ECE 538
Deep Learning for Embedded Real Time Intelligence

Spring 2022

Professor Richard Shi   cjshi@uw.edu

ELECTRICAL & COMPUTER ENGINEERING

Course Goal

While most deep learning tasks today are implemented with GPUs or high-performance CPU servers in the cloud, there is an increasing demand in implementing deep learning on internet-of-things (IoT) devices.

ECE 538 is developed to address this demand, especially from the local industry such as Amazon, Microsoft and Boeing. It is among the first courses in the nation to focus on the development of knowledges and skills for implementing embedded real-time artificial intelligence.

This course covers the basic concepts and techniques of deep learning, including feature extraction, various neural networks, and implementation and acceleration of deep neural network inference. The labs include deep learning programming on MCUs running a local Real Time operating system (RTOS) and/or using MCUs with an RTOS connected to a local deep learning accelerator chip. The target real-time applications are from intelligence in vision, nature language processing, and medical systems.

- **Prerequisite**: EE 235/341 (Linear and Discrete-Time Signal Processing)
- ECE 437 (Real-Time Embedded Operating Systems)
- Computer Programming C/C++ (CSE 374 or Equivalent)

ELECTRICAL & COMPUTER ENGINEERING

Eight Learning Objectives

1. Learn fundamentals of deep learning and modern neural networks
2. Learn how to build your own deep learning networks on IoT and mobile devices
3. Learn techniques for embedded deep learning implementation including bit width reduction, sparsification.
4. Learn how to analyze parallelism and data reuse in deep neural networks
5. Learn how to programming MCUs for data intelligence
6. Learn how to use a new programming language HALDL developed out of MIT CSL for programming and analyzing deep learning hardware
7. Learn complex embedded real-time software programming step by step
8. Learn how to use open-source resources (software) for engineering work and research

Tentative Topic Organization

1. Introduction to Artificial Neural Networks and Deep Learning
   a. History of Neural Networks; Hopfield models
   b. Recent explosion in data-driven deep learning
   c. Introduction to MCU/CPU, GPU, DSP, FPGA and domain-specific DL Accelerator
   d. Introduction to Software stacks for Deep Learning
   e. Brief introduction of three applications: object detection, voice recognition and medical data analysis
   f. Embedded computing: Azure RTOS C programming on MCUs
2. Deep Learning Concepts and Constructs
   a. Convolution: depth-wise, point-wise
   b. Transformer, encoder, decoder
   c. Attention
   d. Dilation and reception fieldsnv
   e. Skip, residual connection and 1x1 coolution
   f. Gating
   g. Relu, Pooling, Activation

3. Techniques for Implementing Deep Learning on Embedded Devices

   Quantization
   Sparsification
   Channel separation
   Network disfill: teacher-student-model

4. Techniques for Parallelism and Data Use in Deep learning

   Halide language
   Tensor variables and data objects expressions
   Vectorize, parallelize, unroll, and tile your code
   Realize functions over arbitrary domains
   Multiple-state pipelining and scheduling
   Multiple passes, data updates and reduction
   Cross compilation and compiler optimization

ELECTRICAL & COMPUTER ENGINEERING

Tentative Topic Organization: 2/2

5. Convolution Neural Networks
    5. AlexNet
    6. ResNet
    7. Yolo
    8. MobileNet
    9. SqueezeNet
6. Recurrent Neural Networks
    5. RNN
    6. LSTM
    7. GRU
7. Graph Neural Networks
    5. Basics of Graph
    6. Convolutional GNN
    7. LSTM/GRU GNN
8. Spiking Neural Networks
    5. Leaky-Integrate-Fire (LIF) model
    6. An R-C-Rectifier Implementation (Nature 2019 paper)
    7. The Legendre memory units (LMUs)
    8. Hybrid Spiking Neural Networks

Course Structure and Grading

- **Course Structure and Programming Assignments:** The class meets for three 50-minute lectures and one 50-minute TA-led lab session for students to work on programming assignments. For deep learning, students will learn as much as from programming assignments as from lectures. During the quarter, students will work, in terms of two, on several programming assignments. Each lab will add a new component of deep learning embedded application. By the end of the quarter, students will have built their own “complete” system of intelligent real-time application, including sensor-based real-time data acquisition, MCU data analysis, and neural network inference. The built system will be able to run on real hardware for real industry/business data.

- **Computer Resources and Lab Hardware:** Students will use their own laptops for MCU code development, or PCs in the department computer labs. Industry-popular MCUs with RTOS such as Microsoft Azure Sphere MCU and Azure RTOS will be used for project implementation.

- **Grading:** Approximate distribution: Programming Assignments 50%, Midterm 15%, Final Exam 30%, and 5% intangibles. The grading scheme in any particular offering is the prerogative of the instructor.

Textbooks. References and Open-Source Resources:

- No textbook but lectures and programming assignments will be based on the instructor’s notes, a collection of recent papers, and open-sources projects by industry leaders. Some exemplar papers include
- Yundong Zhang, Naveen Suda, Liangzhen Lai, Vikas Chandra “Hello Edge: Keyword Spotting on Microcontrollers”
  - Paper: https://arxiv.org/abs/1711.07128.
  - Open-source code and data: https://github.com/ARM-software/ML-KWS-for-MCU
  - Deep learning on cortex-M processors, ARM presentation: https://github.com/ARM-software/ML-KWS-for-MCU
- Joel Emer (MIT, NVIDIA), Vivienne Sze MIT, Yu-Hsin Chen MIT, Tien-Ju Yang MIT, “Tutorial on Hardware Accelerators for Deep Neural Networks”: http://eyeriss.mit.edu/tutorial.html
- Microsoft newly open-source real-time operating system (Azure RTOS): **https://github.com/azure-rtos**
- **MIT Halide open source** language for fast, portable data-parallel computation
  - https://github.com/halide/Halide
- M. J. Rozenberg, O. Schneegans & P. Stoliar, ``An ultra-compact leaky-integrate-and-fire model for building spiking neural networks”, **Nature Scientific Report**, 2019; https://www.nature.com/articles/s41598-019-47348-5

ELECTRICAL & COMPUTER ENGINEERING