CSC290/571 Topics in Systems: Machines Learning Systems
Introduction

Sreepathi Pai

August 26, 2024

URCS

Outline

AI/ML: An Overview

A Systems View

Administrivia

Outline

AI/ML: The acronyms

- Artificial Intelligence
  - roots go back to the 50s
- Machine Learning
  - likewise
- Neural Networks (NN)
  - "Universal" functions
  - that can "learn" (or more accurately, can be trained)
- Deep Learning (DL)
  - Neural Networks with lots of layers

ML applications of Interest

- Large Language Models (LLMs)
  - various linguistic tasks
  - e.g., ChatGPT, LLAMA, etc.
- Diffusion Models
  - mostly image generation (?)
  - e.g., Stable Diffusion
- These have:
  - high amounts of compute
  - high amounts of memory usage
  - large amounts of data involved
- Sometimes called "foundation models"

Three tasks

- Training
  - Models are trained on examples
  - “Learning” phase
  - Time-consuming, very expensive for LLMs
  - Almost entirely restricted to large, well-resourced entities
  - Done once
- Fine-tuning
  - “Low-cost” retraining for specific tasks
  - Done once per application
- Inference
  - “Lowest” cost
  - Once per user request

Outline

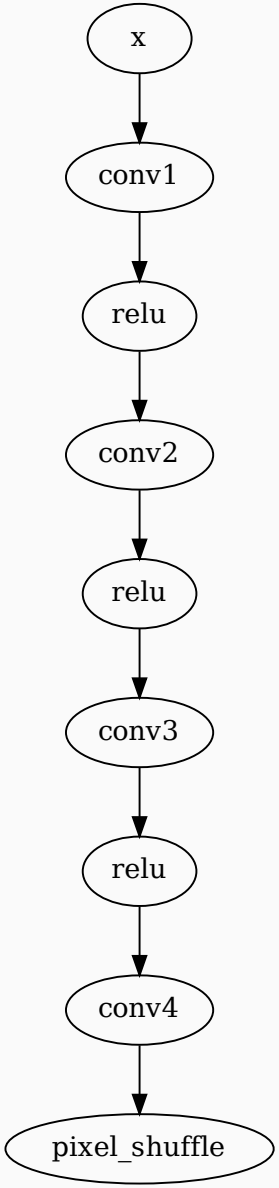
ML Programs

- A ML program will be viewed as a series of operations
  - or "primitives"
  - similar to computer instructions
- Unlike instructions, most operations consume and produce large amounts of data
  - vectors, matrices, etc. (also called tensors)
- ML programs usually have no conditional control flow
  - All operations are executed
- ML programs usually do not have loops at the operation level
  - The programs are direct acyclic graphs (DAGs)

An example

```
import torch
import torch.nn as nn
import torch.nn.init as init


class Net(nn.Module):
    def __init__(self, upscale_factor):
        super(Net, self).__init__()

        self.relu = nn.ReLU()
        self.conv1 = nn.Conv2d(1, 64, (5, 5), (1, 1), (2, 2))
        self.conv2 = nn.Conv2d(64, 64, (3, 3), (1, 1), (1, 1))
        self.conv3 = nn.Conv2d(64, 32, (3, 3), (1, 1), (1, 1))
        self.conv4 = nn.Conv2d(32, upscale_factor ** 2, (3, 3), (1,
        self.pixel_shuffle = nn.PixelShuffle(upscale_factor)

        self._initialize_weights()

    def forward(self, x):
        x = self.relu(self.conv1(x))
        x = self.relu(self.conv2(x))
        x = self.relu(self.conv3(x))
        x = self.pixel_shuffle(self.conv4(x))
        return x

    def _initialize_weights(self):
```

Graph



ML Computers

- Multicore CPUs
  - most common for inference of older models
- GPUs (most common)
  - graphics processing units
  - most common for training and generative AI inference
  - NVIDIA, AMD
- Multicore CPUs + AI accelerators
  - Intel Gaudi
- Specialized Processors
  - Google's Tensor Processing Units
  - Groq "LPU"
  - Cerebras Wafer-scale
  - ...

Running a ML program

- Eager Execution
  - Each operation executes as soon as it is encountered
- Graph Execution
  - A graph of operations is first built
  - Then, the graph is compiled and optimized
  - The compiled graph is executed

Eager Execution

- Each operation maps to one or more primitives
- A primitive implemention exists for each possible device (CPU, GPU, etc.)
    - e.g., as a collection of library functions, GPU kernels, etc.
- An operation decomposes into calls to these pre-existing functions
- It is possible that primitives may be generated on the fly

Graph Execution

- Each operation maps to one of more primitives
- Non-trivial transformations may be applied to graphs
  - Fusion: operations are combined (e.g., Convolution + Relu)
- Operations scheduling may change
  - Reduce memory traffic
  - Use multiple GPUs, etc.
- Primitives may generated and specialized per operation
  - A different matrix multiplication for each different matrix size

ML Primitives

- Examples
  - Convolution
  - Matrix Multiplication
- Started out as handwritten functions for each device
  - Provided by device vendors
  - NVIDIA's cuBLAS, Intel's MKL, etc.
- Now are usually generated by specialized compilers

Compiling ML Computation Primitives

- Example: Matrix Multiplication
- Many different approaches:
  - Tensor compilers: e.g., TVM, Tensor Comprehensions, etc.
  - Tile compilers: e.g., Triton
- Source language for these tools is device independent
  - Usually, some domain-specific language
- Progressively lowered to device-specific code
  - Using, for example, the MLIR framework

ML Communication Primitives

- Once ML progams run on multiple GPUs or multiple machines, some mechanism for communication is required
- Communication Primitives (not exhaustive)
  - Gather
  - Scatter
  - AllGather
  - Broadcast
  - Reduce
  - AllReduce

Goals

- Understand performance of these ML programs
- Improve performance

Outline

People

- Instructor: Dr. Sreepathi Pai
  - E-mail: sree@cs.rochester.edu
  - Office: Wegmans Hall 3409
  - Office Hours: Monday 15:00 to 16:00 (or by appointment)

Places

- Class: Hylan 303
  - M,W 0900–1015
- Course Website
  - `https://cs.rochester.edu/~sree/courses/csc-290-571/fall-2024/`
- Blackboard
  - Announcements, Discussions

Grading

- Homeworks: 15%
- Paper Discussion: 25%
- Mid-term: 10%
- Project: 50%
- Graduate students should expect to read a lot more, and work on harder problems.

There is no fixed grading curve. See course website for grade scale.

See course website for late submissions policy.

Project Outcomes

- Each project must contribute a novel piece of knowledge
  - Could be a system
  - Could be a new perspective
- Must be "solid" contribution
- Can be individual or team
  - Team results need to reflect size of team
- Highly structured process
  - At least 3 ideas and/or proofs of concept by mid October
  - More details as we go along
- Intended to seed future research projects and papers

Academic Honesty

- Unless explicitly allowed, you may not show your code to other students
- You may discuss, brainstorm, etc. with your fellow students but all submitted work must be your own
- All help received must be acknowledged in writing when submitting your assignments and homeworks
- All external code you use must be clearly marked as such in your submission
    - Use a comment and provide URL if appropriate
- If in doubt, ask the instructor
- It is a violation of course honesty to make your assignments on GitHub (or similar sites) public
- Use of AI tools for Homeworks and Paper Discussions is prohibited.
    - All use of AI tools for other purposes must be discussed and approved by the instructor

Course Goals

- Read and understand computer systems literature
- Develop skills to propose and carry out research projects
- Learn presentation and writing for research skills

How is a seminar course different?

- Mostly reading papers
- Your instructor is probably reading the papers for the first time
  - Same as you
  - Has a lot more experience than you though
- Aim to turn into expert on the literature