딥러닝 기초와 FPGA 구현
-- Deep learning from basic and implementation on FPGA --

기안도

adki@future-ds.com

2022년6월30일 ~ 7월1일

1

강좌의 개요

이 강좌는 인공지능과 딥러닝에 대한 기초적인 내용부터 응용까지를 다루고,

딥러닝을 이해하고 응용을 개발하는데 필요한 다양한 이론적 배경과 개발 환경에 대해 상세하게 설명하며,

딥러닝 응용을 프로그램하여 실습하는 과정을 통해 직접 경험해 보고,

FPGA를 활용한 환경에서 딥러닝을 실습해 봄으로써 딥러닝 활용 기회를 확장해 볼 수 있도록 한다.

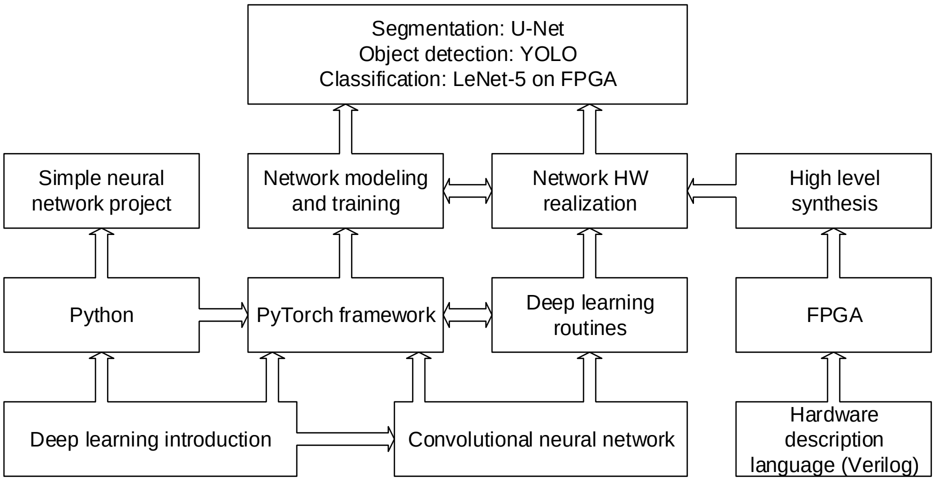
Copyright (c) Ando Ki

2

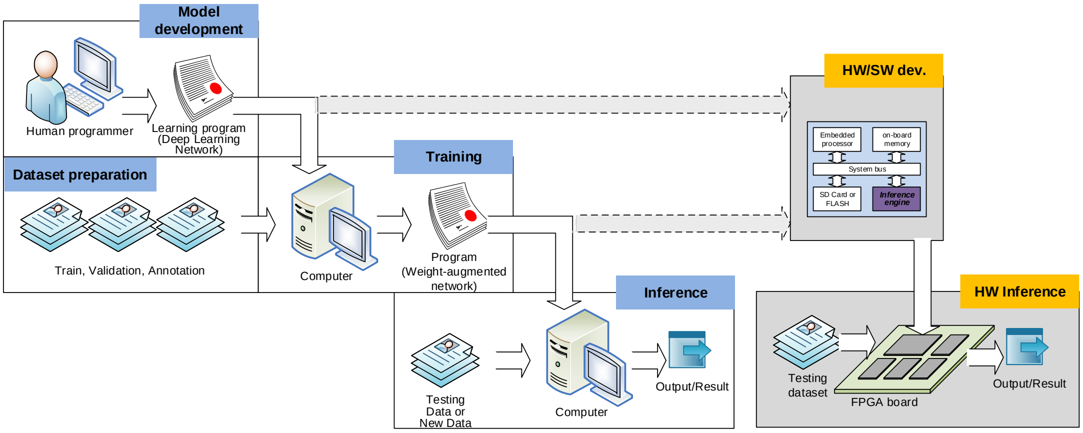
수강자와 선수지식

- 대상 수강자 (Target audience)
  - 이 강좌는 인공지는/딥러닝 응용을 개발하는 것에 관심이 있는 개발자와 학생을 대상으로 한다.
    - This lecture is prepared for engineers and students who are interested in developing deep-learning application.
- 선주지식 (Prerequisites)
  - Experience with industry standard Operating Systems such as Linux (Ubuntu)
  - Experience with text editor such as Vi/Vim/Gvim.
  - Basic knowledge of the C/C++ language
  - Experience with industry standard C++ compilation tool-chain; GNU GCC/G++
  - Basic knowledge of digital logic design and the Verilog hardware description language
  - Experience of FPGA and its development environment; Xilinx FPGA, Xilinx Vivado

주요내용

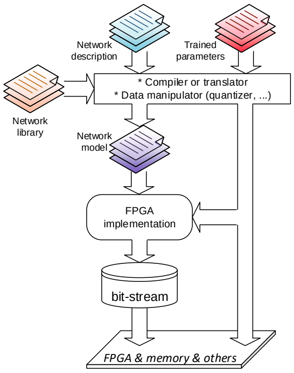

Segmentation: U-Net
Object detection: YOLO
Classification: LeNet-5 on FPGA
Simple neural network project
Network modeling and training
Network HW realization
High level synthesis
Python
PyTorch framework
Deep learning routines
FPGA
Deep learning introduction
Convolutional neural network
Hardware description language (Verilog)

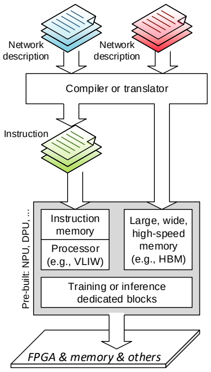
Design flow for deep learning network



Two approaches for HW-assisted

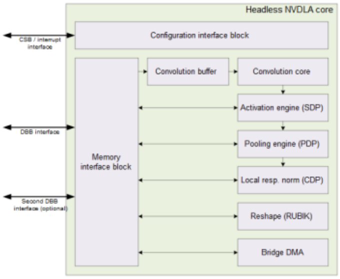
- Application specific
  - RTL generating
  - Utilize neural network IP
- Generic
  - Fixed structure that contains programmable or configurable neural network IP



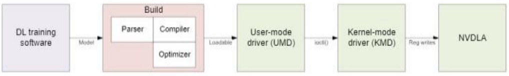


Open source DPU/TPU

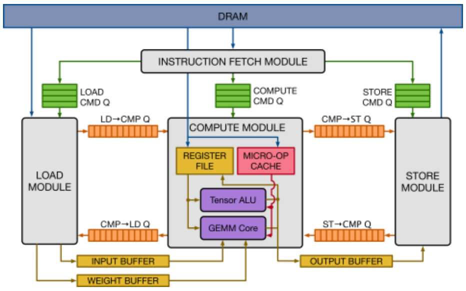
- Deep-Leaning Processing Unit style
  - NVDLA (NVIDIA Deep Learning Accelerator)



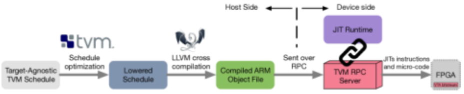
  - NVDLA SW 환경



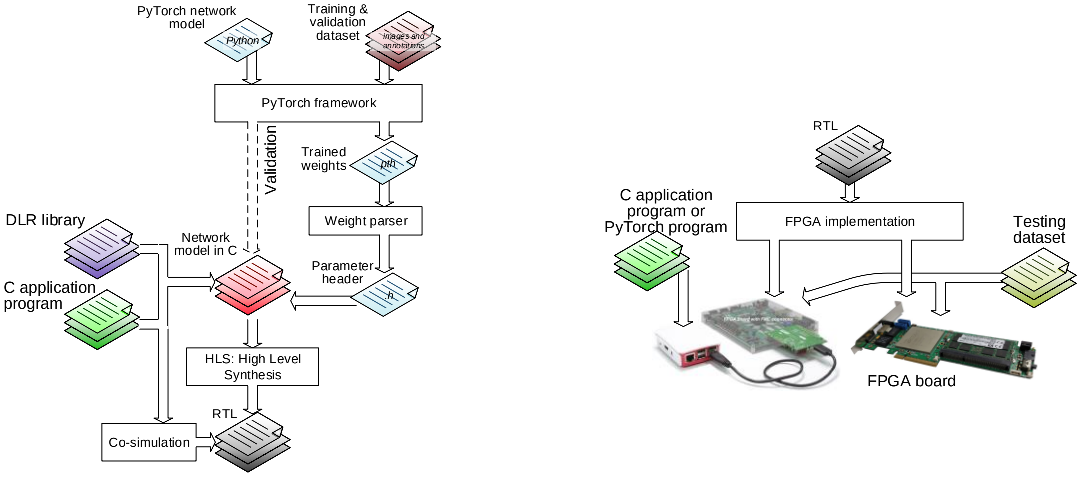
- Tensor Processing Unit style
  - VTA (Versatile Tensor Accelerator)



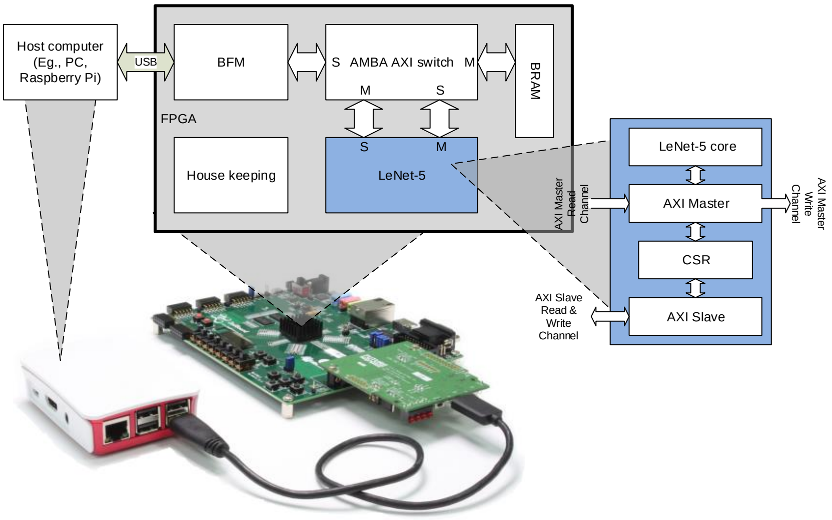
  - TVM: TVA compiler environment



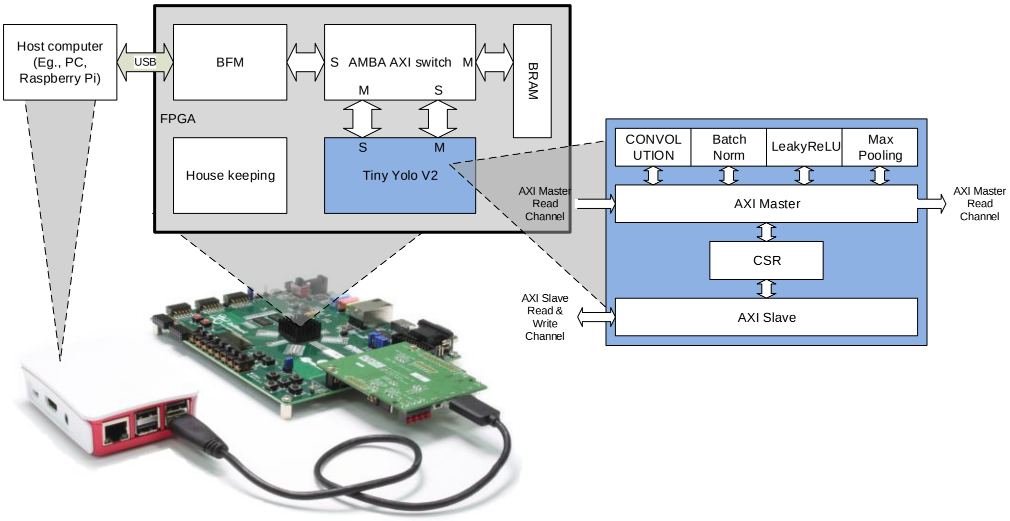
Design flow for a specific network for HW

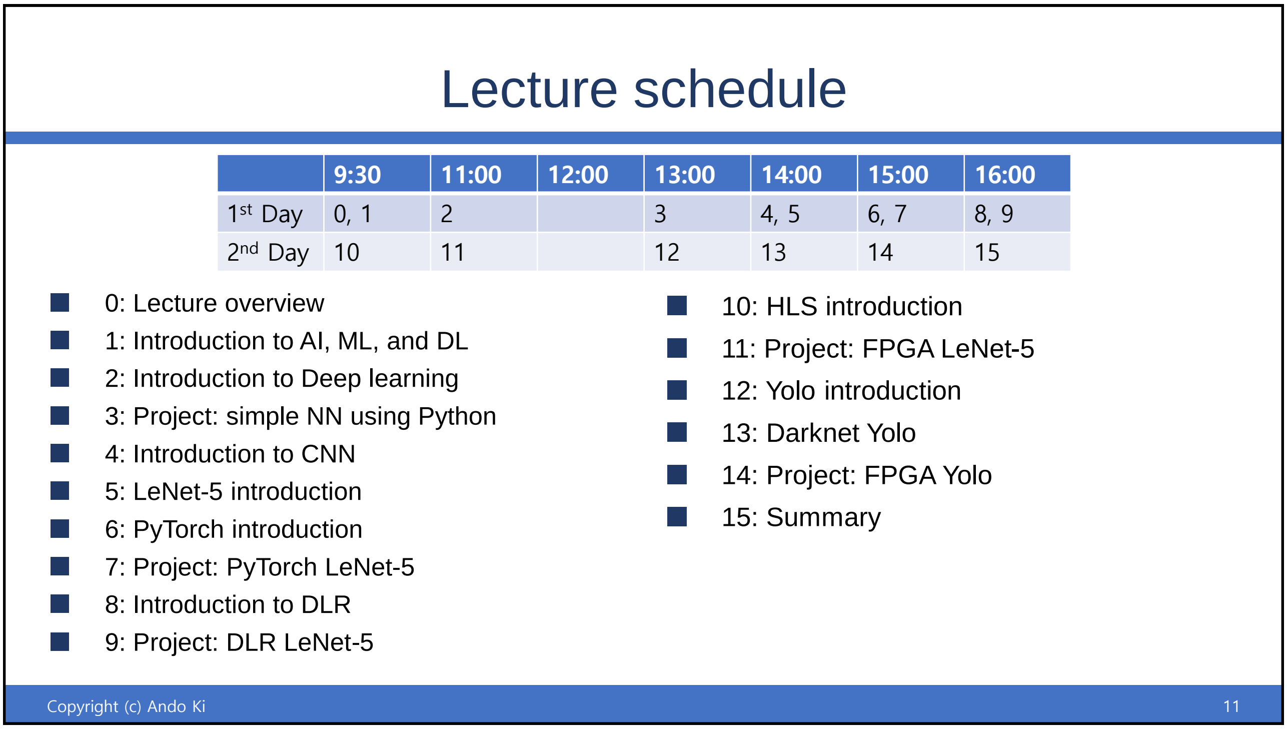


LeNet-5 example



Tiny Yolo V2 example



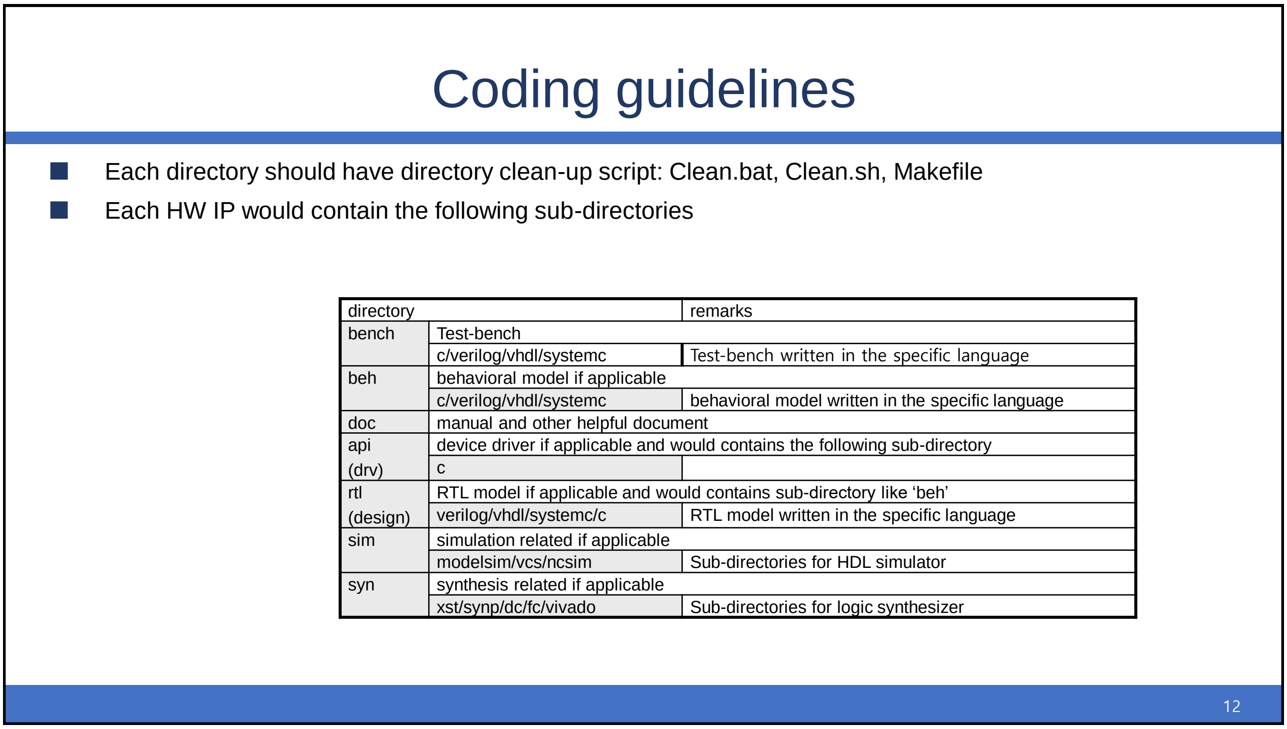


Lecture schedule

	9:30	11:00	12:00	13:00	14:00	15:00	16:00
1st Day	0, 1	2		3	4, 5	6, 7	8, 9
2nd Day	10	11		12	13	14	15

- 0: Lecture overview
- 1: Introduction to AI, ML, and DL
- 2: Introduction to Deep learning
- 3: Project: simple NN using Python
- 4: Introduction to CNN
- 5: LeNet-5 introduction
- 6: PyTorch introduction
- 7: Project: PyTorch LeNet-5
- 8: Introduction to DLR
- 9: Project: DLR LeNet-5
- 10: HLS introduction
- 11: Project: FPGA LeNet-5
- 12: Yolo introduction
- 13: Darknet Yolo
- 14: Project: FPGA Yolo
- 15: Summary

Copyright (c) Ando Ki



Coding guidelines

- Each directory should have directory clean-up script: Clean.bat, Clean.sh, Makefile
- Each HW IP would contain the following sub-directories

directory		remarks
bench	Test-bench	
	c/verilog/vhdl/systemc	Test-bench written in the specific language
beh	behavioral model if applicable	
	c/verilog/vhdl/systemc	behavioral model written in the specific language
doc	manual and other helpful document	
api (drv)	device driver if applicable and would contains the following sub-directory	
	c	
rtl (design)	RTL model if applicable and would contains sub-directory like ‘beh’	
	verilog/vhdl/systemc/c	RTL model written in the specific language
sim	simulation related if applicable	
	modelsim/vcs/ncsim	Sub-directories for HDL simulator
syn	synthesis related if applicable	
	xst/synp/dc/fc/vivado	Sub-directories for logic synthesizer