ECE479 LAB 3: IoT and Cognitive Computing Systems

- Demo: May 6th in lab
- Report Due: May 6th, 11:59 PM

In this last lab, you will design and implement an IoT or cognitive computing system for real applications. We have two different tracks for you to choose from. One will focus on designing an IoT system; the other will focus on accelerating a cognitive computing application. You can pick either **ONE** of the two tracks.

You will need to submit a project proposal, do a middle checkpoint demo, a final demo, and submit a final report. Before you start to work on this lab, please read through this whole document carefully.

Track 1: IoT Systems

In this track, you can propose, design, and implement a Raspberry Pi IoT system that solves some real-world problems.

The system you propose must be related to IoT topics we covered in the lectures. You can take advantage of the GPIO interface on the Raspberry Pi, the PiCamera, and the Coral stick that we provided in the kit. You may also purchase extra hardware if necessary, but please keep the budget under a reasonable amount. Your project will not be graded completely based on the hardware you purchase.

The examples of IoT systems include but are not limited to:

- Security cameras and video surveillance systems
- Smart home assistant including environment monitor, voice assistant, smart door bell, etc.
- Your own idea of applying IoT technologies into real-world problems

You may find ideas from online resources such as PiMyLifeUp, SeeedStudio, etc. Please make sure to cite your resource in your proposal and report when you find the project ideas online.

Track II: DNN Accelerator and DNN Acceleration Library

In this track, you can design the algorithm and accelerate on a specialized hardware including Coral TPU, FPGA, and GPU.

There are many good examples of hardware accelerators, libraries, and implementations of the state-of-the-art machine learning algorithms, and many of them are open-sourced. You can build your own design on top of those open source projects, but make sure that you build substantial components on your own.

Here are some DNN accelerators / libraries / models that can be good starting points for you:

- 2021 DAC System Design Contest (DAC-SDC) designs. GitHub repo: https://github.com/pjh177787/dacsdc21, and here's the SkyNet paper
- HiKonv. GitHub repo: https://github.com/pjh177787/HiKonv, and here's the HiKonv paper
- ARM Computer Vision and Machine Learning library. https://github.com/ARM-software/ComputeLibrary
- cuDNN and cuBLAS: https://developer.nvidia.com/cudnn https://developer.nvidia.com/cublas

In the above projects, except for the official libraries from ARM and NVIDIA, others are designed by us from the Chen's lab. DAC System Design Contest is a contest focused on low-power high-accuracy object detection and tracking. This repo includes SkyNet and UltraNet, which are hardware-efficient Deep Neural Network (DNN) for object detection and tracking, optimized for edge AI solutions and demonstrated in unmanned aerial vehicle (UAV) applications, winning first and second places in DAC-SDC. Hi-Konv is a tool to maximize the multiplier utilization for arbitrary bitwidth data in convolution operations. It is the key technology for us to win DAC-SDC 2021 as it was applied to the implementation of UltraNet.

You can use embeded CPU (Raspberry Pi), embeded FPGA (PYNQ), GPU (Google Colab), or AIE (AMD Ryzen laptops) to implement your design. You may ask the TAs for access to PYNQ FPGAs. Google Colab is a Python-Notebook environment very similar to the format of Jupyter notebook, which you have been working with so far. If you choose to work with FPGA or GPU, we highly recommend you to finish the two Mini-Labs first, as these are simple tutorials for you to get started with FPGA developments and Google Colab. If you choose to work with AIE, you need to have an AIE-enabled laptop that has AMD Ryzen CPUs. Details of each platform are listed below.

Please note that the difficulties of using different platforms are significantly different. Set a suitable goal for your chosen platform. Here are some potential project directions (the topics are not constrained to these) and the **minimum** requirement for each platform.

- **CPU**

  - *Projects*: You can choose open source DNN libraries for CPU (e.g. ARM Computer Vision and Machine Learning library), run the libraries on your Raspberry Pi, and try different techniques of optimizing inference latency on CPU. For example, you can explore the possibilities of different quantization schemes, weight pruning schemes, DNN layer fusions, etc. ARM Computer Vision and Machine Learning library mentioned above will be a good starting point.
  - *Minimum requirement*: Approach 1: Choose one published model, and reproduce the same accuracy result as the published models with TensorFlow (or other ML frameworks you like). Propose and implement techniques (e.g. quantization, layer fusion, etc) to shorten the computation latency as much as possible, without losing much accuracy (within 5%). Approach 2: Choose one accelerator library, such as ARM Machine Learning Library and HiKonv. Evaluate the improvements of this library over native baseline. Explore possible extensions to this library. Propose and implement techniques to shorten the computation latency. Demonstrate that your design is **working**.
- **FPGA**
  - *Projects*: You can choose a deep learning model that you are interested in, and implement the inference accelerator of the DNN with FPGA. Specifically, you may want to work with a Edge FPGA device with a light-weight neural network, as this class focuses on IoT systems. For example, you can implement MobileNet (or a modified version of it) with FPGA. Alternatively, you can choose an open source DNN accelerator for specific machine learning operations, map the accelerator onto FPGA, and optimize the design. If you need a FPGA to work with, you can borrow from us and return at the end of the semester.
  - *Minimum requirement*: Demonstrate that your proposed accelerator is **working** on FPGA for the target DNNs, and you have FPGA-specific optimization in your implementation (e.g. low-bitwidth number representations). The target DNNs should be more complicated than LeNet or the FashionNet we used in labs. If you target to accelerate one specific operation, such as convolution, transformer module, sparse matrix multiplication etc., you will need to propose reasonable benchmarks to demonstrate the improvements.
- **GPU**
  - *Projects*: You can choose a published model, implement the model in GPU, and try different techniques of optimizing inference latency on GPU. For example, you can explore the possibilities of different quantization schemes, weight pruning schemes, DNN layer fusions, etc. You may choose to use either Tensorflow or PyTorch on Google Colab.
  - *Minimum requirement*: Choose one published model, and reproduce the same accuracy result as the published models with TensorFlow (or other ML frameworks you like). Propose and implement techniques (e.g. quantization, layer fusion, etc) to shorten the inference latency as much as possible, without losing much accuracy (within 5%). Demonstrate that your design is **working**.

- **AIE**
  - *Projects*: AI Engine by AMD is a new AI accelerator. It has a similar hardware architecture as the TPU but provides more flexibility and programmability. If you have a laptop with a latest AMD Ryzen CPU, you may be able to explore the capability of AI Engine that is integrated on the CPU. To see if your laptop qualifies, you can check the list at the bottom of Ryzen AI website.
  - *Minimum requirement*: Walk through the tutorials on the Ryzen AI document. Design a **working** accelerators on AIE for a modern DNN of your choice. Demonstrate your design is working, optimized, using unique features offered by AIE, and potentially outperforms similar designs on other platforms.

You need to evaluate the performance of your design, and compare to the existing published designs (if any). The designs for different platforms (CPU/FPGA/GPU/AIE) will be evaluated separately. Please refer to the **NOTE** at the end of this document about grading designs on different platforms.

Computation Resources

- EWS machines: https://it.engineering.illinois.edu/ews/lab-information/remote-connections

The Engineering WorkStation (EWS) provides powerful workstations and servers, and it allows for the remote connection.

- Campus Cluster: https://campuscluster.illinois.edu/resources/docs/start/

The Illinois Campus Cluster provides access to computing and data storage resources. Users submit computation tasks to compute node via a *batch job*, which is submitted to a queue to be scheduled to run when the compute node resources requested are available.

- Google Colab: https://colab.research.google.com/notebooks/gpu.ipynb

Google provides free GPU usage through Colab. Your project will be developed in Python, and you can access your code on the Colab remotely.

If you need access to PYNQ, you can borrow one from us. You will need to return the PYNQ board at your final demo.

Demo and Report Requirements

Project-Proposal (March 31st, 11:59 PM), In-Lab Discussion (April 1st) (6 pts)

You need to submit a proposal (1 to 2 pages) in the first week of this lab. Here are the basic requirements for the proposal:

- State clearly which track you choose and which platform you will use. (1 pt)
- Briefly describe the system you are proposing, and how you will implement the system. Describe the differences between your system and the existing ones. (2 pt)
- List the key features you propose and the difficulties of your project. (2 pts)
- List the **deliverables** for the middle checkpoint demo and the final demo. (1 pt)

You need to come to the April 1st lab and discuss your proposal with your TA. TAs will check whether the workload of your proposed project is suitable and will give you feedback.

Middle Checkpoint Demo (April 15th) (8 pts)

During this demo, you need to show your TAs the progress you have made and TAs will check whether you are making good progress as planned in your proposal.

- Report your progress to your TA. Explain clearly what has been done and how you achieved that. (3 pts)
- Demo the middle checkpoint deliverables as planned in your proposal. (3 pts)
- Show the remaining TODOs to your TA and explain what's your plan to complete those TODOs. (2 pt)

In-class Presentation (April 18-25th) (6 pts)

You will need to prepare for a short 5-minute presentation about your project. It is an opportunity for you to showcase your exciting designs to the professors, TAs, and your classmates. More details will be anounced when we get closer to the presentaion, including signing up for a time slot.

Your presentation should roughly consist of four slides:

- Title slide: title of the project, names of the team members, name of your TA, workload partitioning for each team member.
- Problem slide: which track (track 1 or track 2), describe the problem you are working on, summarize why you are working on this problem, and what the challenges are.
- Design slide: draw/summarize the whole design you have done so far, with key features.
- Results slide: demo and/or describe the functionality of the IoT system (track 1); show the initial performance results, if any (track 2). Indicate what you plan to do during the remaining time and final expected functionalities or results.

We will take notes during the presentation and your grades will be based on:

- Quality of slides and presentation. (2 pts)
- Ideas and initial results/functionalities. (2 pts)

- Difficulty level. (1 pt)
- Finish presentation on time. (1 pt)

Final Demo (May 6th) (30 pts)

In this final demo, you need to show a working system as proposed in your proposal.

- Show your design in detail, emphasize the novelty and contribution to the field of your project.
- Run your whole system end-to-end, and demo that it can achieve the functionality as described in your proposal.
- Evaluate your design quantitatively, and compare your design with the existing designs. For example, for ML algorithms and accelerators, you may want to report accuracy, FLOP/Parameter size, latency, throughput, energy consumption, etc. For IoT systems, you can discuss latency, throughput, cost, size/form factor, communication bandwidth, scalability, etc.

TAs may ask you questions related to the lab during your demo, please be prepared to answer TAs' questions.

Grading rubric:

- **Completeness**: Does your design achieve the goal in your proposal and have a working demo? (15 pts)
- **Quality**: How good is your design's quality? The quality includes many different aspects, e.g. accuracy, latency, power consumption, usability, etc. (5 pts)
- **Difficulty**: How difficult is it to implement your design? This evaluates the workload of your project. (5 pts)
- **Novelty/Uniqueness**: How different is your design from the existing ones? Are there any special features in your design? (5 pts)

Final Report (10 pts)

You need to submit a report that covers the details of your design.

- State clearly which track you chose and which platform you used. (1 pt)
- State clearly the problem you've solved and the challenges of solving the problem. (2 pts)
- Show your design in detail, emphasize the novelty and contribution to the field of your project. (3 pts)
- Evaluate your design quantitatively, and demonstrate how your design is better than the existing ones. (3 pts)
- Add references to your report if you referred to any resources when you work on your lab. (1 pt)

0.5 pt is the smallest grading step for all the scores above.

NOTE: We understand that the difficulty levels for implementing the same functionality can vary across different platforms. Therefore, we will take into consideration the platform you use when assigning difficulty points. For instance, making a DNN model's inference work on an FPGA is generally more challenging than on a GPU or CPU. Implementing a working CNN inference engine for a computer vision task on an FPGA aligns well with this lab's objectives. However, building one on a GPU may not suffice for a lab project due to the availability of numerous GPU libraries. Therefore, if you opt for GPU or CPU platforms, you should develop significant components not found in existing libraries or demonstrate a considerable workload/novelty in terms of algorithms/software, such as developing more accurate DNNs. AIE is a relatively new platform, and exploring its ML capabilities on a Ryzen CPU presents a challenging yet intriguing task. Not many people have experience with mapping Neural Networks onto AIE devices. Embarking on this venture could position you among the pioneers. We will also consider the challenging nature of exploring such new AI platform when grading your project.

Team with 2+ teammates:

You are responsible to convince the TAs in your proposal and throughout the project that your work justifies for a bigger team. We expect the significance of your final project is more substantial than that of the regular two-person teams. Please plan carefully.

Extra Credit: Mini-labs

Each team can choose to complete at most one mini-lab for extra credits. You will work on either designing and training a neural network with GPU or accelerate convolution with FPGA. You will demo your code during your final demo and include a brief report in your final project report as well.

Google Colab with GPU (5 pts)

Starting with this notebook as a template, design and train a network to classify the CIFAR-10 dataset. Please refer to this notebook for the usage of GPU in Google Colab.

You will need a Google account for Colab. The format of the Colab is the same as the Jupyter Notebook which you have been working with in lab 1 and lab 2. Once you click the link above, you can directly run the code on Google Colab and save to your account. The original notebook uses the MNIST dataset and shows how to instantiate a simple neural network. You should change the dataset to the CIFAR-10 dataset and design a new and more complicated network architecture. You need to achieve at least 85% classification accuracy to receive full credit.

Acceleration convolution on FPGA (5 pts)

In this mini-lab, we will explore the possibilities of accelerating convolution code on FPGA devices. Please download VivadoHLS 2019.2 here and open the pre-built project downloadable from our website. We also have recorded a video to help you get started, which is posted on the course website. This video will guide you through installing the tools and run the High-Level Synthesis for the code. You will insert loop unrolling, array partitioning, and pipelining pragmas through High-Level Synthesis to speed-up the computation latency.

In the given code, we have already given you a baseline implementation of a regular CNN-style convolution. Your task is to insert pragmas at the appropriate location to achieve better performance. After you have adjusted your code, you will need to collect your latency results by going through the simulation in VivadoHLS. Please check the video posted on the website for how to evaluate your FPGA design. You need to achieve at least 20 times speed-up against the original code to receive full credit.