Machine Learning Systems (MLS)

Semester 2 (2024 - 2025)

Luo Mai

Timetable

- Thursday Q&A Lecture (14:10 – 15:00)
  - Demos, tutorials, coursework announcement and Q&A
  - **Locations vary over weeks**
- Friday Live Lecture (15:10 - 17:00)
  - Teaching key concepts in ML systems and Q&A
  - **Locations vary over weeks**

- https://browser.ted.is.ed.ac.uk/generate/?courses%5B%5D=INFR11269_SS1_SEM2&period=SEM2&week=26-37

Information

- The learning outcomes and information can be found here http://www.drps.ed.ac.uk/24-25/dpt/cxinfr11269.htm
- Course schedule and slides are on Learn https://www.learn.ed.ac.uk/ultra/courses/_122906_1/outline

Lecturers

- Prof. Luo Mai (luo.mai@ed.ac.uk)
  - Assistant Professor, School of Informatics
  - Homepage: https://luomai.github.io

- Prof. Yang Cao (yang.cao@ed.ac.uk)
  - Assistant Professor, School of Informatics
  - Homepage: https://homepages.inf.ed.ac.uk/ycao/

Guest lecturers

- Dr. Antreas Antoniou (iam@antreas.io)
  - Researcher, Malted AI
  - Lecture: LLM 101 and Model Compression

- Dr. Alec Diallo (alec.frenn@ed.ac.uk)
  - Researcher, University of Edinburgh / NetAI
  - Lecture: Security in ML Systems

Communication

- Piazza (accessible via Learn) is our main discussion platform.
- TAs will monitor and address questions
  - Selected ones discussed in weekly tutorials.
- You are welcome to answer questions and make contributions
  - Feel free to contribute by answering questions and providing feedback.
  - Join us in building MLS, the first program of its kind worldwide!

Topics

- Introduction to ML systems and frameworks
- GPU architecture and programming
- Vector search and management
- Data processing pipelines
- ML model compression (guest lecture)
- Distributed training systems
- Systems for serving ML models
- Security issues of ML systems (guest lecture)
- Research topics of ML systems

Group coursework

- Goal: Learn to develop, optimize, evaluate and report a ML system
- Group size: 3 – 4 students (formed by your own)
- Stage 1: Implement a ML systems for two tasks
  - Task 1: GPU programming and optimization for a given ML task
  - Task 2: Operating and optimizing a distributed ML systems
  - Code template will be provided
- Stage 2: Write a paper to describe system design and evaluation
  - The paper is structured as a typical ML conference submission
  - Paper template will be provided

Individual coursework

- Goal: Learn to read and assess an ML system paper
- Write a review for the report and code of an ML system
  - Paper review
  - Source code and documentation evaluation review
  - A review template will be provided

Marking

- Goal: Peer learning, rich feedback and marking transparency
- Paper submission and reviewing platform will be provided
  - The reviewing process is double-blinded
- The final mark is decided by
  - The quality of group paper and code, assessed by peers and lecturers
  - The quality of the reviews you wrote, assessed by lecturers
  - Group work contributes 70% and individual work contributes 30%

Group

- **Group Registration**: To register your group, please visit:
  - https://bit.ly/4amU9O6
- **Finding a Group**:
  - If you are looking for a group to join, add your details to: https://bit.ly/3DVc5mX
  - If you are seeking group members or wish to invite someone to join your group, use the sheet above to connect with potential teammates.

For any questions related to grouping, please contact:

- **Email**: y.jiang-84@sms.ed.ac.uk

Questions?

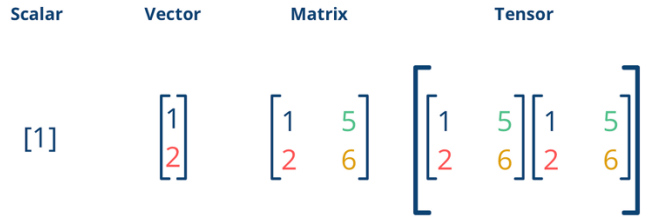
3 factors for AI booming

- Bigger **data** – the fuel of powering AI
  - Booming of Internet
  - Wide adoption of mobile phones and intelligence devices
- Stronger **algorithm** – the art of creating AI
  - Deep neural network (DNN), such as Transformers
  - DNN training methods, such as Adam
- Faster **hardware** – the foundation of running AI
  - GPU and NPU – 100X faster than CPUs in tensor operations
  - Moore-law – GPU performance double every year
  - Data centres – consolidating massive parallel GPUs becomes easy

Data in ML systems

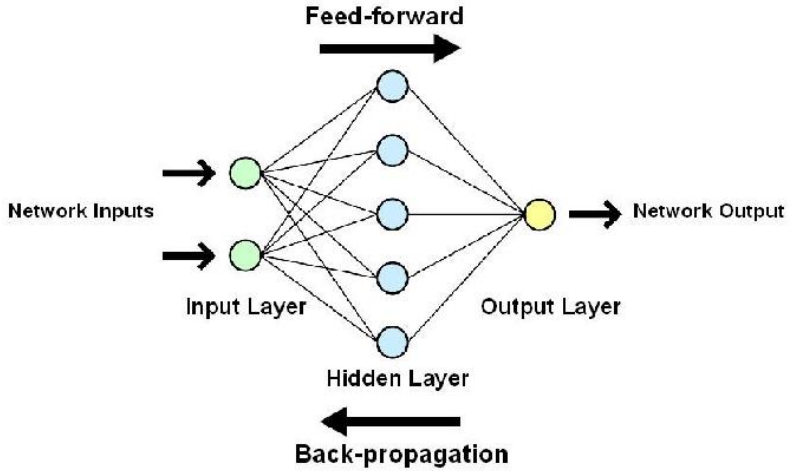
- Data types
  - Text
  - Images
  - Video
  - Audio

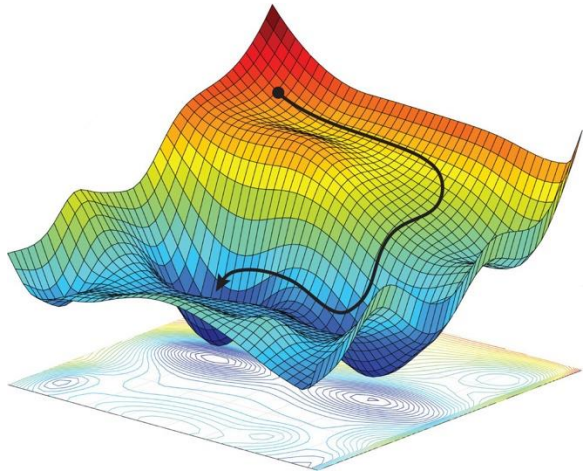
- Data representation
  - Tensors



[1] https://spotintelligence.com/2023/12/20/data2vec/
[2] https://medium.com/@er.sumitsah/an-introduction-to-tensors-understanding-the-mathematical-powerhouse-e04f53be9bee

Deep neural networks and training methods



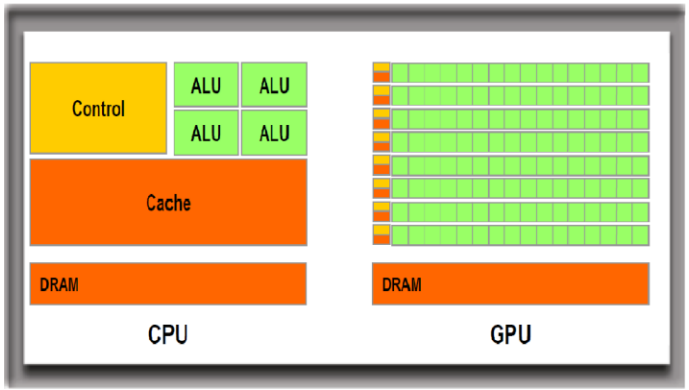


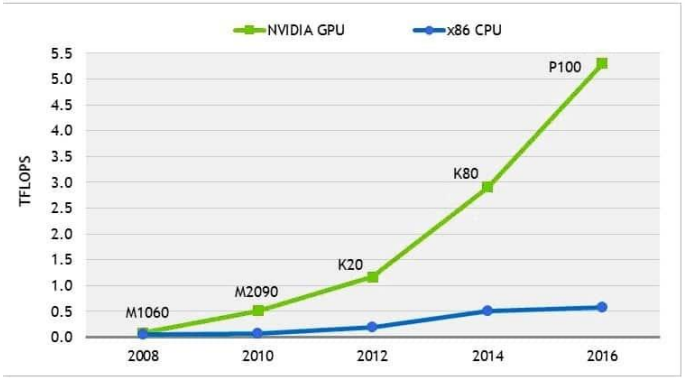
Stochastic **gradient descent** in a DNN's loss space

[1] https://botpenguin.com/glossary/deep-neural-network
[2] https://towardsdatascience.com/on-optimization-of-deep-neural-networks-21de9e83e1

AI accelerators

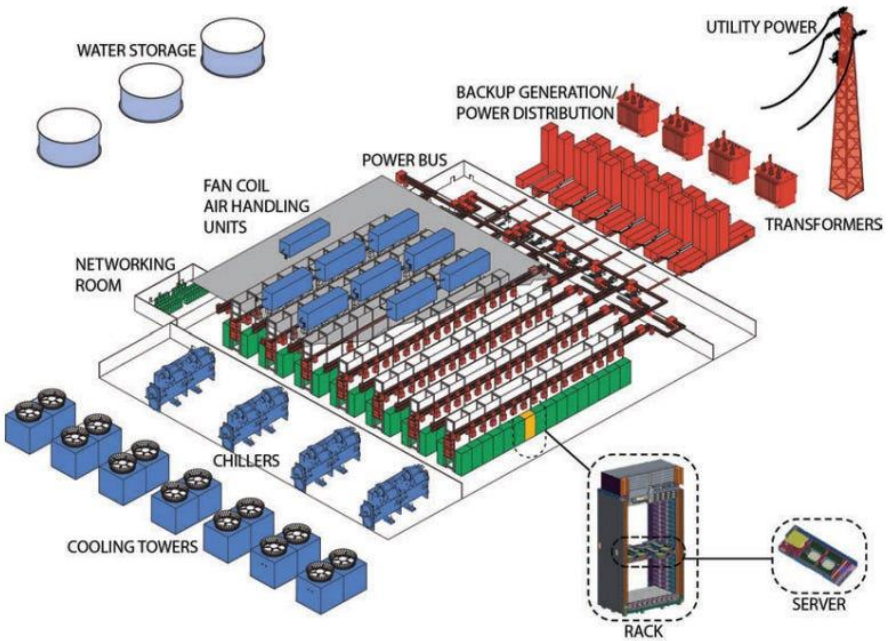
- Graphics processing unit (GPU)
  - Nvidia, AMD, Intel
- Tensor processing units (TPU) / Neural Processing Units (NPUs)
  - Google, Amazon, Apple, …





[1] https://www.heavy.ai/technical-glossary/cpu-vs-gpu
[2] https://www.rtinsights.com/gpus-the-key-to-cognitive-computing/

AI data centres



Independent report

AI Opportunities Action Plan

Published 13 January 2025

2. **Expand the capacity of the AI Research Resource (AIRR) by at least 20x by 2030 - starting within 6 months.** The AIRR should evolve into a set of mission-oriented clusters that bring together compute, data, and talent to pursue frontier AI research and other national priorities. Expansion by at least 20x by 2030 would ensure the AIRR enables the training of multiple AI models a year and provides an up to date research capability.[footnote 1] Given trends in hardware performance, this would not mean a 20x increase in investment if the government procures smartly.[footnote 2] Such expansion is needed to keep up with the expected increases in computing power that we should assume will be needed for AI workloads. This is unlikely to slow down; we need to "run to stand still". As part of this, government should ensure that the public compute ecosystem hosts a range of hardware providers to avoid vendor lock-in and ensure value for money.

[1] https://www.construction-physics.com/p/how-to-build-an-ai-data-center
[2] https://www.gov.uk/government/publications/ai-opportunities-action-plan/ai-opportunities-action-plan

Requirements for ML systems

- **DNN programming**
  - Simplify the definition, modification, and testing of DNN using high-level APIs.
- **Automatic differentiations**
  - Automatically compute gradients for backpropagation in DNNs.
- **Tensor processing**
  - Transform diverse data into a unified tensor format for efficient processing.
- **Training & deployment**
  - Implement common training methods and enable easy model deployment.
- **AI accelerators**
  - Offload computations to various AI accelerators for improved performance.
- **Distributed execution**
  - Distribute DNN workloads when a single accelerator’s memory is insufficient.

Questions?

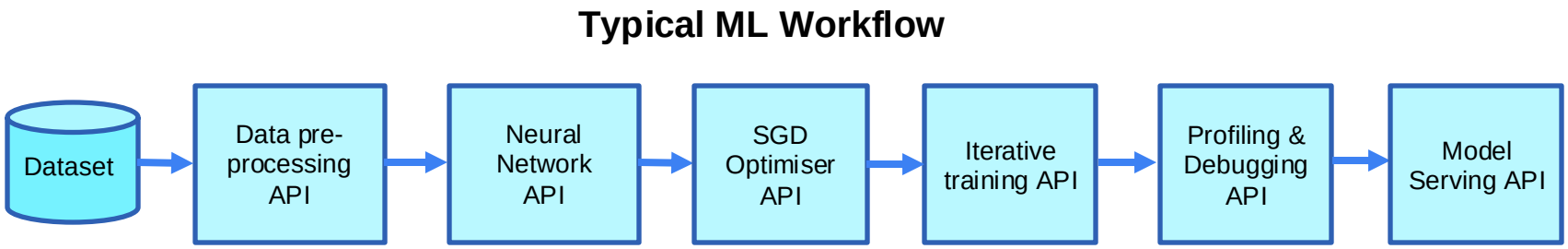
Deep Dive of ML Frameworks

Luo Mai

ML Framework vs. Prior Systems

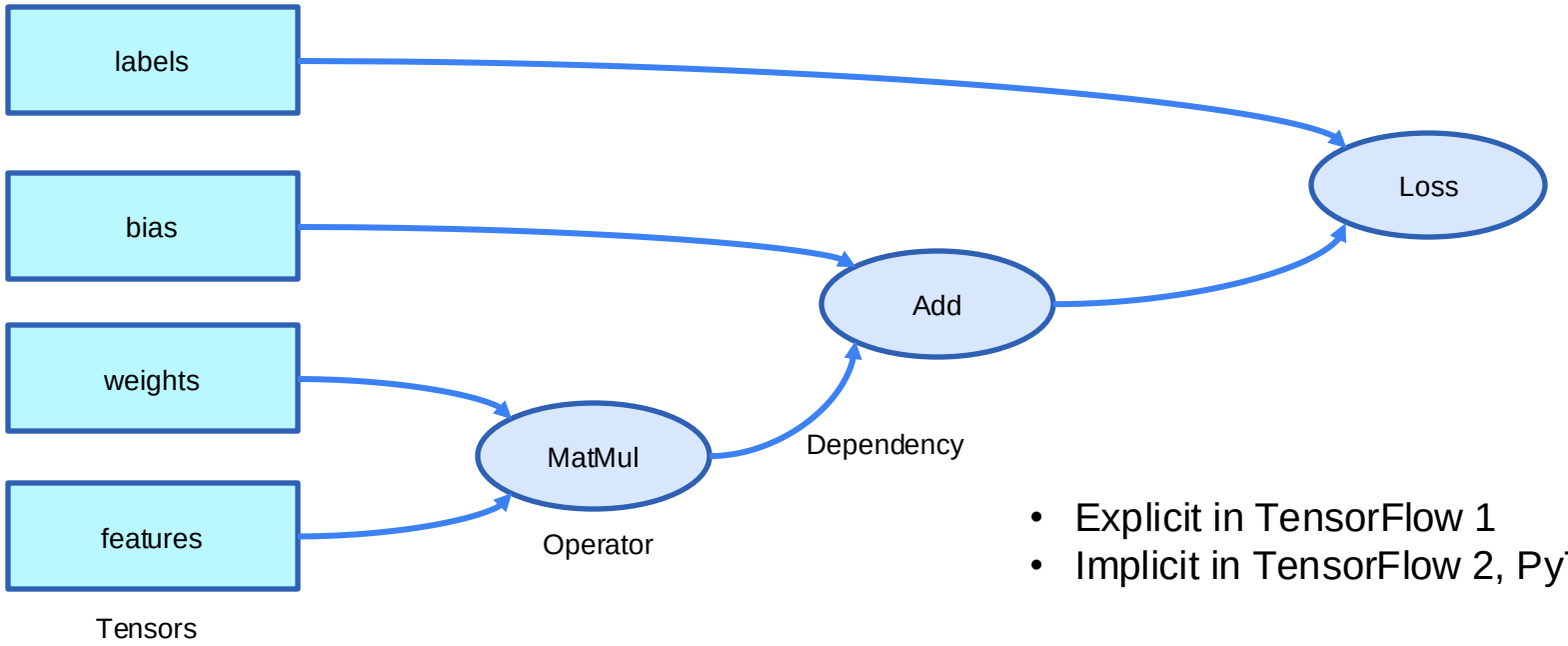
	DNN programming	Automatic Differentiation	Tensor data processing	Training & Deployment	AI Accelerators	Distributed Execution
Neural network libraries (Theano, Caffe)	✓	✓	✗	✗	✓	✗
Data parallel systems (Spark, Giraph)	✗	✗	✗	✗	✗	✓
ML framework (PyTorch, TensorFlow)	✓	✓	✓	✓	✓	✓

ML framework programming abstraction



A unified programming abstraction for different ML applications (DNNs, GNNs, DRLs, …)

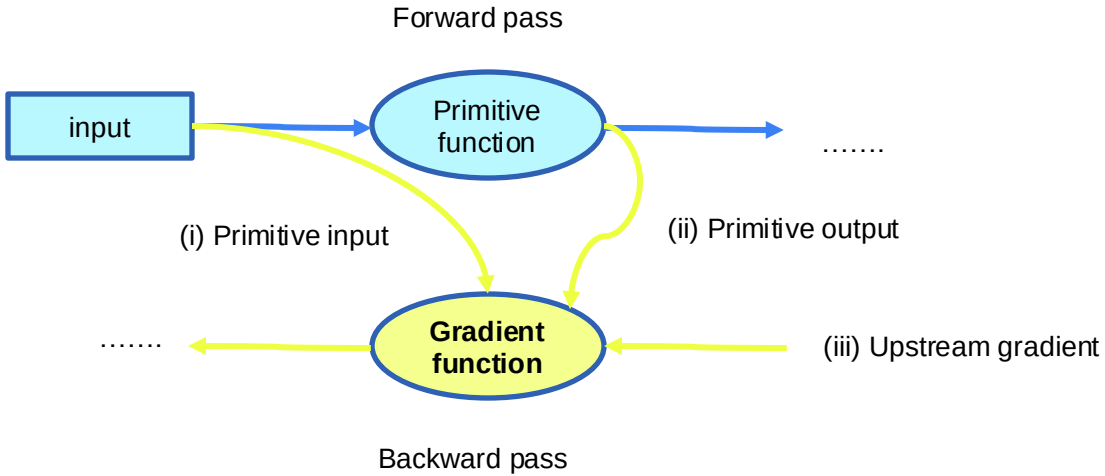
Expressing ML programs using a computational graph



- Explicit in TensorFlow 1
- Implicit in TensorFlow 2, PyTorch

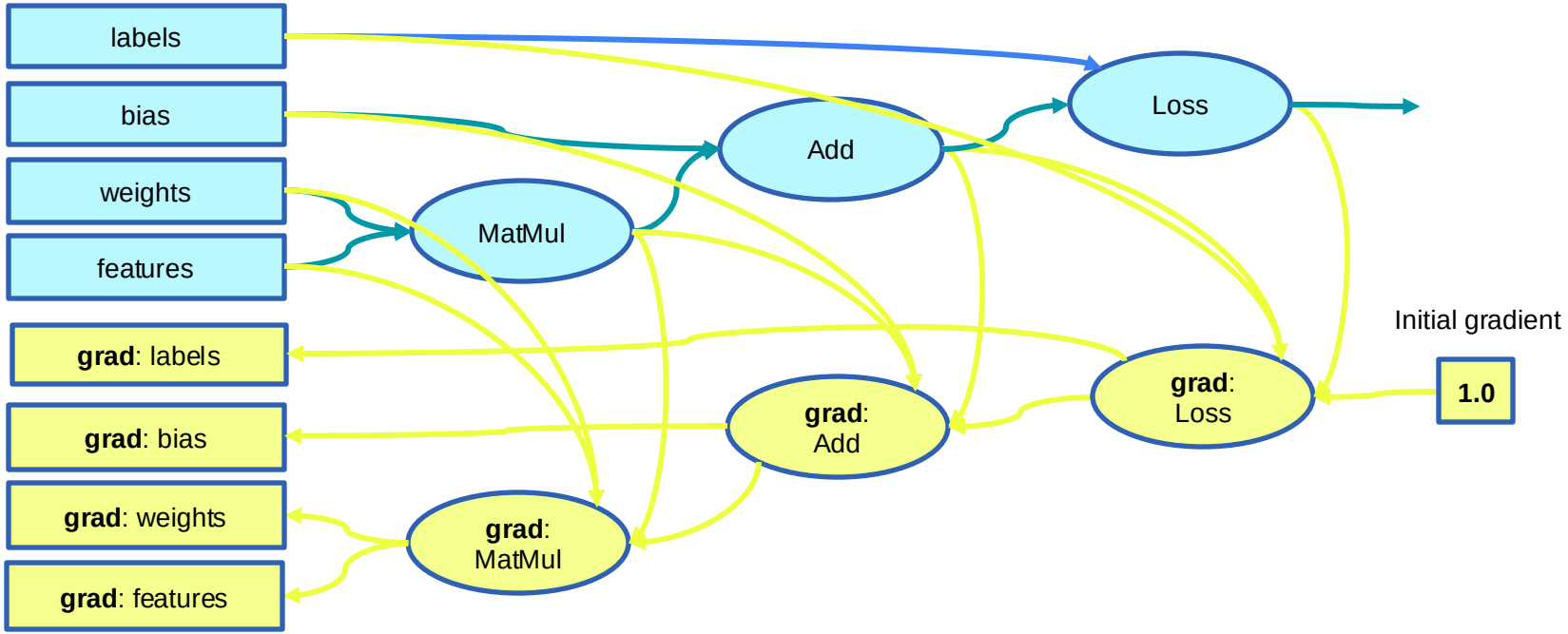
Automatic differentiation via the computational graph

Automatic differentiation through the chain rule: Gradient function takes its primitive function (i) inputs and (ii) output as parameters along with the (iii) gradient of the function outputs with respect to the final outputs.




```python
import torch

class MyMultiply(torch.autograd.Function):
    @staticmethod
    def forward(ctx, x, y):
        # Save inputs for backward computation
        ctx.save_for_backward(x, y)
        # Perform forward computation
        return x * y

    @staticmethod
    def backward(ctx, grad_output):
        # Retrieve saved tensors
        x, y = ctx.saved_tensors
        # Compute gradients of inputs
        grad_x = grad_output * y
        grad_y = grad_output * x
        return grad_x, grad_y

# Using the custom function
x = torch.tensor(2.0, requires_grad=True)
y = torch.tensor(3.0, requires_grad=True)

# Apply the custom operation
z = MyMultiply.apply(x, y)
z.backward()

print(x.grad)  # Output: 3.0 (dz/dx)
print(y.grad)  # Output: 2.0 (dz/dy)
```

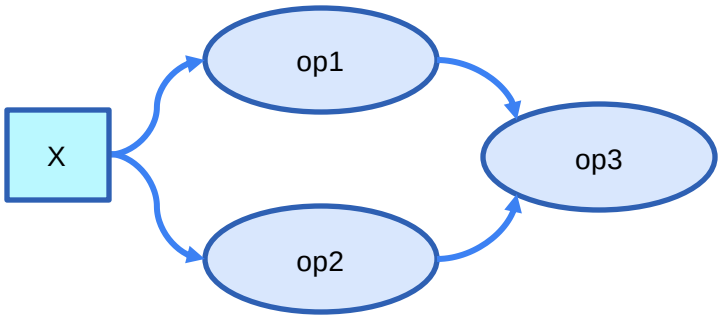
Generalizing the chain rule across the graph

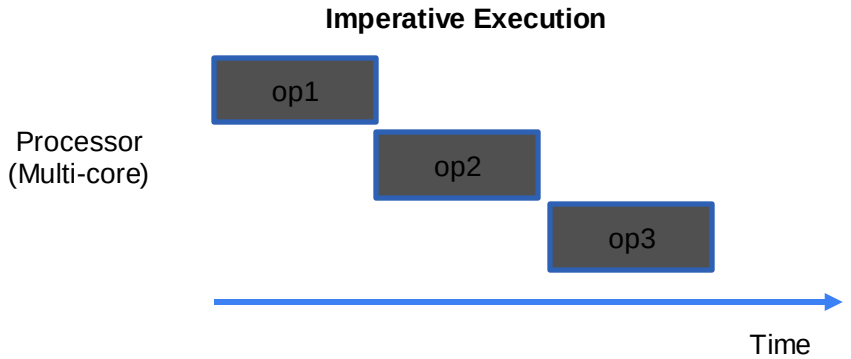


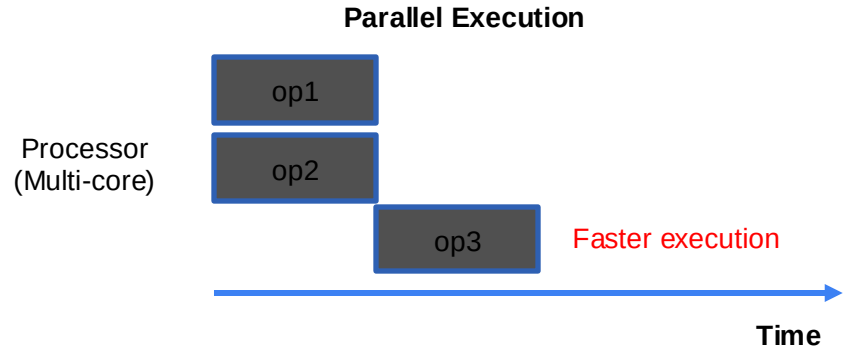
Discovering parallelism for improved performance

```
def model(x):
    y1 = op1(x)
    y2 = op2(x)
    return op3(y1, y2)
```

Equivalent Graph





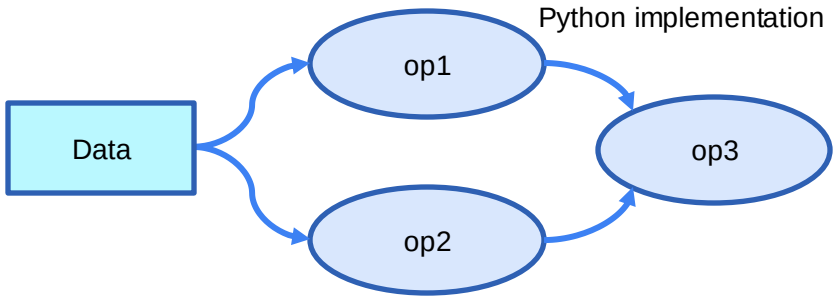


Questions?

Frontend and backend programming languages

Front-end language: Python

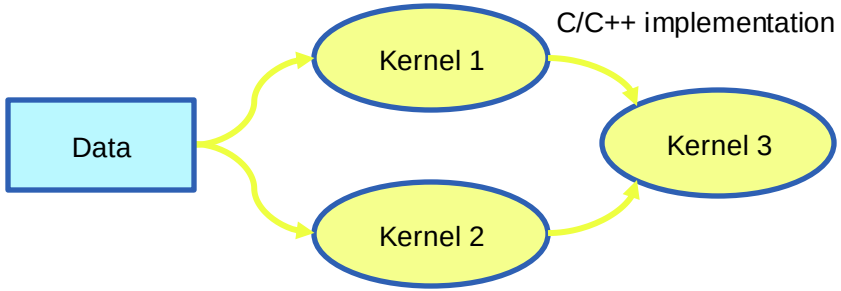
- Simple and flexible
- Poor performance
- Global Interpreter Lock (GIL)



Equivalent Back-end Graph

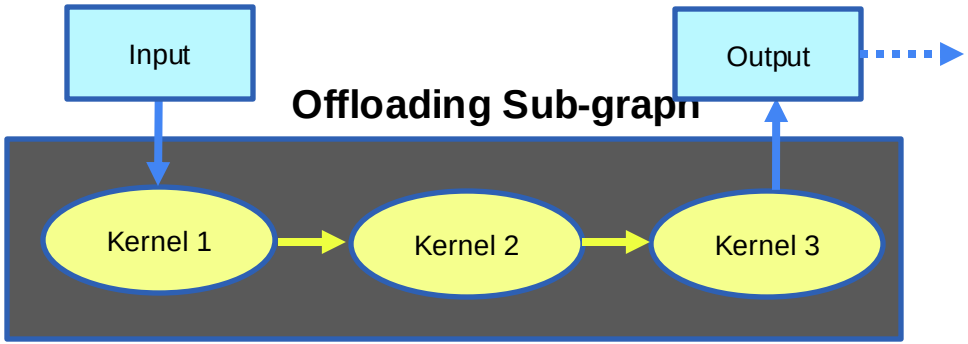
Back-end language: C/C++

- Hardware-friendly
- Excellent performance



Offloading sub-graphs to accelerators

Problem: Frequently launching C++ kernels (e.g., system calls) in Python has large overhead



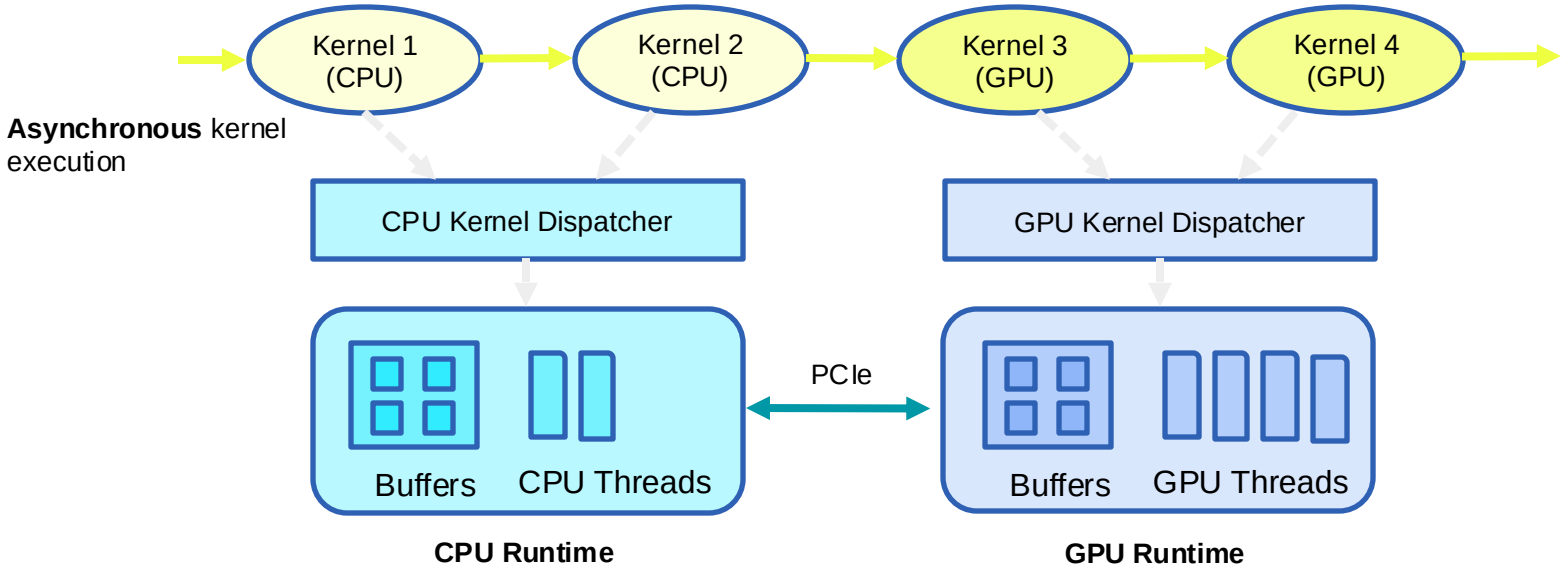

```python
import torch

# Example model
class MyModel(torch.nn.Module):
    def forward(self, x):
        return x * 2 + 3

# Convert to TorchScript
model = MyModel()
scripted_model = torch.jit.script(model)

# Execute the compiled sub-graph
x = torch.tensor(5.0)
print(scripted_model(x))  # Output: 13.0
```

Discovering Sub-graph

- **User annotation** to discover sub-graphs: @tf.function (TensorFlow 2), @jit.script (PyTorch)
- **Just-in-Time (JIT)** compilation: @jit.script (PyTorch)

Asynchronous kernel dispatching for parallel threads

Benefits: (i) Non-blocking execution, (ii) parallelism + pipeline, (iii) latency hiding and (iv) scalability



Summary of overall ML system architecture

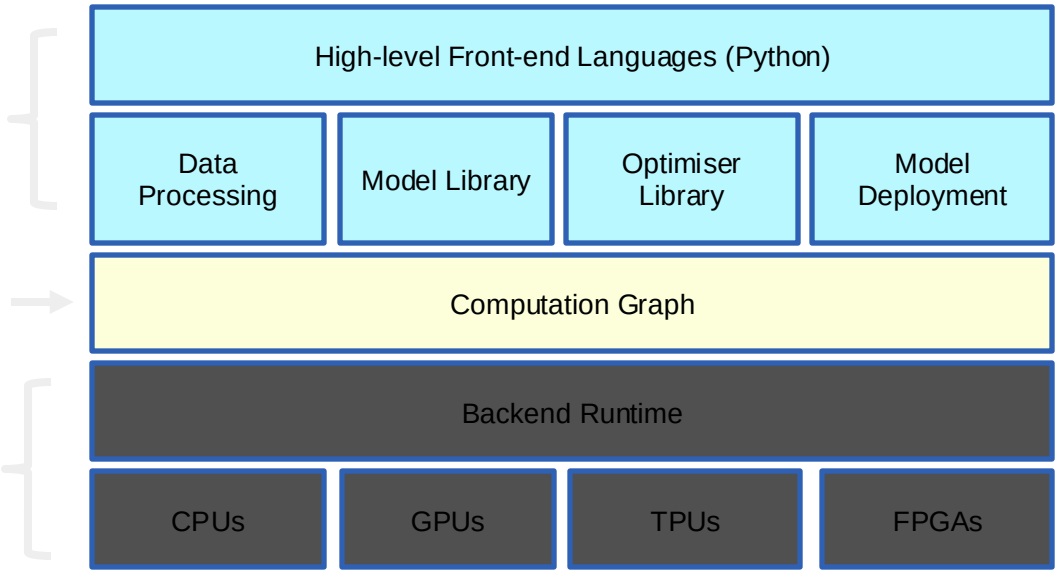
Benefits

- **Simple and flexible** frontend
- **Full life-cycle support**

- **Unified expression** of computation
- **Automatic differentiation**
- Enabling **backend execution**: parallelism, offloading, …

- Asynchronous kernel dispatchers
- Supporting different accelerators

ML Systems Architecture



One more thing

- MLSys Centre for Doctoral Training (CDT) is hiring
- Website: https://mlsystems.uk/#about
  - Simply search ”ML systems CDT” on Google
- The world-first training centre of ML system PhDs
  - Access to world-class supervision and resources (compute, industry, governments).
  - Topics span across the ML system stack: scalability, efficiency, reliability, safety, …
  - Collaborate with experts across the entire ML stack for break-through ideas.
  - Promising career path thanks to the booming demand for AI infrastructure.
  - Join a talented cohort for a life-long learning and growth experience.
- Deadline: 22nd of Jan 2025