Embedded and Frugal AI (C5)

Course plan — SETI Master, academic year 2026–2027

Sessions: eight Wednesdays, 08:45–12:30 (3h45), room 3B01
Exam: Wednesday 10 March 2027, 08:30–10:30, room 3B01

What the course is about

The course teaches how to make a neural network fit an embedded platform and how to *prove* that it does. The labs follow one thread on the AMD Kria KV260: train a small model, compress it (quantization, pruning, distillation), compile it for the DPU, run it on the board, and measure accuracy, latency, throughput and energy in the same run. The last labs open the accelerator box: one convolution layer written in Vitis HLS, then a whole MNIST CNN as one streaming accelerator on the board.

The goal is an engineering habit rather than tool knowledge: *given a budget (accuracy, latency, memory, power), choose an optimisation, predict its effect, measure it, and report what it cost.*

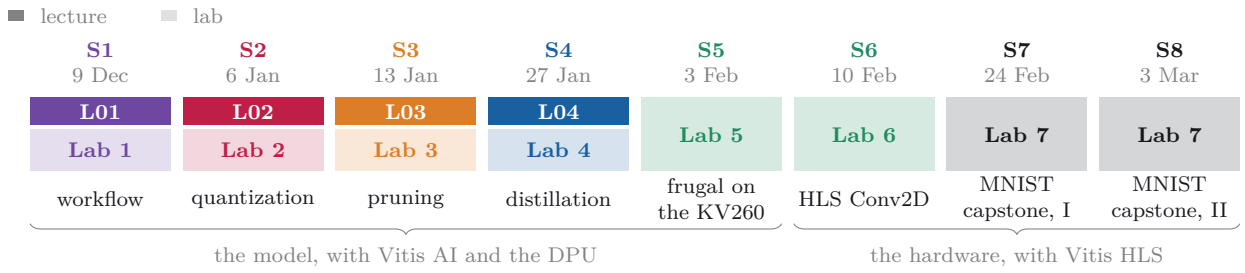
Learning outcomes

At the end of the course, you can:

1. compute parameters, MACs, activation footprint and model size of a CNN, and relate them to memory traffic and latency;
2. run reproducible experiments (one tag, one run folder, validation for decisions, test once) and report accuracy together with its cost;
3. apply post-training quantization and quantization-aware training with Vitis AI, and diagnose deployment failures (unsupported operators, export batch size, DPU fingerprint);
4. compare pruning policies (unstructured versus structured, criteria, allocation), let a search choose the layer widths (sensitivity analysis, one-step search, once-for-all NAS), and explain why sparsity alone does not speed up a dense accelerator;
5. train a compact student network with knowledge distillation and evaluate it against controlled baselines;
6. deploy INT8 models on the KV260 with XIR/VART, and separate DPU latency, pre-processing, application FPS and energy per inference;
7. design a streaming Conv2D in Vitis HLS (line buffers, unrolling, output-channel parallelism, fusion) and read synthesis reports;
8. take a trained CNN to one fused accelerator, bit-exact with its INT8 model, choose the parallelism of each layer, and run it behind an AXI DMA on the KV260.

Calendar

Sessions 1–4 start with a lecture, followed by its lab; sessions 5–8 are labs, and their handouts are the reference material. Follow each handout's guided sequence and student TODO contracts.



Session	Lecture	Lab (handout)
S1 Wed 9 Dec 2026	**L01 Introduction**: workflow, metrics, edge AI, accelerators, FPGAs	**Lab 1** Vitis AI workflow: Apptainer, conda, tagged runs, MLP vs CNN on FashionMNIST, first PTQ XMODEL
S2 Wed 6 Jan 2027	**L02 Quantization**: scale, rounding and clipping, calibration, PTQ, fast fine-tuning, QAT	**Lab 2** PTQ vs QAT on CIFAR-10, compilation for the KV260 DPU, first XIR/VART run, board accuracy
S3 Wed 13 Jan 2027	**L03 Pruning and Architecture Search**: pruning policies, fine-tuning, OFA/NAS and MAC budgets	**Lab 3** two experiments: pruning policies on a small CIFAR-10 CNN, fine-tuned, deployed and profiled; VGG6-BN automatic pruning and OFA using the course container and supplied baseline/supernet; final task: one policy end to end
S4 Wed 27 Jan 2027	**L04 Knowledge Distillation**: soft targets, temperature, response and attention transfer	**Lab 4** CE vs KD vs attention-transfer students on MNIST, controlled comparison, KV260 deployment; your own KD
S5 Wed 3 Feb 2027	—	**Lab 5** standard vs frugal (physically pruned + KD + INT8) rock–paper–scissors classifier on the KV260, live USB camera
S6 Wed 10 Feb 2027	—	**Lab 6** Conv2D versions 00–04 in Vitis HLS, `OC_PAR` design-space exploration, fused Conv–ReLU–MaxPool (Stage 05); optional: DATAFLOW, IP export, Vivado, PYNQ
S7 Wed 24 Feb 2027	—	**Lab 7, part 1** MNIST capstone: compact Conv–Pool–FC (6 810 parameters, 31 096 MACs), INT8 export with bit-exact vectors; implement FC and DATAFLOW top, C simulation, first synthesis
S8 Wed 3 Mar 2027	—	**Lab 7, part 2** parallelism per layer against resources and interval; the accelerator on the KV260 (correctness, FPS, energy)
Exam Wed 10 Mar 2027	Written exam, 08:30–10:30	

How the labs connect

- **One experiment, one tag.** Every lab writes its results under `out/runs/<TAG>/` (Labs 1–4) or `work/` (Lab 5); reports are rebuilt from these folders, never from memory.
- **Decide on validation, report on test.** Labs 1, 3, 4 and 5 hold out a validation split; Lab 2 compares quantization methods on the test split, which its handout states.

- **Same pipeline every time.** Float model → compression → Vitis AI quantization (batch-1 XMODEL export) → `vai_c_xir` with the board's `arch.json` → XIR/VART on the KV260.
- **Accuracy is never reported alone.** Parameters, MACs, XMODEL size, DPU latency, application FPS and, from Lab 2 on, energy per inference accompany accuracy when the required runtime and sensors are available; otherwise mark them unverified.
- **From the DPU to your own hardware.** Lab 5 asks whether about 7× fewer dense MACs translate into measured DPU speedup; Labs 6 and 7 show where the cycles go by building the accelerator.

Environment

Every lab folder holds the same settings file (`course_env.sh`): the lab scripts find the container image, the datasets and the board through it.

Lab machines

- Vitis AI PyTorch CPU image (Apptainer), Vitis AI 3.5 tools: the labs use `$VAI_SIF` if set, then the planned IT-provided shared copy `/comelec/softs/share/vitis_ai/vitis-ai-pytorch-cpu.sif`, then a private copy in `/tmp/$USER` (pulled once; never in the home directory, whose quota is too small).
- Datasets (CIFAR-10, MNIST, FashionMNIST, rock–paper–scissors) are downloaded once for the whole course: keep the lab folders side by side, for example in `~/c5/`, and the lab scripts keep the datasets in `datasets/` next to them; a read-only copy on the lab machines, `/comelec/softs/share/c5_datasets`, is reused when available. A standalone lab falls back to the user cache; an explicit absolute `C5_DATA` overrides discovery.
- Labs 6 and 7: Vitis HLS and Vivado 2021.2 from `/comelec/softs/bin`; `tio` for the serial console of the boards.
- Each lab folder has `c5_clean.py` (in `scripts/`, `tools/` or `host/`): it lists what the lab produced and removes it with `--yes`; your code and the datasets stay.

KV260 boards

- Each board answers to a network name such as `maquette-07`, or to its IP address: use the name you are given in the lab, not the `KV260-xx` inventory sticker. Set it once per terminal with the account of the SD image: `BOARD=root@maquette-07` (PetaLinux) or `BOARD=ubuntu@maquette-07` (Ubuntu/PYNQ).
- Labs 2–5: PetaLinux/VART, account `root`. The supplied compiler fingerprint is `0x101000056010407`. Compare it with `xdputil query` on the actual board; host Vitis AI 3.5 compilation alone does not verify the historical Vitis AI 3.0 board runtime.
- Labs 6 and 7: Ubuntu 22.04 image with Kria-PYNQ, account `ubuntu`; your own bitstreams.

Course materials

- Lectures: `L01_introduction` with `L01_core`, `L02`–`L04` with supplementary slides, and `hls_cnn_guide` for Labs 6–7.
- Labs: `TP01`–`TP07`, one folder and archive per lab; reference sheet `guide_kv260_vitis_ai`.