Accelerator Architectures for Machine Learning

Lecture 1: Course Introduction
Tsung Tai Yeh
Monday: 9:00 – 9:50 am
Thursday: 1:20 – 3:10 pm
Classroom: ED-102
Acknowledgements and Disclaimer

• Slides was developed in the reference with
  Joel Emer, Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, ISCA 2019 tutorial
  Efficient Processing of Deep Neural Network, Vivienne Sze, Yu-Hsin Chen, Tien-Ju
  Yang, Joel Emer, Morgan and Claypool Publisher, 2020
  Yakun Sophia Shao, EE290-2: Hardware for Machine Learning, UC Berkeley, 2020
  CS231n Convolutional Neural Networks for Visual Recognition, Stanford University,
  2020
A New Course!

- The human-being brain comprises different areas (accelerators)
- Designing "Accelerator" brain to boost up Machine Learning

https://askabiologist.asu.edu/sites/default/files/resources/articles/nervous_journey/brain-regions-areas.gif
A New Course!

- Designing “Accelerator” brain to boost up Machine Learning

https://upload.wikimedia.org/wikipedia/commons/2/27/Artificial_Neural_Network_with_Chip.png
Missions of This Course

• To achieve the best **performance and energy efficiency** hardware, **software, systems** for Machine Learning (ML) workloads

Cross-layer solutions:
Programming Language to Circuit

Missions:
1) Digging out performance bugs (bottlenecks) and to kill(fix) them
2) Synthesize ML models to hardware
Goals of this course

• Evaluate approaches to achieve efficient Deep Neural Network (DNN) hardware accelerators
  • What are the **key metrics** and **design objectives**?
  • What are the **challenges** to achieve these design metrics?
  • What are the **design considerations** and tradeoffs?
  • What are the **potential technologies** to advance the design of DNN hardware processors?
What will you learn from this course?

1. Deep Neural Network models
2. Popular types of layers in DNNs
3. Training v.s. Inference

1. TensorFlow/PyTorch/CUDA
2. ML kernel computation
3. cuDNN optimization

1. GPU/TPU micro-architecture
2. DNN specific accelerators
3. DNN hardware design aspects
What will you gain from this course?

• Cross-layer knowledge to accelerate ML workloads

Front-End

PyTorch  TensorFlow  Keras  Scikit Learn

IR

TVM  Tensor Algebra Compiler (TACO)

Back-End

ONNX Runtime  cuDNN library

Accelerators

GPU  TPU  FPGA  ASIC
Prerequisites

• **Courses:**
  • Basic Programming, Computer Organization, Advanced Computer Architecture

• **You should:**
  • Basic understanding of computer architecture and digital logic design
  • Comfortable with programming in C/C++ and Python
Textbook

• Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, **Efficient Processing of Deep Neural Network**, Morgan and Claypool Publisher, 2020

• You can download the e-book from NCTU library through EBSCOhost E-book database within NCTU campus
What will you need to do in this course?

- Reading papers and writing paper summary (20%)
- 2 Lab projects (30%) and 1 homework assignment (5%)
- 1 Final Project (45%)
  - Micro-architecture project (GPGPU-Sim, OpenTPU, SCALE-Sim)
  - FPGA project (Chipyard, NVDLA, FPGA Dev. Board)
  - System Software project (Coral TPU Dev. Board, Nvidia Tegra Dev. Board, GPU card)
- Rule:
  1) 2 – 3 people/group,
  2) Your-decision or My-decision projects
  3) Please talk with me at the office hour before doing the project
Final Project Potential Topics

• Deep learning, machine learning, neuromorphic hardware, software and hybrid techniques
  • Neural processing, near-data and in-memory accelerators
  • Micro-architecture optimizations for GPUs
  • Micro-architecture for optimizing memory hierarchy
  • Micro-architecture modeling and simulation methodology
  • Architectures for embedded platforms including mobile phones, self-driving vehicles
  • Insightful experimental evaluation of existing hardware/software and workloads

• Goals
  • Improve system performance
  • Energy-efficiency
  • Security and reliability
  • Quality of service
Readings

• Submit your paper review on each paper due (11:59 pm) (No count the submission after the deadline!!) (300 - 800 words in English)

• Your paper summary should include:
  • What are the motivations of this work?
  • What are problems of this paper?
  • What are proposed solutions of this paper?
  • How to proceed evaluations to justify proposed solutions of this work?
  • What is YOUR analysis of the identified problem, idea, and evaluations?
  • What questions are you left with?
How to read academic papers?

• Tips for reading academic papers
  • https://web.stanford.edu/class/ee384m/Handouts/HowtoReadPaper.pdf
  • Don’t read the paper word by word !!
  • Figure out problems of a paper at first !!
  • Don’t jump into methodology details easily !!
  • Read the background section if necessary !!
  • Don’t skip reading figures !!
  • Find out answers based on contributions of a paper
  • Ask yourself pros and cons of a paper after the reading
New E3 Course Platform

• Course Website:  
  https://people.cs.nctu.edu.tw/~ttyeh/course/2020_Fall/IOC5009/outline.html

• New E3  
  • Discussion Forum  
  • Reading summary and assignment submission
Resources

• Hardware
  • Jetson AGX Xavier GPU Developer Kit
  • Coral TPU Developer Kit
  • Xlinx PYNQ-Z1 FPGA Developer Kit
  • NVIDIA TESLA T4 GPU
  • Google colab TPU/GPU

https://coral.ai/products/
Introducing the lecturer

• Lecturer: Tsung Tai Yeh
  • E-mail: ttyeh@cs.nctu.edu.tw
  • Office: EC 707
  • Office Hours: Thursday 3:30 – 4:30 pm
• Research topics:
  • Computer architecture
  • Computer systems
  • Memory and storage systems
  • Domain-specific accelerators (GPU, Neural Processing Units)

“Hiring graduate and under-graduate students”
Why Deep Neural Network become popular?

• DNN model outperforms human-being on the ImageNet Challenge

![Graph showing Top-5 Error Rate (%)](https://arxiv.org/ftp/arxiv/papers/1911/1911.05289.pdf)
No free lunch on DNN computation

- AlexNet to AlphaGo Zero: A 300,000 x Increase in Compute

Increasing transistors is not getting efficient

General purpose processor is not getting faster and power-efficient because of

Slowdown of Moore’s Law and Dennard Scaling

Need Specialized/Domain-specific accelerators to improve computing speed and energy
Moore’s Law

• The number of transistors per chip **doubles** every 18-24 months
• That has not been true for years
• It is getting to be increasingly difficult to maintain this exponential improvement !! Why?
Dennard Scaling

• As the size of the transistor becomes small
  • The voltage is reduced
  • Circuits can be operated at higher frequency at the same power

\[ \text{Power} = \alpha \times CFV^2 \]

\( \alpha \): percent time switched
\( C \): capacitance
\( F \): Frequency
\( V \): Voltage

What’s wrong on Dennard Scaling?

Dennard Scaling ignores “leaking current”, “threshold voltage”

So, as transistors get small, power density is increase !!
Domain Specific Architecture (DSAs)

• Achieving higher performance by tailoring characteristics of domain applications to the architecture
  • Need domain-specific knowledge to work out good DSAs
  • Domain Specific Languages (DSLs) + DSAs (not strict ASIC)
  • Specialize to a domain of many applications

• Examples
  • GPU for computer 3D graphics, virtual reality
  • Neural processing unit (NPU) for machine learning
  • Visual processing unit (VPU) for image processing
Domain-Specific Accelerators in SoCs

• What’s difference among each Apple processor generation?

2010 Apple A4
65 nm TSMC 53 mm²
4 accelerators

2014 Apple A8
20 nm TSMC 89 mm²
28 accelerators

2019 Apple A12
7 nm TSMC 83 mm²
42 accelerators

https://edge.seas.harvard.edu/files/edge/files/alp.pdf
Why DSAs can win?

• More effective parallelism for a specific domain
  • SIMD vs. MIMD
  • VLIW vs. Speculative, out-of-order
• More effective use of memory bandwidth
  • User controlled vs. caches
• Eliminate unneeded accuracy (Quantization)
  • Lower FP/INT data precision (32 bit integers -> 8 bit integers)
• Increase the hardware utilization
  • Reduce the idle time on pipelining and LD/ST
Domain Specific Languages (DSL)

• DSLs target specific operations on a domain of applications
• Need vector, matrix or sparse matrix operations
• DSLs tailors for these operations
  • OpenGL, TensorFlow, Halide
• Compilers are important if DSLs are architecture-independent
  • Translate, schedule, map ISAs to right DSAs
Designing your own DL accelerator

• Build better **algorithms**
• Build better **programs/runtimes**
• Build better **hardware**
• Mapping DNN dataflow to the hardware
<table>
<thead>
<tr>
<th></th>
<th>Nvidia V100 GPU (2019)</th>
<th>Nvidia A100 GPU (2020)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transistor count</td>
<td>21 billion</td>
<td>54 billion</td>
</tr>
<tr>
<td>FP32 performance</td>
<td>15.7 TFLOP/s</td>
<td>19.5 TFLOP/s</td>
</tr>
<tr>
<td>Tensor FP32</td>
<td>125 TFLOP/s</td>
<td>156 TFLOP/s</td>
</tr>
<tr>
<td>TDP</td>
<td>300 W</td>
<td>250 W</td>
</tr>
<tr>
<td>Die size</td>
<td>815 mm²</td>
<td>862 mm²</td>
</tr>
<tr>
<td></td>
<td>TSMC 12 nm</td>
<td>TSMC 7 nm</td>
</tr>
</tbody>
</table>

2.57 X
1.24 X
1.25 X
DSA (II) Google Tensor Processing Unit (TPU)

- Systolic-array accelerator
  - V1: Inference only
  - V2: Training with bfloat
  - V3: 2X powerful than v2
- Edge TPU
  - Coral Dev Board
  - 4 TOPS
  - 2 TOPS/Watt
  - Support TensorFlow Lite

https://coral.ai/products/
https://cloud.google.com/tpu/docs/tpus
DSA (III) Nvidia Tegra AGX Xavier GPU

- 32 TOPS
- 512-core Volta GPU with 64 Tensor Cores
- 8-core Carmel ARM v8.2 64-bit CPU
- 32GB 256-Bit LPDDR4x

https://en.wikichip.org/wiki/nvidia/tegra/xavier
DSA (II) Cerebras: Wafer-Scale DL Engine

- Largest DL Chip Ever Built!!
- 46225 mm² (WoW !!)
- 1.2 trillion transistor
- 400,000 optimized AI cores
- 18 GB on-chip memory
- TSMC 16 process

https://twitter.com/CerebrasSystems/status/1163443985714753537
MLPerf Benchmark

• DL inference/training benchmarks for DL models with different DSLs (TensorFlow/pyTorch...)

https://mlperf.org/
At the end of this course

Input

ML Models/benchmarks

Your SW Optimization

Your HW Optimization

Output

Accelerators

Accelerator System

Accelerator System