor*. For more information refer to *HailoRT User Guide / Installation / Validating the PCIe driver was successfully installed on Linux*.

3.1.3. Docker Suite Upgrade

1. Copy all private files from the currently used container in case those will be needed in the new container.
2. Make sure to close previously activated container.
3. Follow the *Docker installation* section in order to set up the new Hailo AI SW Suite docker container.
4. Copy the previously copied files from the previously used container to the newly set up container.

3.2. Self Extracted Executable

Hailo AI SW Suite self extracted executable will install all SW suite components directly into the system.

Note: TAPPAS component requires a dedicated flag.

3.2.1. System requirements

The following requirements are needed for the installation:

1. Ubuntu 20.04/22.04, 64 bit
2. 16+ GB RAM (32+ GB recommended)
3. Python 3.8/3.9/3.10, including `pip` and `virtualenv`
4. `python3.X-dev` and `python3.X-distutils` (according to the Python version), `python3-tk`, `graphviz`, and `libgraphviz-dev` packages. Use the command `sudo apt-get install PACKAGE` for installation
5. build-essential package (needed for compiling the PCIe driver)
6. For TAPPAS: `ffmpeg`, `x11-utils`, `python-gi-dev`, `libgirepository1.0-dev`, `libzmq3-dev`, `gcc-12` and `g++-12` apt packages,

   `git`, `Opencv4`, `Gstreamer`, `pygobject`, `opencv_flann`, `calib3d` and `features2d`

11. (Optional) bison, flex, libelf-dev and dkms packages (needed to register the PCIe driver using DKMS)
12. (Optional) cmake (needed for compiling the HailoRT examples and for the TAPPAS installation

    when using the self extracted executable)
13. (Optional) Node.js (minimum version: v20.9.0)- required for the DFC Studio

To install TAPPAS apt packages, run the following:

```
sudo apt-get install -y ffmpeg x11-utils libgstreamer-plugins-base1.0-dev python-gi-
↪dev libgirepository1.0-dev libzmq3-dev gcc-12 g++-12
```

To install OpenCV:


```
# Download Opencv and unzip
wget https://github.com/opencv/opencv/archive/4.5.2.zip
unzip 4.5.2.zip

# cd and make build dir
cd opencv-4.5.2
mkdir build
cd build

# Make and install
cmake -DOPENCV_GENERATE_PKGCONFIG=ON \
  -DBUILD_LIST=core,imgproc,imgcodecs,calib3d,features2d,flann \
  -DCMAKE_BUILD_TYPE=RELEASE \
  -DWITH_PROTOBUF=OFF -DWITH_QUIRC=OFF \
  -DWITH_WEBP=OFF -DWITH_OPENJPEG=OFF \
  -DWITH_GSTREAMER=OFF -DWITH_GTK=OFF \
  -DOPENCV_DNN_OPENCL=OFF -DBUILD_opencv_python2=OFF \
  -DINSTALL_C_EXAMPLES=ON \
  -DINSTALL_PYTHON_EXAMPLES=ON \
  -DCMAKE_INSTALL_PREFIX=/usr/local ..
make -j4
sudo make install

# Update the linker
sudo ldconfig
```

To install Gstreamer:

```
sudo apt-get install -y libcairo2-dev libgirepository1.0-dev libgstreamer1.0-dev \
libgstreamer-plugins-base1.0-dev libgstreamer-plugins-bad1.0-dev gstreamer1.0-
↪plugins-base \
gstreamer1.0-plugins-good gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly
↪gstreamer1.0-libav \
gstreamer1.0-doc gstreamer1.0-tools gstreamer1.0-x gstreamer1.0-alsa gstreamer1.0-
↪gl \
gstreamer1.0-gtk3 gstreamer1.0-qt5 gstreamer1.0-vaapi gstreamer1.0-pulseaudio \
gcc-12 g++-12 python-gi-dev
```

For further information, please refer to Gstreamer installation guide

To install pygobject:

```
sudo apt install python3-gi python3-gi-cairo gir1.2-gtk-3.0
```

For further details, please refer to pygobject installation guide

The following additional requirements are needed for GPU based hardware emulation:

1. Nvidia's Pascal/Turing/Ampere GPU architecture (such as Titan X Pascal, GTX 1080 Ti, RTX 2080 Ti, or RTX A4000)
2. GPU driver version 525
3. CUDA 11.8
4. CUDNN 8.9
5. (Recommended for TensorFlow) AVX instructions support on CPU

3.2.2. Installation

The executable is an archive that contains all required Hailo SW and an embedded installation script. The file will be named according to the following convention:

```
hailo_ai_sw_suite_<version>.run
```

For example:

```
hailo_ai_sw_suite_2024-10.run
```

After the executable is launched, the following will be present on the system:

- A new virtual environment named **"hailo_venv"** will be created and will contain the following:
    1. HailoRT Python package and its dependencies
    2. DFC Python package and its dependencies
    3. ModelZoo Python package and its dependencies
    4. All Python packages specified in the included requirements.txt file
    5. A target directory will be created containing all suite contents
- Hailo docs will be present in the directory from which the executable was launched
- Hailo PCIe driver installed

Usage:

First, make sure the file is executable by running `ls -l`. If not, turn it into an executable with:

```
sudo chmod 770 hailo_ai_sw_suite_<version>.run
```

The following command will extract the archive contents to `hailo_ai_sw_suite` directory and will install Hailo AI SW suite:

```
./hailo_ai_sw_suite_<version>.run
```

Example:

```
./hailo_ai_sw_suite_2024-10.run
```

For those who only want to extract and keep archive contents:

```
./hailo_ai_sw_suite_<version>.run --noexec
```

To run an internal validation hook after the installation is completed:

```
./hailo_ai_sw_suite_<version>.run -- validate-installation
```

To install TAPPAS, specify the `install-tappas` flag at the end of the command:

```
./hailo_ai_sw_suite_<version>.run -- install-tappas
```

To install HailoRT and enable the **multi-process service**, add:

```
./hailo_ai_sw_suite_<version>.run -- activate-hailort-service
```

After the installation is complete, activate the newly created virtual environment:

```
source hailo_ai_sw_suite/hailo_venv/bin/activate
```


3.3. Manual installation

3.3.1. System requirements

Each package comes with its own requirements. See more details in Dataflow Compiler requirements, Model Zoo requirements, and HailoRT and TAPPAS requirements in their documentation, accordingly.

3.3.2. Downloading Hailo software packages

1. Download Dataflow Compiler, HailoRT, and TAPPAS from Developer Zone.
2. Clone Hailo Model Zoo Git repository https://github.com/hailo-ai/hailo_model_zoo.

3.3.3. Packages installation

1. Install Dataflow Compiler: See `Dataflow Compiler Installation` on the Dataflow Compiler user manual.
2. Install HailoRT in same virtual environment as Dataflow Compiler. See `Ubuntu Installation` on the HailoRT User Guide.
3. Install Model Zoo in same virtual environment as Dataflow Compiler. See Hailo Model Zoo Getting Started page.
4. For TAPPAS installation see the TAPPAS User Guide.

3.3.4. Sanity tests

1. Validate Dataflow Compiler installation

```
hailo parser tf ~/workspace/dataflow_compiler/tutorials/models/resnet_
↪v1_18.ckpt --start-node-names resnet_v1_18/conv1/Pad --end-node-
↪names resnet_v1_18/predictions/Softmax
hailo optimize resnet_v1_18.har --use-random-calib-set
hailo compiler resnet_v1_18_quantized.har
```

2. Validate HailoRT installation

```
hailortcli fw-control identify
```

3. To validate TAPPAS and Model Zoo, follow the installation validation in their installation guides, accordingly.

4. Release versions compatibility

Hailo SW products are compatible with each other on specific versions. When upgrading a product, the others should be updated accordingly. The preferred option is to use Hailo SW Suites - both the **AI SW Suite** and the **Vision Processor Software Package** (**VPSP**), which align compatible versions.

4.1. Accelerators

Note: Refers to Hailo-8 and Hailo-8L.

Table 1. Release versions compatibility for Accelerators

AI SW Suite	Dataflow Compiler	HailoRT	Integration Tool	Model Zoo	TAPPAS
2024-10	v3.29.0	v4.19.0	v1.19.0	v2.13.0	v3.30.0
2024-07.1	v3.28.0	v4.18.0	v1.18.0	v2.12.0	v3.29.1
2024-07	v3.28.0	v4.18.0	v1.18.0	v2.12.0	v3.29.0
2024-04	v3.27.0	v4.17.0	v1.17.0	v2.11.0	v3.28.0
2024-01	v3.26.0	v4.16.0	v1.16.0	v2.10.0	v3.27.0
2023-10	v3.25.0	v4.15.0	v1.15.0	v2.9.0	v3.26.0
2023-07.1	v3.24.0	v4.14.0	v1.14.1	v2.8.0	v3.25.0
2023-07	v3.24.0	v4.14.0	v1.14.0	v2.8.0	v3.25.0
2023-04	v3.23.0	v4.13.0	v1.13.0	v2.7.0	v3.24.0
2023-01.1	v3.22.1	v4.12.1	v1.12.0	v2.6.1	v3.23.1
2023-01	v3.22.0	v4.12.0	v1.12.0	v2.6.0	v3.23.0
		v4.11.0		v2.5.0	v3.22.0
2022-10	v3.20.0	v4.10.0	v1.10.0	v2.4.0	v3.21.0
		v4.9.0			v3.20.0
	v3.19.0	v4.8.1		v2.3.0	
2022-07.1	v3.18.1	v4.8.1	v1.8.0	v2.2.0	v3.19.1
2022-07	v3.18.0	v4.8.0	v1.8.0	v2.2.0	v3.19.0
	v3.17.0	v4.7.0		v2.1.0	v3.18.0
2022-04	v3.16.0	v4.6.0	v1.6.0	v2.0.0	v3.17.0
	v3.15.0	v4.5.0	v1.5.0		v3.16.0
	v3.15.0	v4.4.0	v1.4.0		v3.15.0
	v3.14.0	v4.4.0	v1.4.0	v1.5	v3.15.0
2022-01	v3.14.0	v4.3.0		v1.4	v3.14.0

Note: The HailoRT column relates both to the HailoRT library and the driver. The Integration Tool column relates to Hailo Accelerator Integration Tool.

4.2. Vision Processor Units (Hailo-15H, Hailo-15M)

Note: Refers to Hailo-15H and Hailo-15M.

Table 2. Release versions compatibility for VPUs

AI SW Suite	Dataflow Compiler	HailoRT	Model Zoo	TAPPAS	VPSP Suite	VPSP sub-packages
2024-10	v3.29.0	v4.19.0	v2.13.0	v3.30.0	2024-10	1.5.0
	v3.28.0	v4.18.0	v2.12.0	v3.29.0	2024-07.1	1.4.1
2024-07	v3.28.0	v4.18.0	v2.12.0	v3.29.0	2024-07	1.4.0
		v4.17.1		v3.28.1	2024-04.1	1.3.1
2024-04	v3.27.0	v4.17.0	v2.11.0	v3.28.0	2024-04	1.3.0
	v3.26.0	v4.16.0	v2.10.0	v3.27.0	2024-01.2	1.2.2
	v3.26.0	v4.16.0	v2.10.0	v3.27.0	2024-01.1	1.2.1
2024-01	v3.26.0	v4.16.0	v2.10.0	v3.27.0	2024-01	1.2.0
2023-10	v3.25.0	v4.15.0	v2.9.0	v3.26.0	2023-10	1.1.0

5. Working with Docker Containers

5.1. Docker common commands

Docker is an open-source project that allows deployment of portable applications as containers, using OS-level virtualization. One of the Hailo AI SW Suite installation methods is using a Docker file, therefore we appended some common Docker commands.

> **Note:** In order to run all "docker" commands without "sudo", you'll have to add your user to group "docker"; instructions on how to do that can be looked up in "Suite installation" section.

1. Display the existing images

```
sudo docker images
```

2. Display the existing containers (both currently running and stopped)

```
sudo docker container ls -a
```

3. Start an existing container

```
sudo docker container start -i $container_name
```

4. Attach to a started container ("get inside it")

```
sudo docker container attach $container_name
```

5. Exit the container and stop it

```
exit
```

6. Stop a container

```
sudo docker container stop $container_name
```

7. Remove a container (it must be stopped first)

```
sudo docker container rm $container_name
```

8. Remove all stopped containers

```
sudo docker container prune
```

9. Remove an image (there should not be any containers or images based on it or its layers; image name is REPOSITORY:TAG when viewed with *docker images*)

```
sudo docker image rm $image_name
```

10. Remove all images (the images which currently have containers based on them will not be removed)

```
sudo docker image prune
```

11. Copy files and directories between host and container (can be done with both started and stopped containers). On Hailo AI SW Suite container, your workspace is on /local/workspace/

```
sudo docker cp $path_on_host $container_name:$path_inside_container
sudo docker cp $container_name:$path_inside_container $path_on_host
```

12. Attach additional bash terminal to a running container


```
sudo docker exec -it $container_name bash
```

5.2. Docker Containers and VSCode integration

Visual Studio Code is a common source-code editor made by Microsoft for Windows, Linux and macOS. It supports many programming languages and features a variety of plugins. For anyone who is willing to use VSCode (which is external to the Docker) to work with the contents of a Docker:

1. Start VSCode
2. Navigate to Extensions
3. Install the following extensions:
    1. Docker (by Microsoft)
    2. Remote - Containers (by Microsoft)
4. Now there should be an extra “Docker” icon on the left pane of VSCode
5. When you click “Docker” icon, you’ll be able to browse existing docker images and containers and have some additional actions available for you by right-clicking them, like starting, stopping, attaching, removing existing containers
6. Unfortunately there is no easy way to properly create (with all required arguments) new container from “Hailo Software Suite” image by right-clicking the image in VSCode; in order to work with “Hailo Software Suite” via VSCode, you’ll have to first create a container from terminal using the “hailo_ai_sw_suite_docker_run.sh” and then you’ll be able to “Attach Visual Studio Code” to the created container.

6. Known Issues

6.1. 2024-10 Suite

This section is relevant for 2024-10 Suite, and accompanying released packages:

- Dataflow Compiler v3.29.0
- Hailo Model Zoo v2.13.0
- HailoRT v4.19.0
- TAPPAS v3.30.0

6.1.1. Dataflow Compiler

- No known issues for now

6.1.2. Hailo Model Zoo

- No known issues for now

6.1.3. HailoRT

Note: Starting this version, Hailo-8 Ethernet-based platforms support is deprecated, so communication between the host and Hailo-8 is supported only over PCIe. HailoRT 4.18.0 was the last version to support Hailo-8's Ethernet interface.

6.1.4. TAPPAS

- Some example application HEFs are not aligned to Model Zoo HEFs (for example - `yolov5m_wo_ssp_60p` Detection)

