HAILO

Empowering Intelligence

Hailo SW Suite User Guide

Release 2023-07.1

16 July 2023

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 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

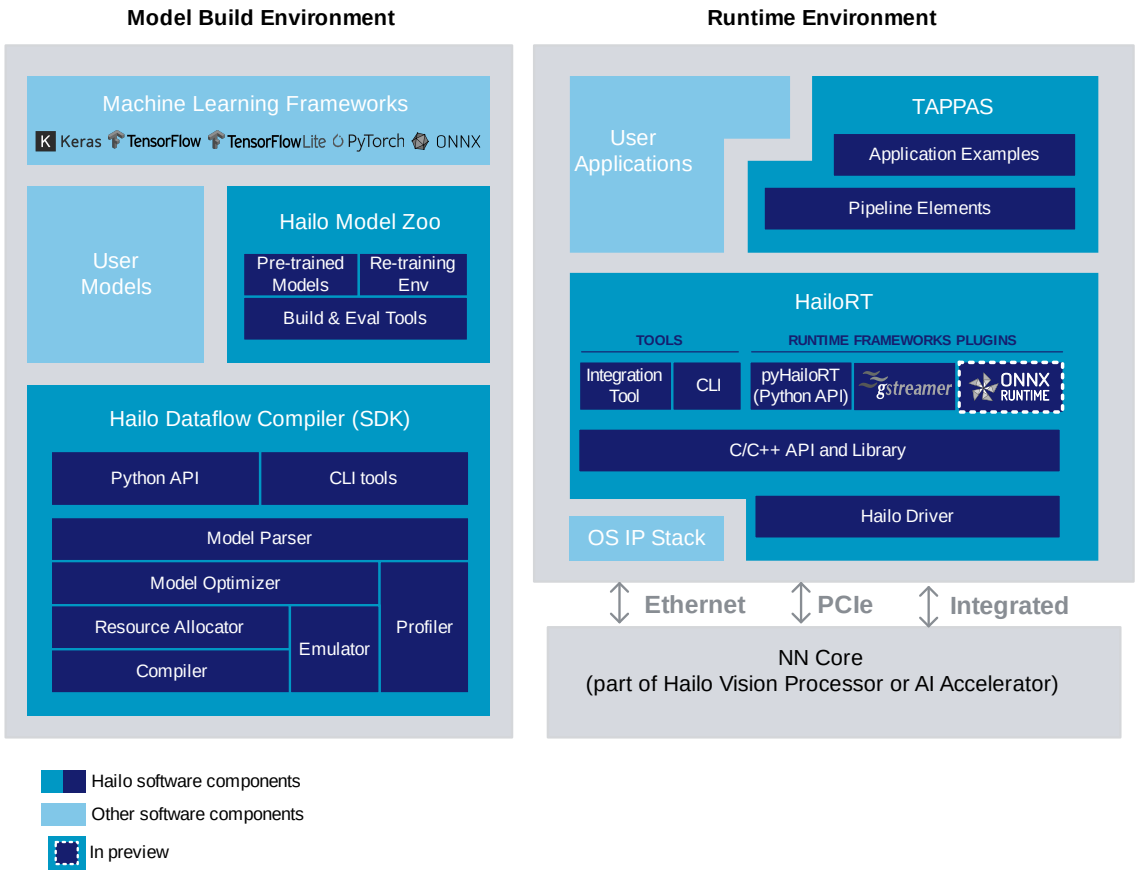

Figure 1. Detailed block diagram of Hailo software packages

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

1. Install the Hailo device. If you are using the M.2 module, refer to the Hailo-8 M.2 hardware installation guide.
2. Install the latest suite: *Suite Installation*.
3. Use Hailo Model Zoo usage guide to compile, run and evaluate pre-trained models on your device, using a convenient CLI tool.

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

4. Use TAPPAS usage guide to see various real-time demos in action using your PC's camera.

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

5. If Hailo Model Zoo models are good fit for your application, you can use your own dataset for retraining the models using Hailo Retraining Dockers.
6. If you'd like to use your own pre-trained model, use the Dataflow Compiler to convert the model to Hailo binary format. See the Tutorials section.
7. See HailoRT Tutorials section and API Reference for information about using HailoRT API in your realtime application.

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

2. 2023-07.1 Hailo SW Suite - New Features

- Installed missing CUDA driver components

2.1. Previous Suite Versions

2.1.1. 2023-07 Hailo SW Suite

Dataflow Compiler

General

- Hailo Dataflow Compiler now supports the newly-released Hailo-15H device

Kernels and Activations

- 16bit precision mode can be applied to Conv layers inside the model to increase their accuracy

Profiler

- You can use the new profiler HTML design, by appending the `--use-new-report` flag to the CLI command (preview; will be default starting 2023-10)

Full Precision Optimization

- Add support to fully connected layers with big amount of parameters

High-level and Documentation

- NMS auto detection - YOLOv5 NMS with custom anchors is automatically detected and can be applied to the model easily
- Added set_seed command for reproducing of quantization results, affects the seed of tensorFlow, numpy, and python.random libraries (preview)
- Apply sigmoid automatically whenever NMS is added

Compiler

- Improved Performance Mode algorithm
- Improved FPS on models that are compiled to Hailo-15H

Hailo Model Zoo

- updated to use Dataflow Compiler v3.24.0
- Updated to use HailoRT 4.14.0
- The Hailo Model Zoo now supports the following vision transformers models:
  - vit_tiny / vit_small / vit_base - encoder based transformer with batchnorm for classification
  - detr_resnet_v1_18_bn - encoder/decoder transformer for object detection
  - clip_resnet_50 - Contrastive Language-Image Pre-Training for zero-shot classification
  - yolov5s_c3tr - object detection model with a MHSA block
- Using HailoRT-pp for postprocessing of the following variants:
  - yolov5
  - yolox
  - ssd

  - efficientdet
  - yolov7
- New Models:
  - repvgg_a1 / repvgg_a2 - classification
  - yolov8_seg: yolov8n_seg / yolov8s_seg / yolov8m_seg - instance segmentation
  - yolov6n_0.2.1 - object detection
  - zero_dce - low-light enhancement
- New retraining dockers for:
  - yolov8
  - yolov8_seg
- Enable compilation for hailo15h device
- Enable evaluation of models with RGBX / NV12 input format

HailoRT

Hailo-15

- This version is supported on Hailo-15H

HailoRT post-processing support (release)

- Added Yolox support
- Added Argmax support (preview), for single-output models

Model Scheduler

- Model `priority` is now supported (release)

CLI

- `hailortcli run2` is now released
- Added `hailortcli run2` functionality for latency measurements

Examples

- Added support for Hailo-15 (relevant examples only)
- Changed various examples to show a wider range of use-cases

Windows

- Multi-Process service can now work with PyHailoRT (preview), which enables using the Python streaming API (send/recv)

API

- Async API - added the C/C++ Async API for raw streams (preview)

TAPPAS

Example Applications Updates and Optimizations

- Improved Yolov5seg post-process performance
- Updated Yolo networks to use the HailoRT native post-process (selected models)
- Added `non-blocking mode` and `wait-time` properties to `hailoroundrobin` element

2.1.2. 2023-04 Hailo SW Suite

Dataflow Compiler

General

- Netron now supports the Hailo Archive (.har) format

Parser

- HailoRT supports SSD NMS post processing on the host platform, using the nms_postprocess model script command
- Supporting additional topologies when compiling models for Hailo's ONNX Runtime plugin

Compiler

- Introducing Performance Mode, for pushing resource utilization limits to achieve high FPS (preview, single context only)

Kernels and activations

- Transformer building block Multi head Attention is now supported, when parsed from `torch.nn.MultiheadAttention` API in PyTorch 1.11 (preview)
- Added support for on-chip i420->YUV conversion

Model Optimization

- Quantization-Aware-Training is supported using the new runner.set_keras_model() API. More details are available on the Quantization-Aware-Training Tutorial
- Supporting full-net 16-bit, in case all layers are supported (preview)
- 16-bit output layer is enabled automatically when supported, for small output tensors
- Bias Correction algorithm is used as default (optimization_level=1)
- Improved GPU memory consumption during the optimization phase
- Improved the FineTune algorithm for models with multiple output nodes
- Added emulation support for RGBX->RGB, NV12->YUV, NV21->YUV and i420->YUV conversions
- Added support for emulating YOLOv5 and SSD NMS with *engine=cpu*
- The new API *runner.infer* replaces the *runner.get_tf_graph* API

Profiler

- The HTML profiler now displays a quick version of the layer analysis tool (*Accuracy* tab) automatically

Documentation and Tutorials

- Layer Analysis Tool Tutorial has been updated to demonstrate how to increase accuracy

Hailo Model Zoo

New Models

- New transformers-based models:
  - vit_base - classification model
  - yolov5s_c3tr - object detection model with a self-attention block
- New state of the art YOLOv8 models: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, YOLOv8x
- New state of the art DAMO-YOLO models: damoyolo_tinynasL20_T, damoyolo_tinynasL25_S, damoyolo_tinynasL35_M

General

- Examples for using HailoRT post process - support for seamless integration of models and their corresponding postprocessing
  - yolov5m_hpp
  - ssd_mobilenet_v1_hpp
- Configuration YAMLs and model-scripts for networks with YUY2 input format
- DAMO-YOLO retraining Docker
- Bug fixes

HailoRT

HailoRT post-processing support (preview)

- HailoRT now supports running SSD post-processing on the host CPU

Model Scheduler

- **Multi-Device** Model Scheduler is now enabled by default for C/C++ API
- Added support for model `priority` (preview)

Note: Model Scheduler will be default for PyHailoRT in future versions, for Linux and Windows.

CLI

- Added various `hailortcli run2` functionality – including multiple-device inference, and measurements. See `hailortcli run2 --help` for more information
- Monitor is now released, with metric updates and support for multiple-devices

Windows

- Windows 11 is now supported
- Multi-Process service is now supported for C/C++ API (preview)

API

- Python API
  - Model Scheduler and Multi-Process service are now supported (Linux only)

TAPPAS

New Example Applications

- Added another VMS pipeline which supports multiple devices (supported on Ubuntu 22.04 only)
- New applications for i.mx8 - Object Detection and Pose Estimation (cascaded) and Multi-Stream Detection

Example Applications Updates and Optimizations

- VMS (x86_hw_accelerated) single device - this example application now works on Ubuntu 20.04 and is part of the Hailo SW Suite
- Instance segmentation example application - added support for yolov5seg (Improved performance)
- Century - added yolox for the general use-case

New Platforms

- Added support for Rockchip RK3588 with example applications - Detection, LPR, Tiling and Multi-Stream Detection

Infrastructure

- Added a TAPPAS Graphic User Interface to easily run selected general example applications on the TAPPAS Docker (preview) - to activate it, run *tappas-gui*
- Reduced Docker size

2.1.3. 2023-01.1 Hotfix

Dataflow Compiler

- Added support for Global Maxpool operator in ONNX parser
- Fixed an issue that has prevented YOLOv8 from parsing
- Fixed prints to screen during compilation, regarding single/multi context flow and resources utilization

Hailo Model Zoo

- Bug fixes

HailoRT

- Fixed a bug which might cause a race condition in multi context HEFs with NMS output
- Fixed a bug which sometimes caused a timeout when deactivating a network while running inference
- Fixed a bug in HailoNet which caused a memory leak
- Fixed a bug in `hailortcli` which caused some default flags to be ignored
- Fixed a bug which prevented VStream recovery-after-timeout, when NMS runs in the NN core

TAPPAS

- Aligned version to HailoRT 4.12.1

2.1.4. 2022-01 Hailo SW Suite

Dataflow Compiler

Package updates

- Added support for Ubunbu 22.04, Python 3.9, and Python 3.10
- Ubuntu 18.04, Python 3.6 and Python 3.7 are no longer supported
- Updated Tensorflow, ONNX, ONNXRuntime, PyTorch package versions

Parser

- HailoRT post-processing support, activated using a new 'engine' flag in the nms_postprocess model script command in the Dataflow Compiler side. This flag instructs HailoRT to complete YOLOv5 NMS post-processing on the host platform (preview)

Profiler

- Introducing Accuracy Tab on the HTML Profiler, to be used as a tool to analyse and improve accuracy

Compiler

- Optimized the compiler for better stability and performance

Documentation and Tutorials

- Updated the Model Optimization workflow section with simple and advanced optimization flows
- Updated the Model Optimization Tutorial with step-by-step instructions for validating accuracy through the optimization and compilation phases
- Updated the Layer Analysis Tool tutorial to utilize the new HTML profiler Accuracy tab

Kernels and activations

- Added support for on-chip NV12->RGB, NV21->RGB and YUV->BGR format conversions
- Further increased support for Bilinear and Nearesr Neighbor Resize, and Deconvolution layers
- Add support for EfficientGCN pooling block
- Added support for Less operator
- Add support for dual broadcast in element-wise mult (Hx1xC * 1xWxC -> HxWxC)
- Added support for multiplication by 0: (x*0, x*0+b, (x+b)*0)
- Added support for depthwise with depth multiplier as group convolution in TFLite
- Add support for ADD_N from TFLite models

Hailo Model Zoo

New Models

- ViT (Vision Transformer) - new classification network with transformers-encoder based architecture
- New instance segmentation variants:
  - yolov5n_seg
  - yolov5s_seg
  - yolov5m_seg
  - yolov5l_seg
- New yolov5 detecion variants for high resolution images:
  - yolov5n6_6.1

  - yolov5s6_6.1
  - yolov5m6_6.1
  - yolov7e6

API changes

- New `--performance` flag to reproduce highest performance for a subset of networks

HailoRT

Model Scheduler

- Model Scheduler is enabled by default for the C/C++ API, `hailortcli` and HailoNet (GStreamer element). Working without the scheduler is still fully supported – see the HailoRT documentation regarding how to disable it
- Added support for **multi-device** Model Scheduler (preview) - VDevice with multiple physical devices and Model Scheduler enabled
- Stream Multiplexer is now released

Multi-Process service

- Multi-Process service is now released (C/C++ API), which enables multi-process inference:
  - Over a single device per `group_id` – release grade
  - Via VDevice with more than 1 device – preview

HailoRT post-processing support (preview)

- HailoRT now supports running Yolov5 post-processing on the host CPU, under the HailoRT API

Note: Use the Dataflow Compiler to enable the host post-processing feature for a specific HEF.

Supported Environments

- Ubuntu 22.04 is now supported (release grade)

Yocto

- Added support for offline builds (preview)

Firmware

- **Ethernet** and **PCIe** are now supported via two separate types of firmware
- The Hailo-8 PCIe firmware is an optimization feature that increased the number of supported contexts/Network Groups that can be configured

CLI

- Introducing a new `run2` command (preview) for running multiple HEFs simultaneously (using the Model Scheduler)

  See `hailortcli run2 --help` for more details

TAPPAS

New Example Applications

- Added x86 (hardware accelerated) example pipelines that use Video Acceleration API (VA-API) over Intel processors that support Quick Sync:
  - Video Management System - a pipeline that demonstrates a VMS application which runs several streams and different tasks - Face Recognition, Face Attributes and Person Attributes. Currently this example pipeline is supported on Ubuntu 22.04 only
  - Multi-stream detection
  - Century
- Pose Estimation pipeline with two cascading networks
- Face recognition
- Added Depth Estimation, Object Detection and Classification pipelines for i.MX6 platforms

Example Applications updates and optimizations

- Added new models to Instance Segmentation Pipeline:
  - yolact_regnetx_1.6gf
  - yolact_regnetx_800mf (80 classes)
- Century app now uses a new network (yolov5m)
- Multi-Camera Multi-Person Tracking (RE-ID) - Improved pipeline performance and accuracy

Supported Environments and Frameworks

- Added support for Ubuntu 22.04 (release-grade)

2.1.5. 2022-10 Hailo SW Suite

Dataflow Compiler

Parser

- TFLite Parser is now release-grade, allowing to support many converted TF2 models

Model Optimization

- Quantization to 4-bit for 20% of the model weights is enabled by default for large models, allowing to reach higher performance at the same accuracy levels

Kernels and Activations

- Added support for on-chip RGBX->RGB and BGR->RGB conversions
- Added support for ONNX operator InstanceNormalization
- Added support for L2 Normalization layer on TensorFlow
- Added support for Log and Hard Sigmoid activations

Compiler

- Improved the performance of compiled models

High Level and API

- Simplified CLI output and log files
- Log files could be disabled by setting an environment variable
- HTML Profiler report includes model optimization information

- Windows support under WSL running Ubuntu (preview)
- Model modification commands additions:
  - Command for adding NMS on chip is simplified (preview)
  - New command for changing output activations
- Removed deprecated APIs

Hailo Model Zoo

New Models

- Face Detection - scrfd_500m / scrfd_2.5g / scrfd_10g
- YOLOv6n
- YOLOv7 / YOLOv7-tiny
- NanoDet_RepVGG_A1_640
- EfficientDet_Lite 0,1,2

New Tasks

- MSPN_RegnetX_800mf: Single person pose estimation (including Retraining Docker)
- face_attr_resnet_v1_18: Face attribute recognition
- Super-Resolution
  - Added support for BSD100 dataset
  - The following models were added: espcn_x2 / espcn_x3 / espcn_x4
- Face Recognition
  - Support for LFW dataset
  - The following models were added: arcface_r50, arcface_mobilefacenet (including Retraining Docker)

API changes

- Required FPS was moved from models YAML into the models scripts
- Model scripts use new change activation syntax

HailoRT

Supported Environments and Frameworks

- Added support for Yocto Kirkstone
- Ubuntu 22.04 is now supported for C/C++ API (preview)

Model Scheduler

- Model Scheduler is now released
- Added Stream Multiplexer support (preview)
- Added Monitor support (preview)

Multi-Process support

- HailoRT now supports a multi-process service, which enables multi-process inference (over a single device)

Firmware

- Added over-current throttling mechanism

CLI

- Operations will be executed on all connected devices (same as previously passing `-s *`) by default, unless a device ID is specified
- Added support for device IDs when using `-s`, to execute on the specified devices one-by-one
- In case of using `run` command with multiple device IDs, it is executed via a single VDevice which encapsulates the specified devices (and not one-by-one)
- Alignment of `hailo` command-line tool to `hailortcli`

TAPPAS

New Example Applications

- Multi-stream detection which uses HailoRT's Stream Multiplexer (preview)

Supported Environments and Frameworks

- Added support for Yocto Kirkstone
- Added support for Ubuntu 22.04 (preview)

Infrastructure

- Improved pipeline profiling and debugging capabilities

2.1.6. 2022-07.1 Hotfix

Dataflow Compiler

- Bug fixes

HailoRT

- Bug fixes

TAPPAS

- New Example Application: Multi-Camera Multi-Person Tracking (RE-ID) pipeline (preview)

2.1.7. 2022-07 Hailo SW Suite

Dataflow Compiler

Parser

- Parser now supports TFLite models (preview)
- ONNX models only: Parser now suggests start/end nodes when translation fails (preview)

Model Optimization

- **Introducing Optimization levels:**
  - Using a number between 0 to 3 (default=1), control the complexity of the optimization algorithm
- **Introducing Compression levels:**
  - Using a number between 0 to 5 (default=0), control the compression of the model

Kernels and Activations

- New filter sizes for 2D Convolution kernels (preview)
- New filter sizes for DepthWise Conv kernels (preview)

- New filter sizes for Average Pool kernels (preview)
- Broader Resize N.N support (preview)
- Broader Resize Bilinear support (preview)
- Improved degradation and performance of SiLU, Mish, Swish activations

Compiler

- Improved RAM usage during compilation
- Optimized the loading time of compiled models on hardware; Helps pipelines with frequent model switching
- Compilation performs better utilization of the device
- ONNX models only: Option to export an ONNX model that contains the compiled model. The model could be run using ONNX Runtime (preview)

Documentation and Tutorials

- Updated the parsing tutorial to demonstrate how to convert TF/TF2 models to TFLite
- Added explanation on how to export PyTorch models to ONNX

High Level and API

- Profiler accepts runtime_data.json file to create HTML runtime graph
- Added the input_conversion() model script command as the main API to add input conversion, such as yuv_to_rgb and yuv2_to_yuv

Hailo Model Zoo

General

- Updated to use Dataflow Compiler v3.18 (developer-zone)
- Parser commands were moved into model scripts
- Support Market-1501 Dataset

CLI change

- Hailo model zoo CLI is now working with an entry point - hailomz
- quantize sub command was changed to optimize
- Hailo model zoo data directory by default will be `~/.hailomz`

New models and tasks

- yolov5xs_wo_spp_nms - a model which contains bbox decoding and confidence thresholding on Hailo-8
- osnet_x1_0 - person ReID network
- yolov5m_6.1 - yolov5m network from the latest tag of the repo (6.1) including silu activation
- **Support a new model zoo task - ReID**
  - yolov5s_personface - person and face detector
  - repvgg_a0_person_reid_512 / repvgg_a0_person_reid_2048 - ReID networks which outputs a person embedding. These models were trained in-house as part of our upcoming new application
  - ReID training Docker for the Hailo model repvgg_a0_person_reid_512/2048
- stdc1 - Segmentation architecture for Cityscapes

HailoRT

Model Scheduler (preview)

- Automatic switch between different network groups is now supported (only for C, C++ APIs)

Supported Environments and Frameworks

- Python 3.9 is now supported
- Added support for Yocto Honister
- Yocto Hardknott is now fully supported
- Inference with ONNX Runtime is now supported (preview)

GStreamer

- Added HailoNet support for Model Scheduler

Examples

- Added power measurement example
- Removed `switch_hefs_example` (C, C++)
- Renamed `switch_hefs_example_threads_reuse` to `switch_network_groups_manually_examp` (C, C++)

CLI

- Added support for some `hailortcli` operations (like reset) which now can run on multiple PCIe devices one-by-one, by passing `-s *`. For more information, run `hailortcli --help`
- Added support for `--power-mode` in hailortcli benchmarks
- Removed redundant commands

TAPPAS

New Example Applications

- LPR (License Plate Recognition) pipeline
- Added Cascading networks, Depth Estimation, Pose Estimation and Semantic Segmentation pipelines for i.MX

Platforms

- Added support for Raspberry Pi Raspbian OS

Infrastructure

- Added an option to control post-process parameters via a JSON configuration for the detection application
- Added the ability of compiling a specific TAPPAS target (post-processes, elements)

Note: Ubuntu 18.04 will be deprecated in Hailo SW Suite future version

Note: Python 3.6 will be deprecated in Hailo SW Suite future version

2.1.8. 2022-04 Hailo SW Suite

Dataflow Compiler

Parser

- Added support for PyTorch's PixelShuffle as a DepthToSpace in CRD mode (ONNX) For further details take a look at the Using the ONNX Parser, Supported PyTorch APIs, and Depth to Space sections
- Added support for parsing Tensorflow models in NCHW format
- Parser now shows the original layer name when failing with UnsupportedOperation

Kernels

- Added support for many additional layers and operations
- **List of changes:**
  - Added support for additional Average Pooling cases
  - Added support for Conv 3x3, dilation 16x16, stride 1x1
  - Added support for SiLU, Swish activation types
  - Added support for Mish, Hard-Swish, GELU, PReLU, Tanh, Exp, Sqrt activation types (preview)
  - Added support for Square operator
  - Elementwise Add, Sub, Mul, Div enhanced support for different input types
  - 'VALID' padding scheme for depthwise 3x3, 3x5, 5x3, 5x5
  - Depthwise support for two input data tensors (for siamese networks)
  - Depthwise Convolution kernel optimization
  - Expand operator, as broadcast before elementwise operations
  - ReduceL2 layer, after rank4 tensors such as Conv
  - Conv6x6 stride 2 support

Compiler

- **Released features (from preview)**
  - Join wide support - `join()` API that unifies two models to be compiled together
  - Layer merging - resource optimization by merging two layers to use the same controller
- Added support for NMS with big models
- CenterNet BBox decoder + score threshold on-chip support, more info here
- YOLOv5 BBox decoder + score threshold on-chip support, more info here

Model Optimization

- Improved support for 16 bit quantization, applied to the model's last layer

High-Level and Documentation

- New documentation for adding on-chip NMS post-processing
- Enhanced HTML profiler report

Dataflow Compiler API

- Normalization, Transpose, YUV2RGB, Resize operations are now performed using a model script
- For supported post processing types (such as NMS), adding post processing using a Model Script file
- Deprecated .hn format support, Hailo Archive (.har) format should be used. Hailo CLI tool commands now output .har by default

Hailo Model Zoo

- **Retraining Dockers - use for retraining with custom datasets:**
  - YOLOv3
  - NanoDet
  - CenterPose
  - YOLACT
  - YOLOv4
  - YOLOv5 - better support
  - YOLOx
  - FCN
- **Introducing Hailo Models - in house pretrained networks with compatible Dockerfile for easy retraining:**
  - yolov5m_vehicles
  - tiny_yolov4_license_plates
  - lprnet (license plate recognition)
- New models added to the Model Zoo. See the complete list on *Hailo Model Zoo*.
- **Added support for datasets:**
  - 300W-LP and AFLW2k3d datasets
  - Oxford-IIIT Pet Dataset
- **Multi-network pipeline support**
  - Example: detection_pose_estimation that uses yolov5m + centerpose_repvgg_a0
- Updated to use HailoRT 4.6 and Dataflow Compiler v3.16 (TF v2.5, CUDA v11.2)

HailoRT

General

- HailoRT is now open-source - see HailoRT Github
- All example files (C/C++/Python) have been moved to Github
- Unified the Linux and Windows user guides and revised the installation section
- Device virtualization (VDevice) is now released
- Added multiple_device_example that demonstrates how to work with multiple devices using virtual device (VDevice)

CLI

- Added new `hailortcli` functionality such as parse-hef (get information from the HEF file about its components) and support for run/benchmark on multiple devices Windows
- `hailortcli` now prints results per network, when multiple networks are used
- Added a progress bar per network to `hailortcli` infer

Windows

- Added Python support (preview)
- C++ API is now supported

Yocto

- Removed support for older Yocto versions (Sumo, Warrior)
- Added support for Yocto Hardknott (preview)

API

- Data quantization functions (quantizaiton.hpp) are now exported for C++
- Inference pipeline functions are now exported for C++

GStreamer

- Added user controlled switch support and VDevice support in GStreamer plugin
- HailoNet GStreamer plugin will not be active if there is only one HailoNet and `is-active` property is false
- Prevent multiple HailoNets with the same network

TAPPAS

General

- Code is now publicly available on Github
- Updated infrastructure to use new HailoRT installation packages
- Yocto Hardknott is now supported
- Added Raspberrypi 4 Ubuntu dedicated apps
- Added support for hosts without X-Video adapter

New Apps

- Detection and depth estimation - Networks switch app
- Multi-device and Hailo Century app - Demonstrates detection on one video file source over 6 different Hailo-8 devices
- Python app - A classification app using a post-process written in Python
- LPR (License Plate Recognition) pipeline (preview)
- Detection & pose estimation app
- Detection (MobilenetSSD) - Multi scale tiling app

Elements

- Tracking element “HailoTracker” - Add tracking capabilities
- Python element “HailoPyFilter” - Enables to write post-processes using Python
- HailoCropper now accepts cropping method as a shared object (.so)

3. Suite Installation

The suite can be installed in one of three ways:

- Using all-in-one docker file (see: *Docker installation*)
- Preview: 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

3.1. Docker installation

Note: For more information about dockers, see *Working With Dockers*.

3.1.1. System requirements

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 470
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_sw_suite_<version>.zip
./hailo_sw_suite_docker_run.sh
```

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

```
./hailo_sw_suite_docker_run.sh --help
```

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

```
./hailo_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 Dockers*.

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

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

```
./hailo_sw_suite_docker_run.sh --resume
```

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

```
./hailo_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_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

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*.

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 SW Suite docker container.
4. Copy the previously copied files from the previously used container to the newly set up container.

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*.

3.2. Self Extracted Executable

Hailo SW Suite self extracted executable will install all SW suite components directly into the system. This method is Preview on this Suite version.

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 to compile the PCIe driver)
6. For TAPPAS: `ffmpeg`, `x11-utils`, `python-gi-dev`, `libgirepository1.0-dev`, `libzmq3-dev`, `gcc-9` and `g++-9` 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 to compile the HailoRT examples)

To install TAPPAS apt packages simply 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-9 g++-9
```

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
```

(continues on next page)

(continued from previous page)

```
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-9 g++-9 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 470
3. CUDA 11.2
4. CUDNN 8.1
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_sw_suite_<version>.run
```

For example:

```
hailo_sw_suite_2023-01.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 it's dependencies
    2. DFC Python package and it's dependencies
    3. ModelZoo Python package and it's 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_sw_suite_<version>.run
```

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

```
./hailo_sw_suite_<version>.run
```

Example:

```
./hailo_sw_suite_2023-01.run
```

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

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

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

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

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

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

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

```
source hailo_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 Suite, which aligns compatible versions.

Table 1. Release versions compatibility

SW Suite version	Dataflow Compiler version	HailoRT version	Firmware version	Model Zoo version	TAPPAS version
2023-07.1	v3.24.0	v4.14.0	v4.14.0	v2.8.0	v3.25.0
2023-07	v3.24.0	v4.14.0	v4.14.0	v2.8.0	v3.25.0
2023-04	v3.23.0	v4.13.0	v4.13.0	v2.7.0	v3.24.0
2023-01.1	v3.22.1	v4.12.1	v4.12.1	v2.6.1	v3.23.1
2023-01	v3.22.0	v4.12.0	v4.12.0	v2.6.0	v3.23.0
		v4.11.0	v4.11.0	v2.5.0	v3.22.0
2022-10	v3.20.0	v4.10.0	v4.10.0	v2.4.0	v3.21.0
		v4.9.0	v4.9.0		v3.20.0
	v3.19.0	v4.8.1	v4.8.1	v2.3.0	
2022-07.1	v3.18.1	v4.8.1	v4.8.1	v2.2.0	v3.19.1
2022-07	v3.18.0	v4.8.0	v4.8.0	v2.2.0	v3.19.0
	v3.17.0	v4.7.0	v4.7.0	v2.1.0	v3.18.0
2022-04	v3.16.0	v4.6.0	v4.6.0	v2.0.0	v3.17.0
	v3.15.0	v4.5.0	v4.5.0		v3.16.0
	v3.15.0	v4.4.0	v4.4.0		v3.15.0
	v3.14.0	v4.4.0	v4.4.0	v1.5	v3.15.0
2022-01	v3.14.0	v4.3.0	v4.3.0	v1.4	v3.14.0
	v3.13.1	v4.2.0	v4.2.0		v3.13.0
	v3.13.0	v4.2.0	v4.2.0		v3.13.0
	v3.12.0	v4.1.0, v4.1.1	v4.1.0, v4.1.1	v1.3	v3.12.0
	v3.11.2	v4.0.1	v4.0.1		v3.11.0
	v3.11.1	v4.0.1	v4.0.1		v3.11.0
	v3.11.0	v4.0.1	v4.0.1	v1.2	v3.11.0
2021-10	v3.11.0	v4.0.1	v4.0.1	v1.1	v3.11.0
	v3.10.1	v2.10.0	v2.10.0		v3.10.0
	v3.10.0	v2.9.0	v2.9.0		v3.10.0
	v3.9.1	v2.9.0	v2.9.0		v3.9.0
	v3.9.0	v2.9.0	v2.9.0		v3.9.0
	v3.8.0	v2.8.0	v2.8.0		v3.8.0
	v3.7.1	v2.7.0	v2.7.0		
	v3.7.0	v2.7.0	v2.7.0		
	v3.6.0	v2.6.0	v2.6.0		
	v3.5.0	v2.5.1	v2.5.1		
	v3.4.1	v2.5.0	v2.5.0		
	v3.4.0	v2.4.0	v2.4.0		

Continued on next page

Table 1 – continued from previous page

SW Suite version	Dataflow Compiler version	HailoRT version	Firmware version	Model Zoo version	TAPPAS version
	v3.3.0	v2.3.0	v2.3.0		
	v3.2.0	v2.2.0	v2.2.0		
	v3.1.0	v2.1.0	v2.1.0		
	v3.0.0	v2.0.0	v2.0.0		

5. Working with Dockers

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 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 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. Dockers 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_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. 2023-07.1 Suite

This section is relevant for 2023-07.1 Suite, and its released packages:

- Dataflow Compiler v3.24.0
- Hailo Model Zoo v2.8.0
- HailoRT v4.14.0
- TAPPAS v3.25.0

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

- No known issues for now

6.1.4. TAPPAS

- Detection example application is currently not supported on i.MX6

