A.Y. 2025 / 2026

MULTIDISCIPLINARY PROJECT

Documenting neural-network deployment on STM32N6 NPU

AUTHORS



Patrizio Acquadro
patrizio.acquadro@mail.polimi.it



Giacomo Colosio
giacomo.colosio@mail.polimi.it



Tito N. Drugman
titonicola.drugman@mail.polimi.it



Sebastiano Colosio
sebastiano.colosio@mail.polimi.it

SUPERVISORS

Prof. C. Silvano
Dr. M. Ronzani

POLITECNICO DI MILANO
STMICROELECTRONICS

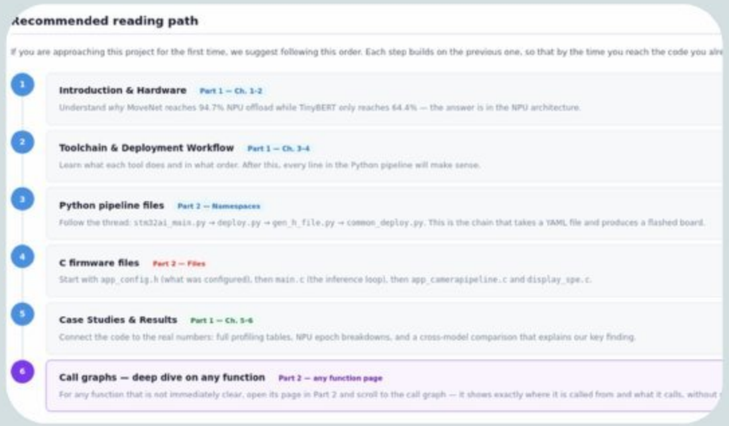
PROJECT THESIS

Code that runs is half the work. The other half is making it readable.

I · RECAP



II · NARRATIVE



III · CODE & LAYERS



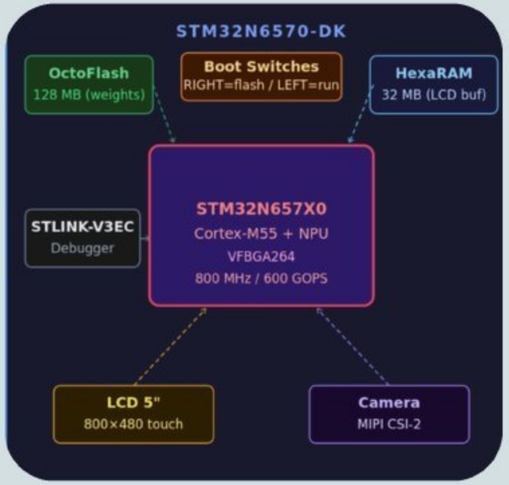
STM32N6 Edge AI Deployment

A complete embedded AI platform to deploy neural networks directly on the board



Board

STM32N6570-DK



Compute

Cortex-M55, Neural-ART NPU



I / O

On-board camera, LCD



Toolchain

ModelZoo, Edge AI Core, CubeIDE

Three Architectures, One Board

From an NPU-friendly CNN baseline to architectures with increasingly less compatible operators

MODEL	ARCHITECTURE	QUANT.	NPU OFFLOAD	LATENCY	VERDICT
MoveNet Lightning Single-person pose estimation	CNN — MobileNetV2 + FPN	INT8 / pc	94.7 %	22 ms	NPU-FRIENDLY
YOLOv8n-pose Multi-person detection + pose	CNN + detection head	INT8	87.9 %	32 ms	MIXED
TinyBERT NLP classification	Transformer — attention	INT8	64.4 %	> 100 ms	OP-BOUND

THE FINDING

Performance depended on **operator compatibility**, not model size.

From Deployment to Documentation

The project worked, but the knowledge was scattered across tools, code and implicit decisions

THE PROBLEM

- **Fragmented sources:** ST notes; Edge AI, CubeIDE and ModelZoo docs
- **Tooling drift:** versions; commands, generated artifacts
- **Tacit knowledge:** flags, buffers, memory choices
- **Two execution domains:** Python host pipeline + C firmware on target

OUR ANSWER

A self-contained site from model selection to on-board deployment

100+	23	3
PAGES OF HTML	ANNOTATED SOURCE FILES	READING MODES

The Documentation in Three Parts

PART I

01 | Tutorial

Read the full arc: from edge computing basics to profiling results.

6 CHAPTERS · ≈ 80 PAGES · FOR SUPERVISORS

PART II

02 | Code Reference

Follow the execution flow: host Python first, then target C after flashing.

23 FILES · 13 PYTHON · 10 C / H · FOR DEVELOPERS

PART III

03 | Module Groups

Locate each file by runtime role: host pipeline or target firmware.

2 GROUPS · HOST VS TARGET · FOR MAINTAINERS

01 - Tutorial

Chapter 1: From Edge to NN Deployment

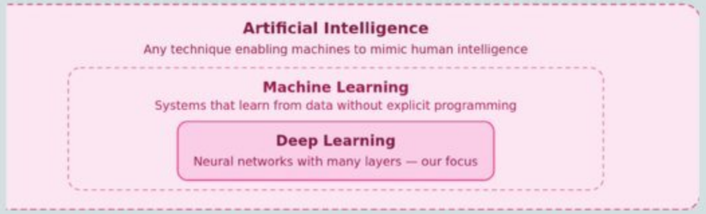
Part 1

From Edge to the Board



Part 2

From AI to NN



Part 3

The Intersection



Part 4

Putting All Together: **Our Goal**

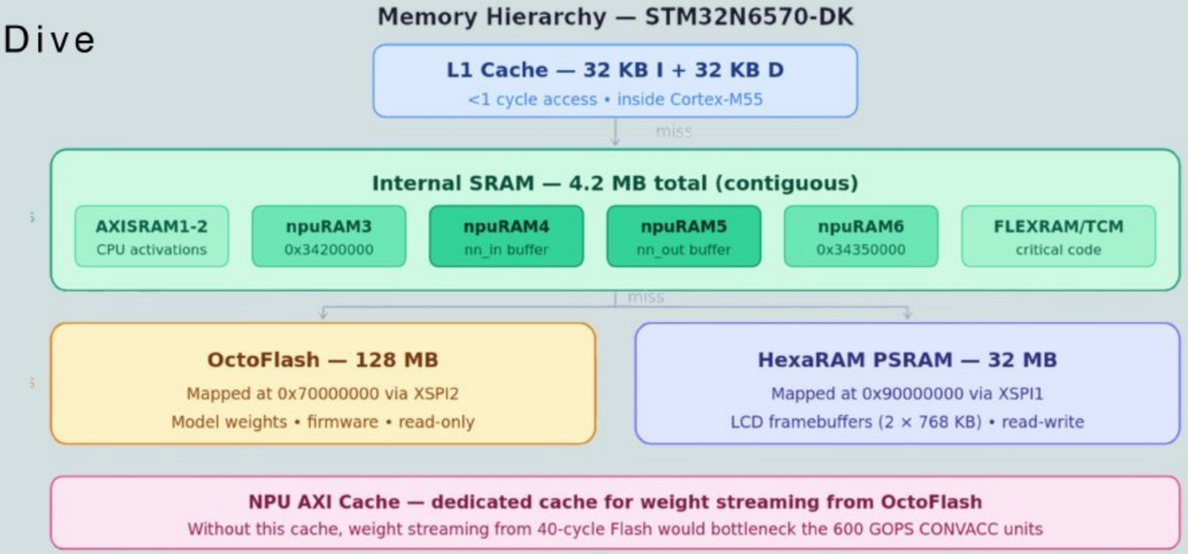
Benchmarking: Test MoveNet, YOLOv8n-pose, and TinyBERT on the STM32N6570-DK NPU.

Validation: Convert, flash, and run the models on-board with live input.

01 - Tutorial

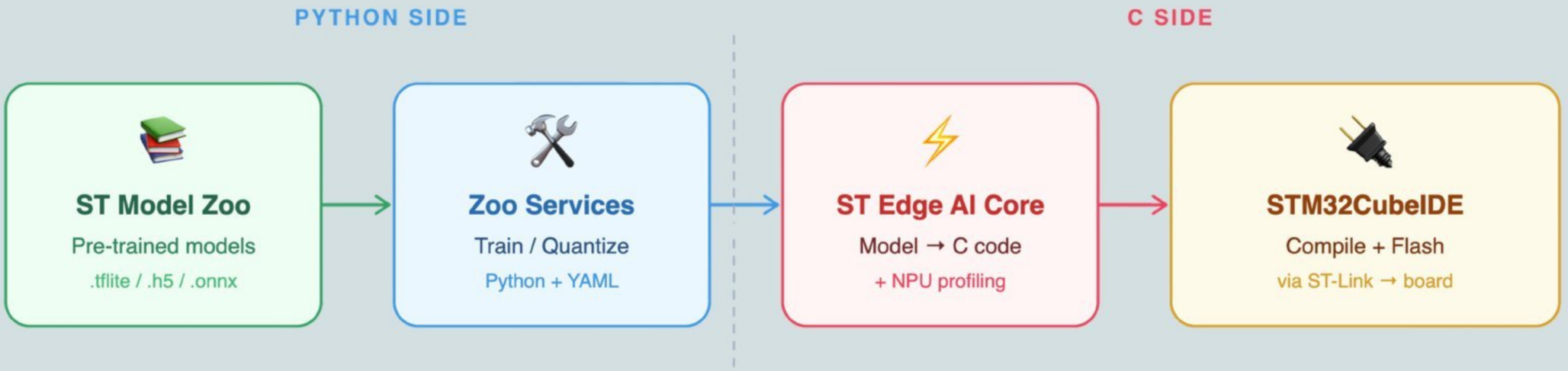
Chapter 2: Hardware Discovery Kit

2.0: The STM32N6570-DK Discovery Kit
2.1: Board Components — What Is on the Board and Why
2.2: The B-CAMS-IMX Camera Module
2.3: The STM32N657X0H3Q Microcontroller
2.4: Memory Architecture
2.5: The Neural-ART NPU — Architecture Deep Dive
2.6: The DCMIPP — Dual-Pipe Camera Pipeline
2.7: NPU Offload — What the Numbers Mean



01 - Tutorial

Chapter 3: The Software Toolchain



01 - Tutorial

3.1 ST ModelZoo: Where the Models Come From

```
stm32ai-modelzoo/
├── arc_fault_detection/
├── audio_event_detection/
├── depth_estimation/
├── face_detection/
├── hand_posture/
├── human_activity_recognition/
├── image_classification/
├── instance_segmentation/
├── neural_style_transfer/
├── object_detection/
├── pose_estimation/     ← used in this project
│   ├── handlandmarks/
│   ├── headlandmarks/
│   ├── movenet/         ← MoveNet Lightning lives here
│   │   ├── ST_pretrainedmodel_public_dataset/
│   │   ├── ST_pretrainedmodel_custom_dataset/
│   │   └── Public_pretrainedmodel_custom_dataset/
│   ├── yolov8n_pose/
│   └── yolov11n_pose/
├── re_identification/
├── semantic_segmentation/
└── speech_enhancement/
```

Subfolder	Description
ST_pretrainedmodel_public_dataset	Models trained by ST on public datasets (e.g. COCO). Best validated option.
ST_pretrainedmodel_custom_dataset	Models trained by ST on custom / proprietary datasets.
Public_pretrainedmodel_public_dataset	Public models (e.g. Google MoveNet) on public datasets.

01 - Tutorial

3.2 ModelZoo Services : Python Pipeline

MAIN FLOW

- **stm32ai_main.py:** Entry point and mode dispatch
- **parse_config.py:** YAML validation
- **deploy.py:** Deployment orchestration
- **gen_h_file.py:** Python-to-C bridge
- **quantize.py:** INT8 model quantization

SUPPORTING MODULES

- **preprocessing/:** Data loading and input preparation
- **postprocessing/:** Decoders and NMS
- **evaluation/:** Metrics and MLflow logging

KEY TAKEAWAY

The Python pipeline validates the configuration, prepares the model, generates C firmware constants, supports evaluation.

01 - Tutorial

3.3 ST EDGE AI: Bridge between Python and C

1. Graph optimization

Fuses adjacent operations (Conv + BatchNorm + ReLU into a single kernel), removes redundant transposes, and reformats tensor layouts for the target hardware.

3. Memory allocation

Places weights in OctoFlash and assigns activation buffers to the correct SRAM banks (npuRAM4, npuRAM5) based on the `.mpool` memory configuration file for the STM32N6570-DK.

2. NPU epoch assignment

Analyses every layer and assigns it to either an **EC epoch** (executed on the Neural-ART NPU hardware) or a **SW epoch** (executed on the Cortex-M55 CPU) based on NPU hardware support.

4. C code generation

Produces `network.c` (layer execution schedule), `network_data_params.c` (weight arrays in flash), and all supporting headers — ready to compile with STM32CubeIDE.

C source files (compiled into firmware)

```
generated/
├── network.c              ← epoch schedule
├── network_data.c         ← weight arrays
├── network_data_params.c
├── Inc/network.h
├── Inc/network_data.h
├── Inc/layers.h
├── Inc/ai_lite.h
├── Inc/ai_lite_interface.h
└── Inc/core_datatypes.h
    (+ 16 more header files)
```

Analysis reports (for documentation)

```
output_dir/
├── network_generate_report.txt
│   ← layers + epochs + memory
├── network_c_info.json
│   ← per-node cycle estimates
├── stm32ai_main.log
├── *_OE_3_2_0.onnx    ← optimised graph
└── *_Q.json           ← quantization params
```

01 - Tutorial

3.4 STM32CubeIDE: Compile, Sign, Flash

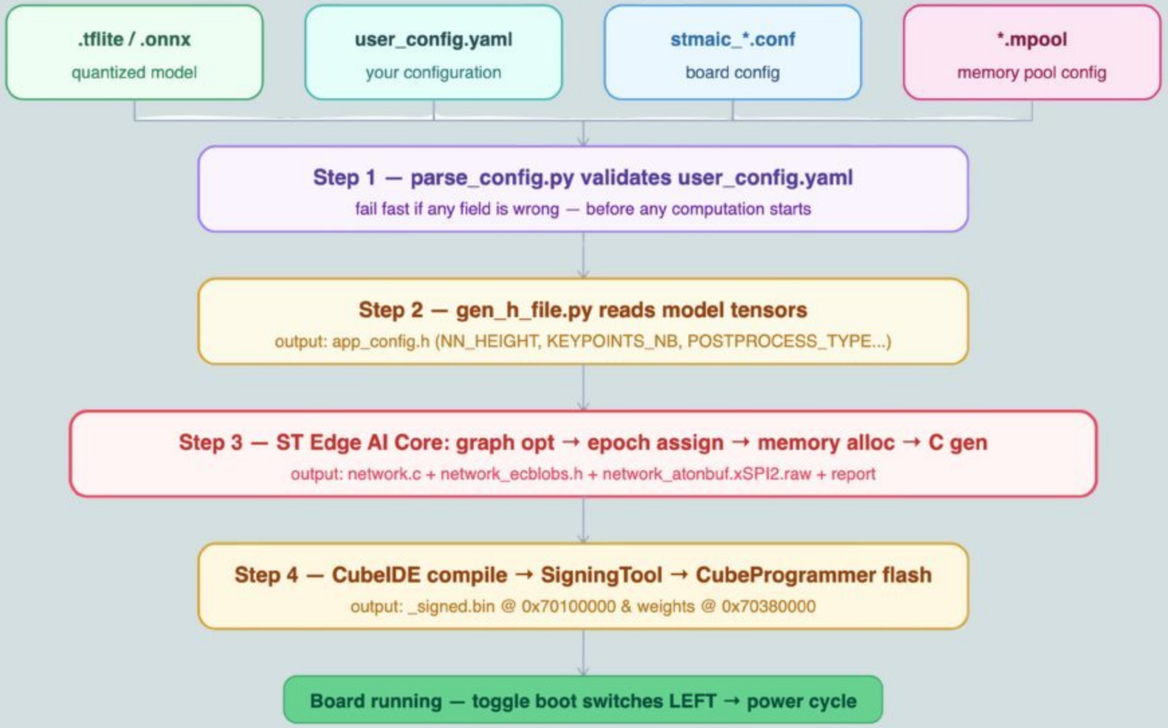
- **Inject files into CubeIDE**
  - Copies generated network files
  - Adds model weights and app_config.h
- **Build firmware**
  - Runs CubeIDE headless build
  - Generates the firmware .bin file
- **Sign binary**
  - Signs with STM32SigningTool
  - Required for STM32N6 secure boot
- **Sign binary**
  - Flashes the FSBL bootloader
  - Flashes signed firmware at 0x70100000
  - Flashes model weights at 0x70380000

```
{
  "description": "STM32N6570-DK Getting Started Pose Estimation",
  "builder": "stm32_cube_ide",
  "env": {
    "cproject_name": "STM32N6570-DK_GettingStarted_PoseEstimation",
    "project_folder": "${app_src_root}/Application/STM32N6570-DK/STM32CubeIDE",
    "network_src_root": "${ProjectFolder}/Model/STM32N6570-DK",
    "stm32_ai_lib_folder": "${ProjectFolder}/Middlewares/AI_Runtime"
  },
  "templates": [
    /* Files copied into the CubeIDE project before build: */
    [ "", "${network_src_root}/network.c", "copy" ],
    [ "", "${network_src_root}/network_ecblobs.h", "copy" ],
    [ "", "${network_src_root}/network_atonbuf.xSPI2.raw", "copy" ],
    [ "", "${stm32_ai_lib_folder}/Lib/GCC/ARMCortexM55", "copy-dir" ],
    [ "", "${app_src_root}/.../Inc/app_config.h", "copy" ]
  ]
}
```

01 - Tutorial

Chapter 4: Deployment Workflow

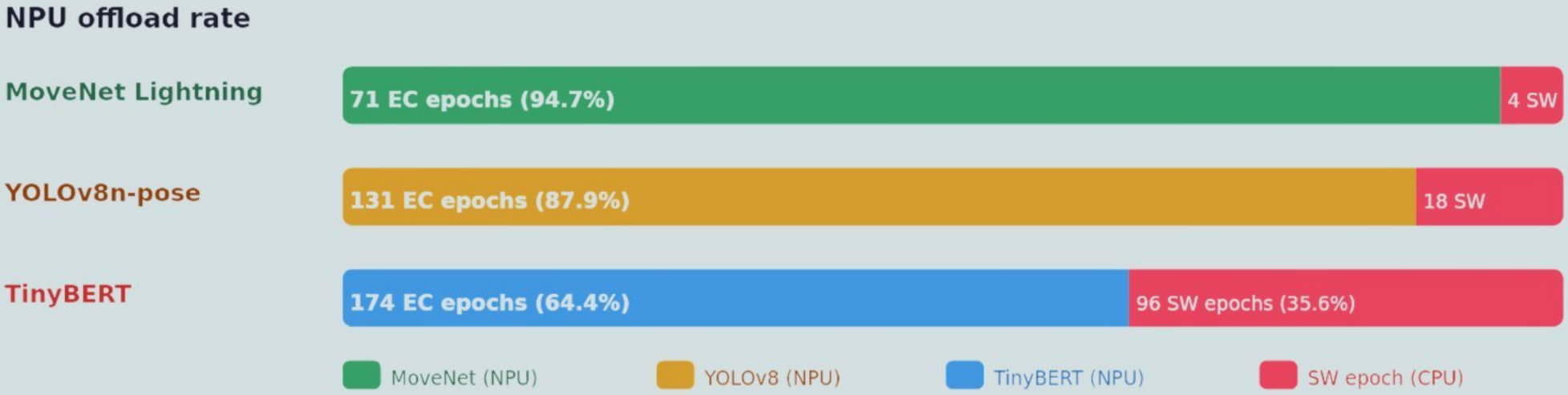
From Model File to Running Board



01 - Tutorial

Chapter 5: Case Studies

Three Models, One Board



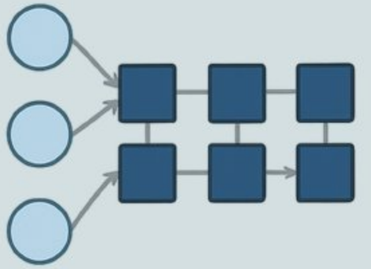
01 - Tutorial

Chapter 6: Results & Analysis

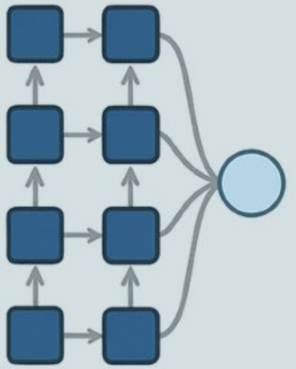
What the numbers tell us

CNN Operations — NPU-Compatible
Conv2D / DepthwiseConv2D: handled efficiently by CONVACC
ReLU / ReLU6: fused with Conv in hardware
BatchNorm: executed as scale-and-shift operations
Residual Add: native element-wise buffer addition
MaxPool / AvgPool: supported by the POOL unit

Transformer Operations — NPU-Incompatible
Softmax: requires exp() and sequence normalization
LayerNorm: needs mean, variance, and multi-step arithmetic
MatMul $Q \cdot K^T$: general matrix product, not Conv-optimized
Split / Transpose: creates non-sequential memory access patterns



CNN



Transformer

02 - Code Reference

Host Pipeline vs. Target Firmware

PYTHON PIPELINE

stm32ai_main.py — Entry point
parse_config.py — YAML validation
deploy.py — Deployment orchestrator
gen_h_file.py — Generates app_config.h
common_deploy.py — ST Edge AI + CubeIDE runner
external_memory_mgt.py — External memory patch
quantize.py — INT8 post-training quantization
preprocessing/ — Data loading and resizing
postprocessing/ — Decoders and NMS
evaluation/ — Metrics and MLflow logs
utils/ — Model and runner utilities
training / models / data_augmentation — Not used

C FIRMWARE

Src/main.c — Entry point and inference loop
Inc/app_config.h — Generated Python-to-C config
Src/app_camerapipeline.c — Camera setup
LL_ATON_RT_Main() — Auto-gen NPU runtime
Src/display_spe.c — MoveNet postprocessing
Src/display_mpe.c — YOLOv8 postprocessing
display_keypoints_13.h — MoveNet layout
display_keypoints_17.h — YOLOv8 COCO layout
crop_img.c / .h — Image crop utilities
main.h — Peripheral handles and typedefs
app_camerapipeline.h — Camera pipeline API
utils.h — Shared helper utilities

02 - Code Reference

Host Pipeline: From YAML to Flashed Board

THE HOST ENTRY POINT

- stm32ai_main.py, decorated @hydra.main
- No command-line args → everything lives in user_config.yaml
- Operation modes:
  - deployment
  - chain_qd (Quantize → Deploy)
  - training
- Execution chain: main(cfg) → get_config(cfg) → deploy(cfg)

Runs entirely on the host PC - the board is not touched yet

```yaml
# user_config.yaml
general:
  model_path:
st_movenet_lightning_heatmaps_192_int8.tflite

preprocessing:
  resizing:
    height: 192
    width:  192

deployment:
  hardware_setup:
    serie: STM32N6
    board: STM32N6570-DK
  conf_threshold: 0.4
  nb_keypoints: 13
```

02 - Code Reference

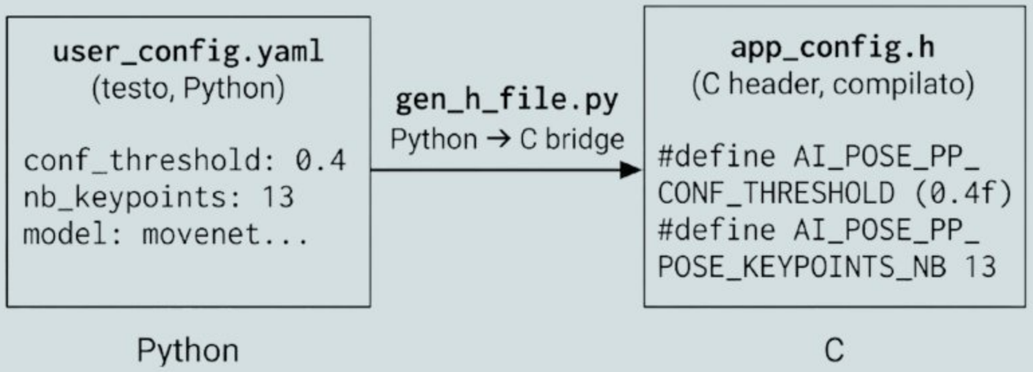
YAML-to-C Bridge

PARSE_CONFIG.PY

- Validates every section of user_config.yaml using Hydra + OmegaConf
- Checks required fields, sets defaults for optional ones
- Raises errors pointing to the exact invalid field
- Returns a shared config object used by all downstream functions

GEN_H_FILE.PY (PYTHON → C BRIDGE)

- Opens the quantized TFLite model with the TFLite interpreter
- Reads tensor shapes directly from the model
- Writes app_config.h with:
- This is the only Python-generated file the C firmware reads directly

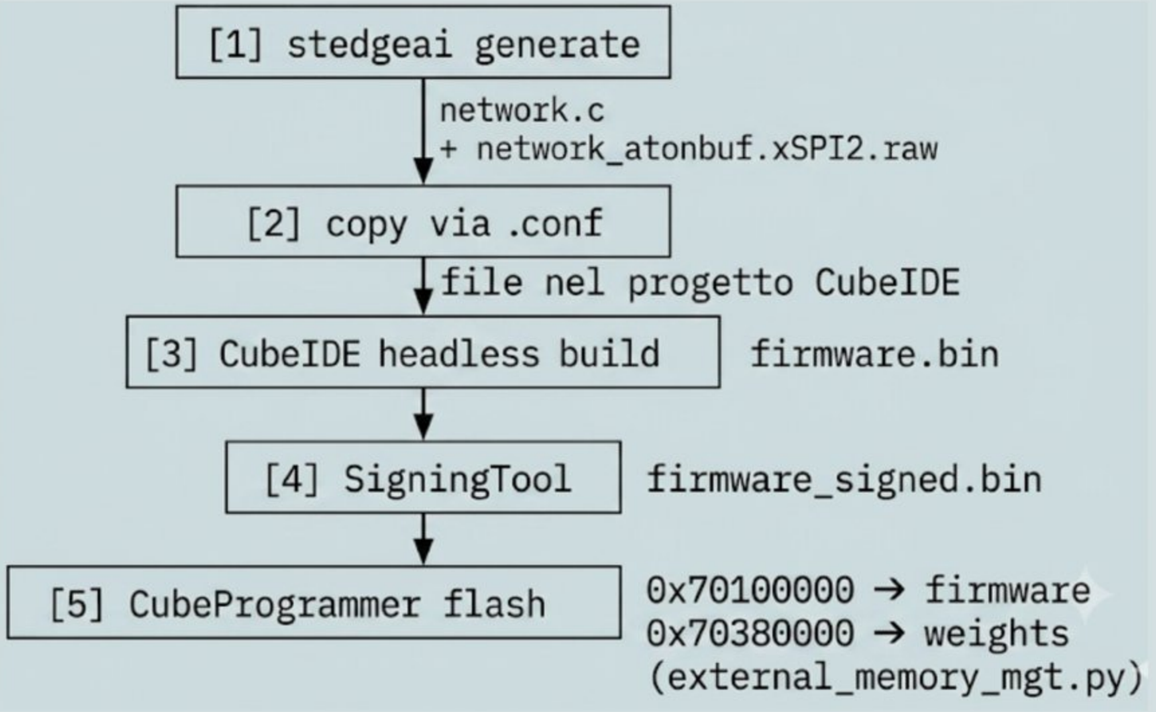


02 - Code Reference

Deploying on the board

deploy.py, common_deploy.py, external_memory_mgt.py

- **deploy.py (Deployment orchestrator)**
  - Calls gen_h_user_file_n6() to generate app_config.h
  - Selects the board .conf file for the specific model
  - Calls stm32ai_deploy_stm32n6() in common_deploy.py
- **common_deploy.py (ST tools invoker)**
  - Calls ST Edge AI Core CLI to compile the model
  - Invokes STM32CubeIDE headless build
  - Runs SigningTool for secure boot signature, flashes with CubeProgrammer
- **external_memory_mgt.py (Linker script patcher)**
  - Patches STM32N657xx.ld to place network_atonbuf.xSPI2.raw at 0x70380000
  - Without this patch: firmware crashes at boot - wrong memory address for weights



02 - Code Reference

Supporting Modules - Quantize, Load, Postprocess, Evaluate

QUANTIZE.PY (INT8 PTQ)

- Converts float32 model to INT8
- .h5 → TFLite Converter · .onnx → ONNX quantizer
- Calibration via representative dataset
- Output: .tflite INT8 or .onnx INT8

DATA_LOADER.PY + PREPROCESS.PY

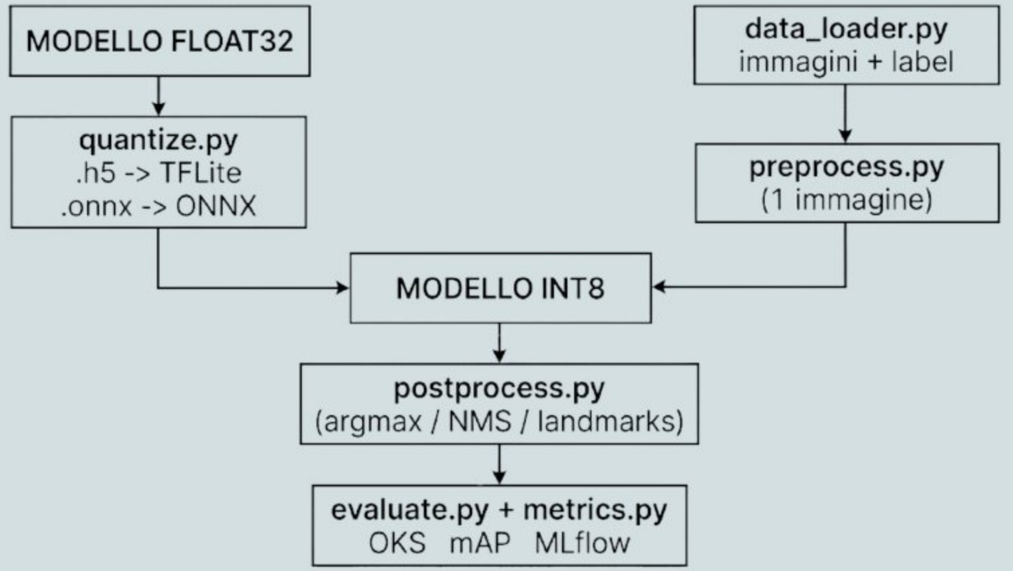
- Loads images + labels, applies resize / crop / normalize
- preprocess.py: single-image preprocessing for evaluation runs

POSTPROCESS.PY

- MoveNet: argmax on 48×48 heatmaps → keypoint (x, y)
- YOLOv8: NMS on bounding boxes + keypoints
- Hand / head landmark decoders

EVALUATE.PY + METRICS.PY

- OKS for MoveNet and mAP@[0.5:0.95] for YOLOv8
- Logs all metrics to MLflow



02 - Code Reference

The Host Pipeline: From Text to Quantized Mode

APP_CONFIG.H (GENERATED BY PYTHON)

- NN_HEIGHT = 192
- NN_WIDTH = 192
- AI_POSE_PP_POSE_KEYPOINTS_NB = 13
- POSTPROCESS_TYPE = POSTPROCESS_SPE_MOVENET_UF
- AI_POSE_PP_CONF_THRESHOLD = 0.4

Every other C file reads from app_config.h
It is the only generated file in the firmware

```
// Initialization (once)
HAL_Init();
SystemClock_Config();
MX_DCMIPP_Init();   // camera hardware
// LCD setup

// Main loop (forever)
while (1) {
    wait for camera snapshot → npuRAM4
    LL_ATON_RT_Main(&network);  // NPU
    postprocess();              // decode
    render skeleton on LCD
}
```

02 - Code Reference

Zero CPU in the Memory Path: app_camerapipeline.c

CAMERA → DCMIPP → PSRAM LCD FRAMEBUFFER [0X90000000]

- Runs continuously and autonomously
- LCD always shows live full-resolution video
- CPU is not involved

CAMERA → DCMIPP → NPURAM4 [0X34270000]

- Triggered on demand
- Delivers a cropped, resized snapshot ready for NPU input
- CPU involvement: zero — DMA handles the transfer

Both pipes run in hardware simultaneously.
The CPU is free for postprocessing while the camera feeds both outputs.

02 - Code Reference

Zero CPU in the Memory Path: app_camerapipeline.c

npuRAM4 [0x34270000] → NPU (71 EC + 4 SW epochs) → npuRAM5 [0x342E0000]
192×192×3 camera frame — 48×48×13 heatmaps (float32, one per keypoint)

TYPE	COUNT	WHAT HAPPENS
EC epochs Hardware	71	EC blob loaded into NPU hardware registers, NPU executes
SW epochs Software	4	ll_sw_* functions run on CPU for unsupported ops
Total	75	

- Not a file you write - injected by common_deploy.py as part of the ll_aton runtime library
- 71 out of 75 epochs on NPU → 94.7% NPU offload for MoveNet

02 - Code Reference

Decoding the Output: display_spe.c VS display_mpe.c

DISPLAY_SPE.C (MOVENET DECODER)

- Reads float32 heatmaps from npuRAM5
- For each of 13 channels: finds argmax over 48×48 grid → keypoint (x, y)
- Filters by confidence threshold (from app_config.h, default = 0.4)
- Draws skeleton lines on LCD foreground layer using connectivity from display_keypoints_13.h

DISPLAY_MPE.C (YOLOV8 DECODER)

- Reads YOLOv8 detection output: bounding boxes + 17 keypoints per person
- Applies NMS filtering by confidence and IoU thresholds
- Draws bounding boxes + 17-keypoint COCO skeletons on LCD
- Handles multiple persons simultaneously

Thank you

Questions?

LIVE DOCUMENTATION

patrizioacquadro.github.io/stm32n6-docs/