CMSC818Q: Special Topics in Cloud Networking and Computing

Machine Learning Frameworks

Instructor: Alan Liu

DEPARTMENT OF
COMPUTER SCIENCE

Class Information

- Presentation sign up this week.

- Review format:
- Each student reviews papers from top conferences or journals. Submit reviews before the class in four sections, including summary, paper strengths paper weaknesses, and detailed comments.

Machine Learning Systems

Researcher

ResNet ....

Transformer

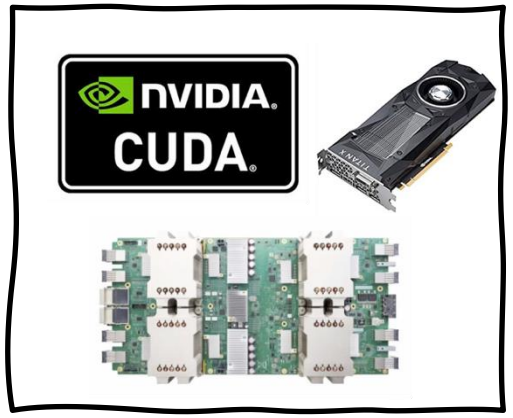
ML Research

100 lines of python A few hours

System Abstractions

Systems (ML Frameworks)

Data

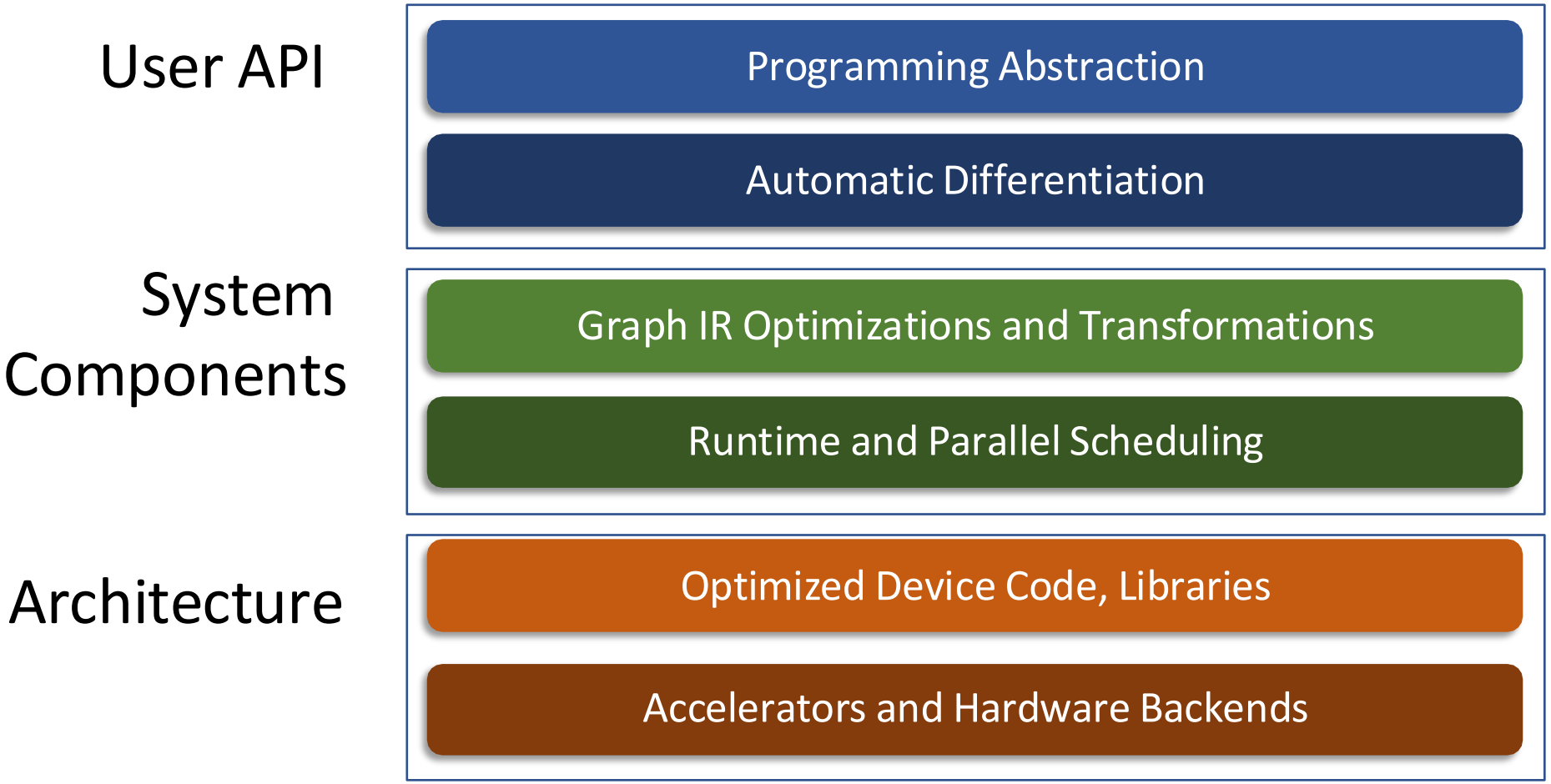


Compute

Machine Learning Frameworks

We won't focus on a specific one, but will discuss the common and useful elements

A Typical Deep Learning System Stack



A Typical Deep Learning System Stack

Transformers ....

Models, Interfaces

AI/ML Research

Programming Abstraction

Automatic Differentiation

Graph IR Optimizations and Transformations

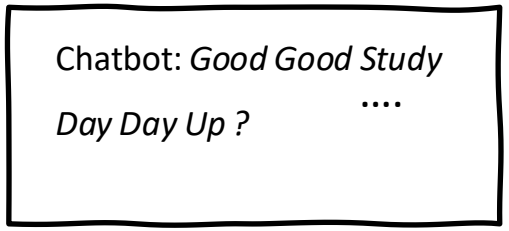
Runtime and Parallel Scheduling

Systems Research

Optimized Device Code, Libraries

Accelerators and Hardware Backends

Interesting Things to Cover in this Semester



Distributed Systems Topics

- Distributed training and networking-collective communication
- LLM inference and serving systems, and opportunities?
- Analysis of system bottlenecks

AI + Systems Topics

- Post-training techniques (Fine tuning variants, RL), and challenges?
- Trendy techniques that improve model inference perf.
- Agent system architectures.

Quick Recap: Elements of Machine Learning

Model

$$x_i = \begin{bmatrix} \text{feature}_0 \\ \text{feature}_1 \\ \dots \\ \text{feaure}_m \end{bmatrix} \quad \hat{y}_i = \frac{1}{1 + \exp\left(-w^T x_i\right)}$$

Objective

$$L(w) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \lambda \|w\|^2$$

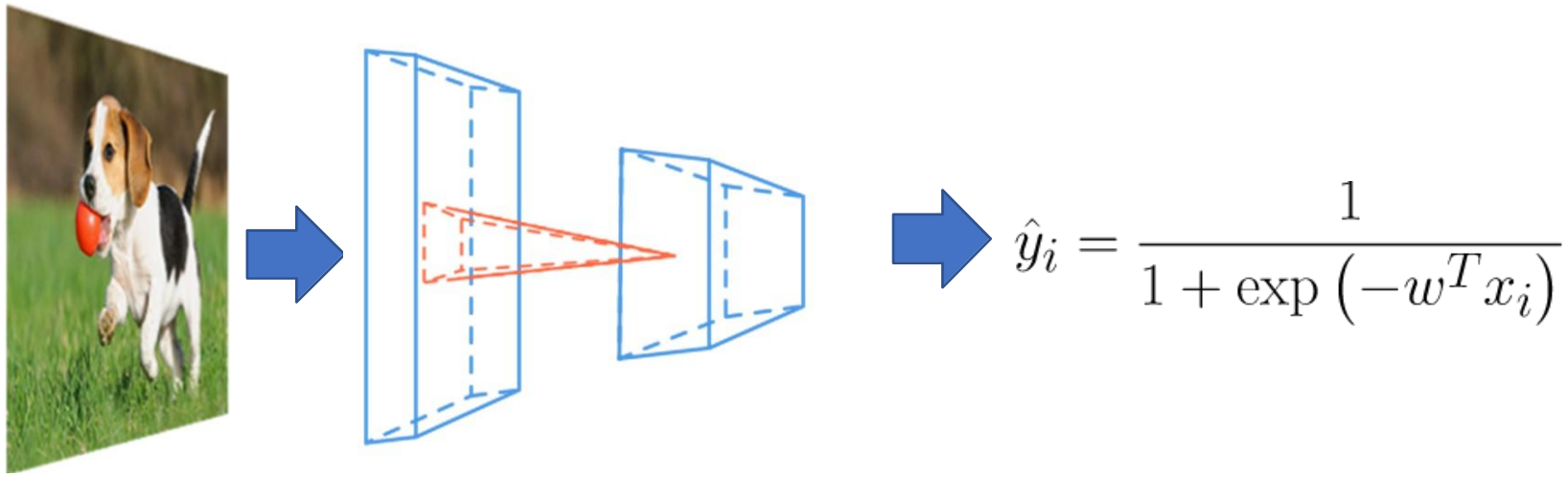
Training (Optimization)

$$w \leftarrow w - \eta \nabla_w L(w)$$

Quick Recap: Deep Learning

Compositional Model

data layer1 layer2 predictor



$$\hat{y}_i = \frac{1}{1 + \exp\left(-w^T x_i\right)}$$

End to end training

Ingredients of a Deep Learning

- Model and architecture

- Objective function and training techniques
  - Which feedback should be used to guide the learning?
  - Supervised, self-supervised, RL, adversarial learning

- Regularization, normalization and initialization (coupled with modeling)
  - Batch norm, dropout, Xavier

- Get good amount of data

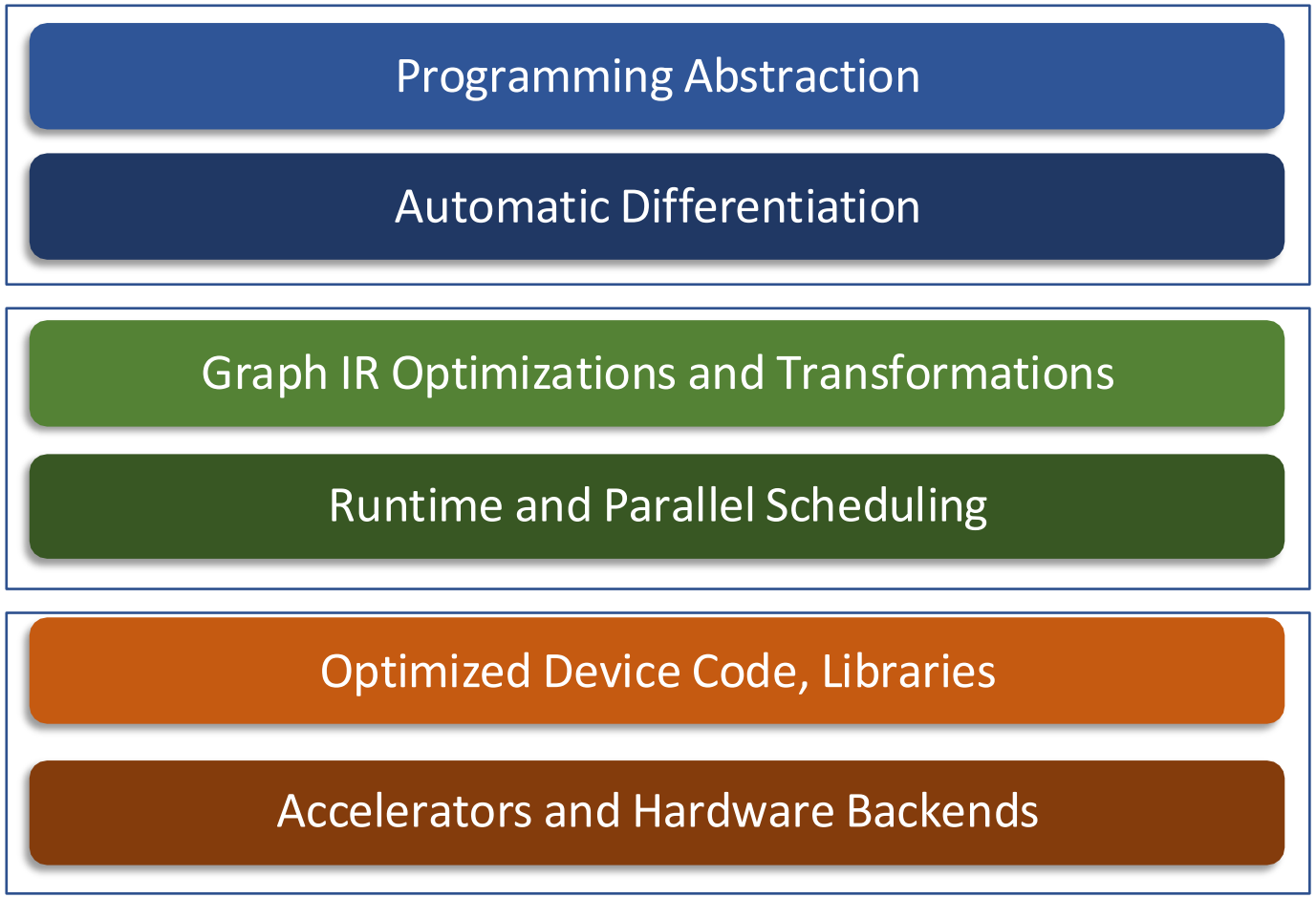
Ingredients of a Deep Learning

- Model and architecture
- Objective function and training techniques
  - Which feedback should be used to guide the learning?
  - Supervised, self-supervised, RL, adversarial learning
- Regularization, normalization and initialization (coupled with modeling)
  - Batch norm, dropout, Xavier
- Get good amount of data

Discussion how can these ingredients affect the system design of ML frameworks

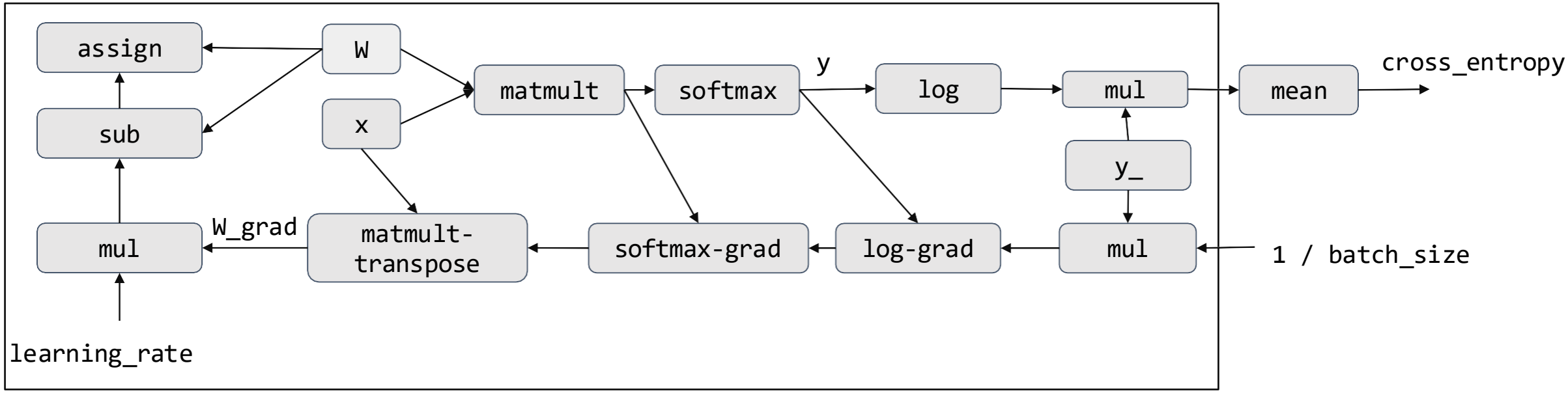
A Typical Deep Learning System Stack

System Components



Computation Graph Optimization

- E.g. Deadcode elimination
- Memory planning and optimization
- What other possible optimization can we do given a computational graph?

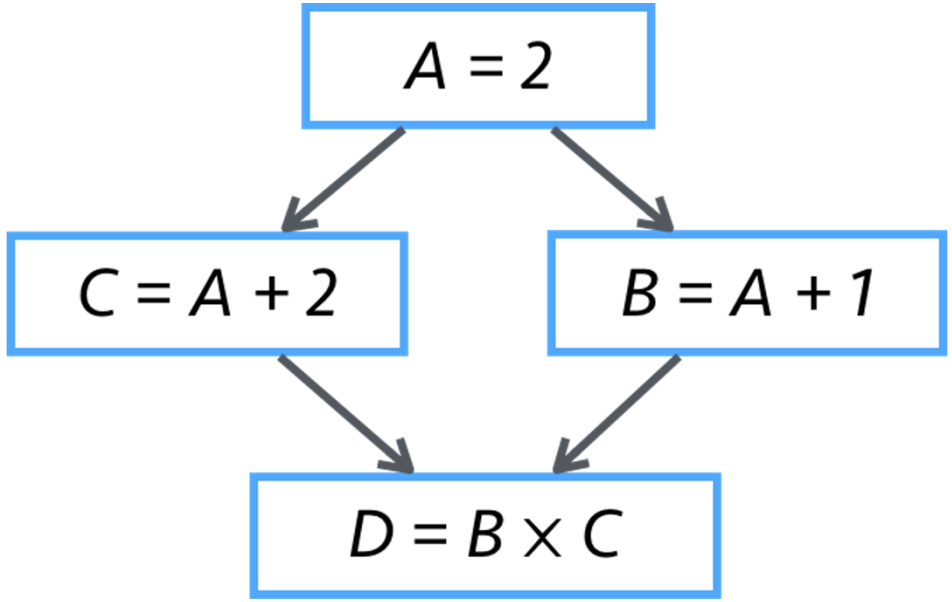


Parallel Scheduling

- Code need to run parallel on multiple devices and worker threads
- Detect and schedule parallelizable patterns

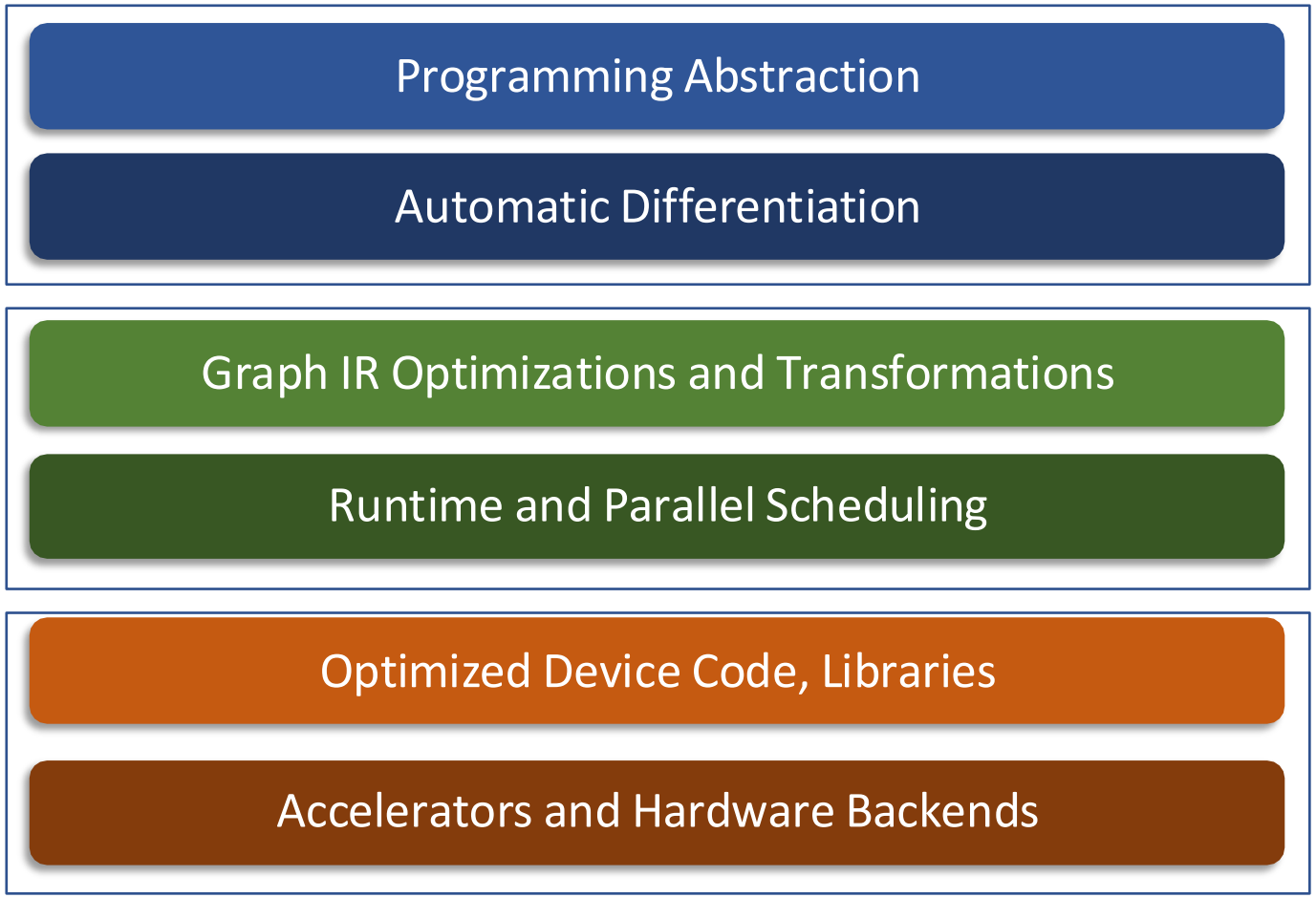
MXNet Example

```
>>> import mxnet as mx
>>> A = mx.nd.ones((2,2)) *2
>>> C = A + 2
>>> B = A + 1
>>> D = B * C
```



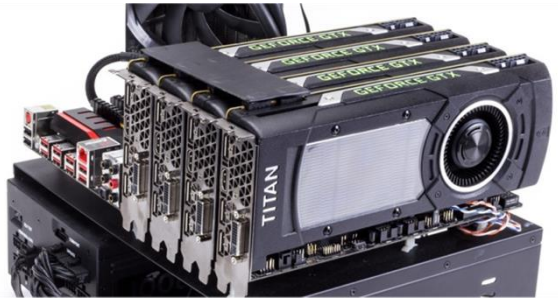
A Typical Deep Learning System Stack

Architecture

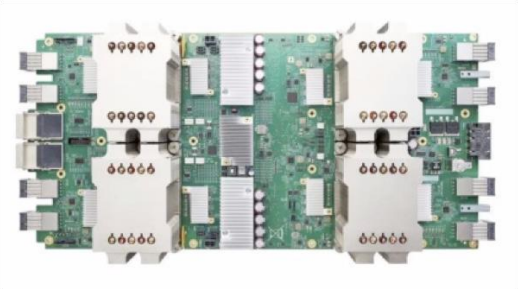


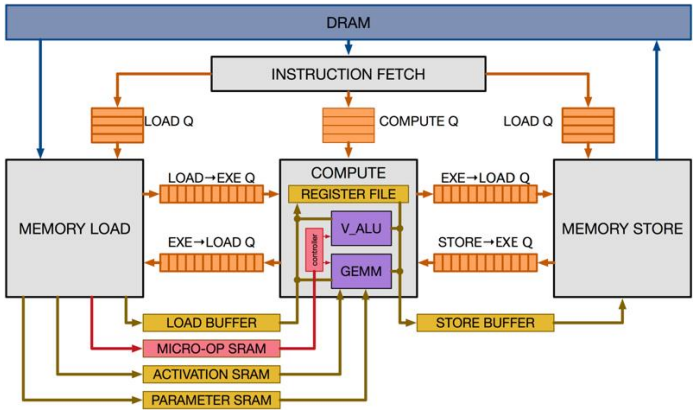
GPU Acceleration

- Most existing deep learning programs runs on GPUs
- Modern GPU have Teraflops of computing power

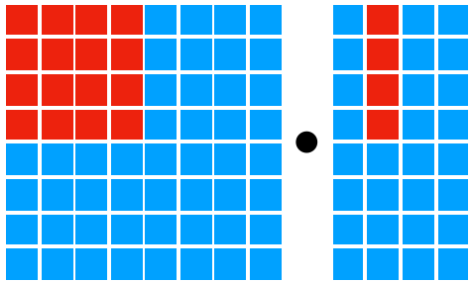


Specialized Accelerators



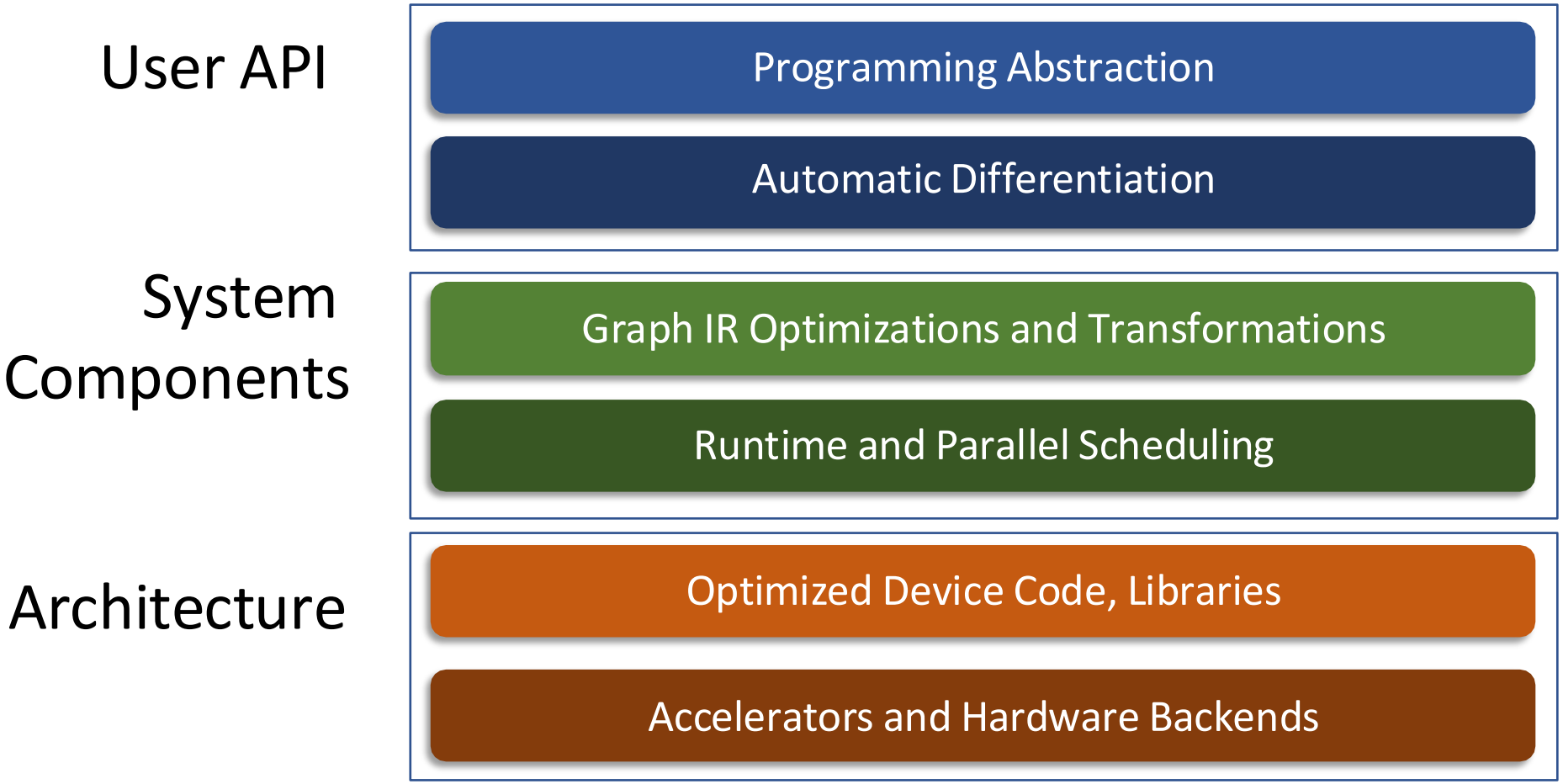


Tensor
Compute Primitives



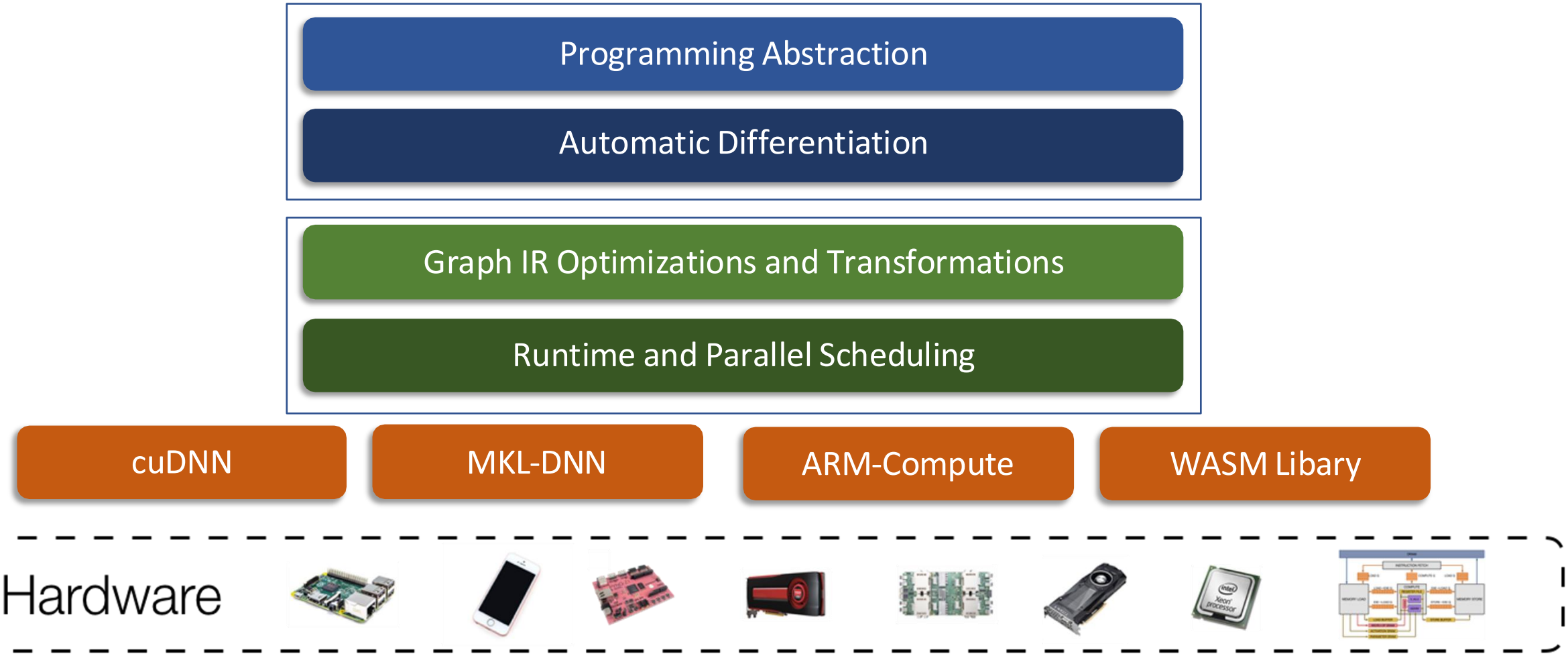
Explicitly Managed
Memory Subsystem

A Typical Deep Learning System Stack

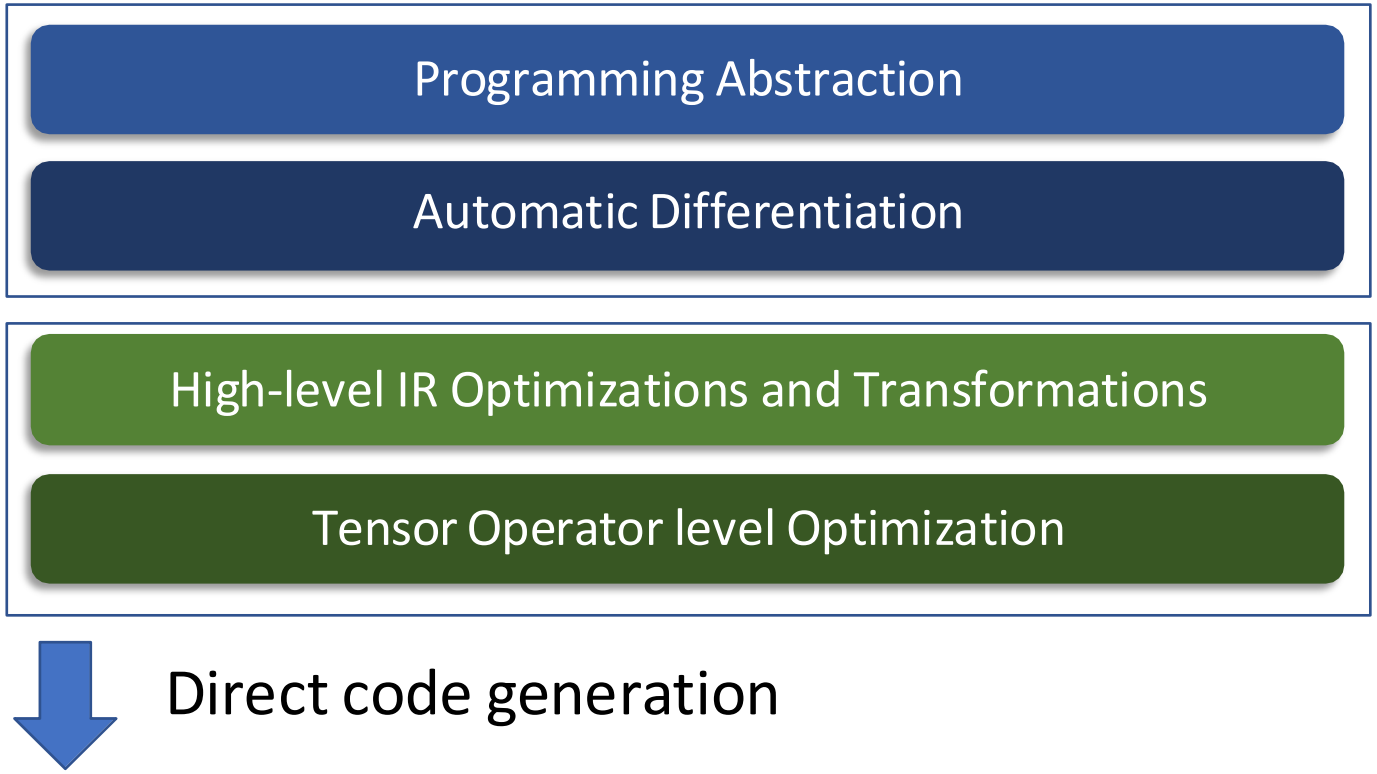


Not a comprehensive list of elements, the systems are still rapidly evolving :)

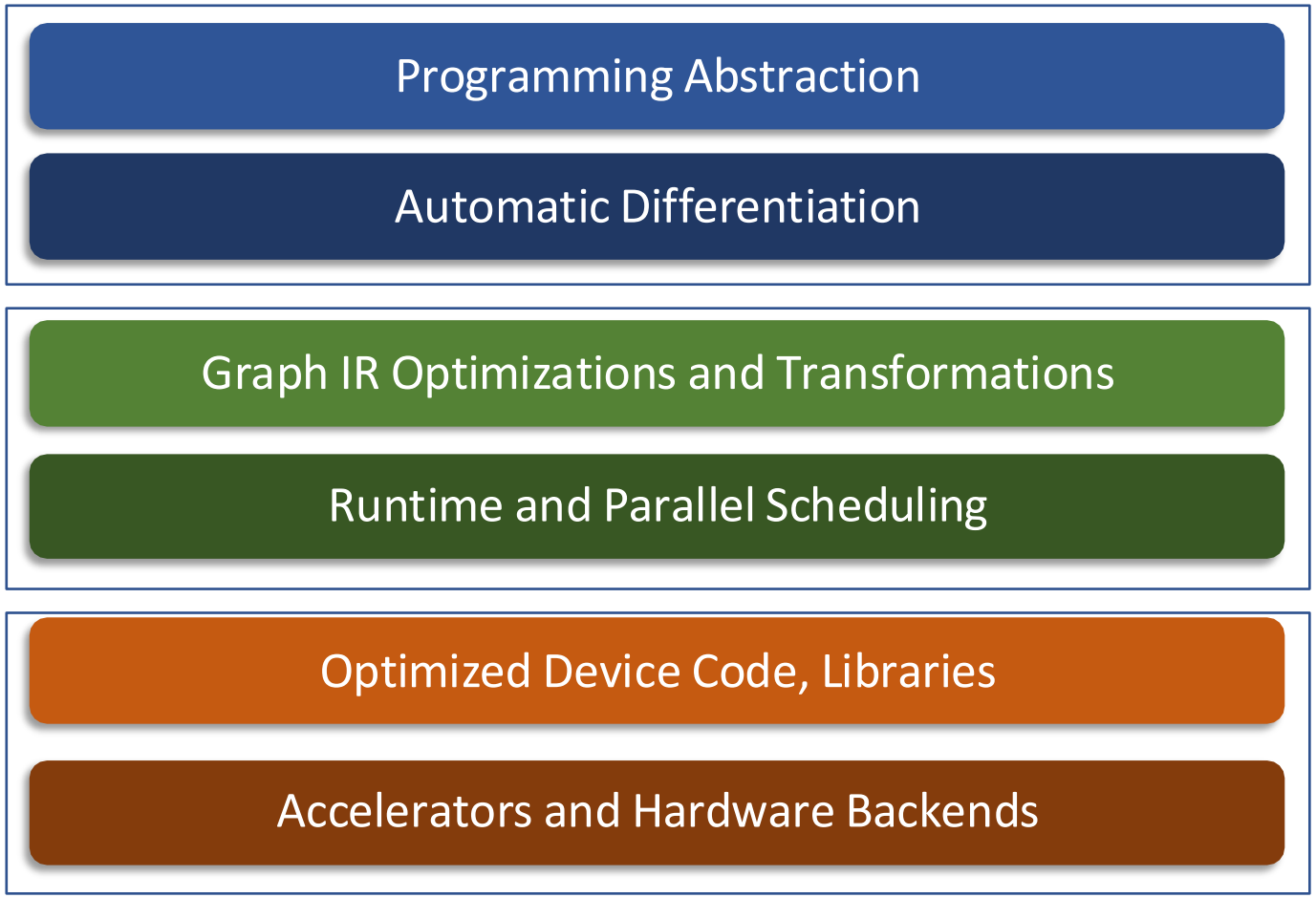
Each Hardware backend requires a software stack



Compiler Based Approach



Questions


Programming Abstraction
Automatic Differentiation
Graph IR Optimizations and Transformations
Runtime and Parallel Scheduling
Optimized Device Code, Libraries
Accelerators and Hardware Backends

A Typical PyTorch Workflow