# Machine Learning Compilation

# Introduction

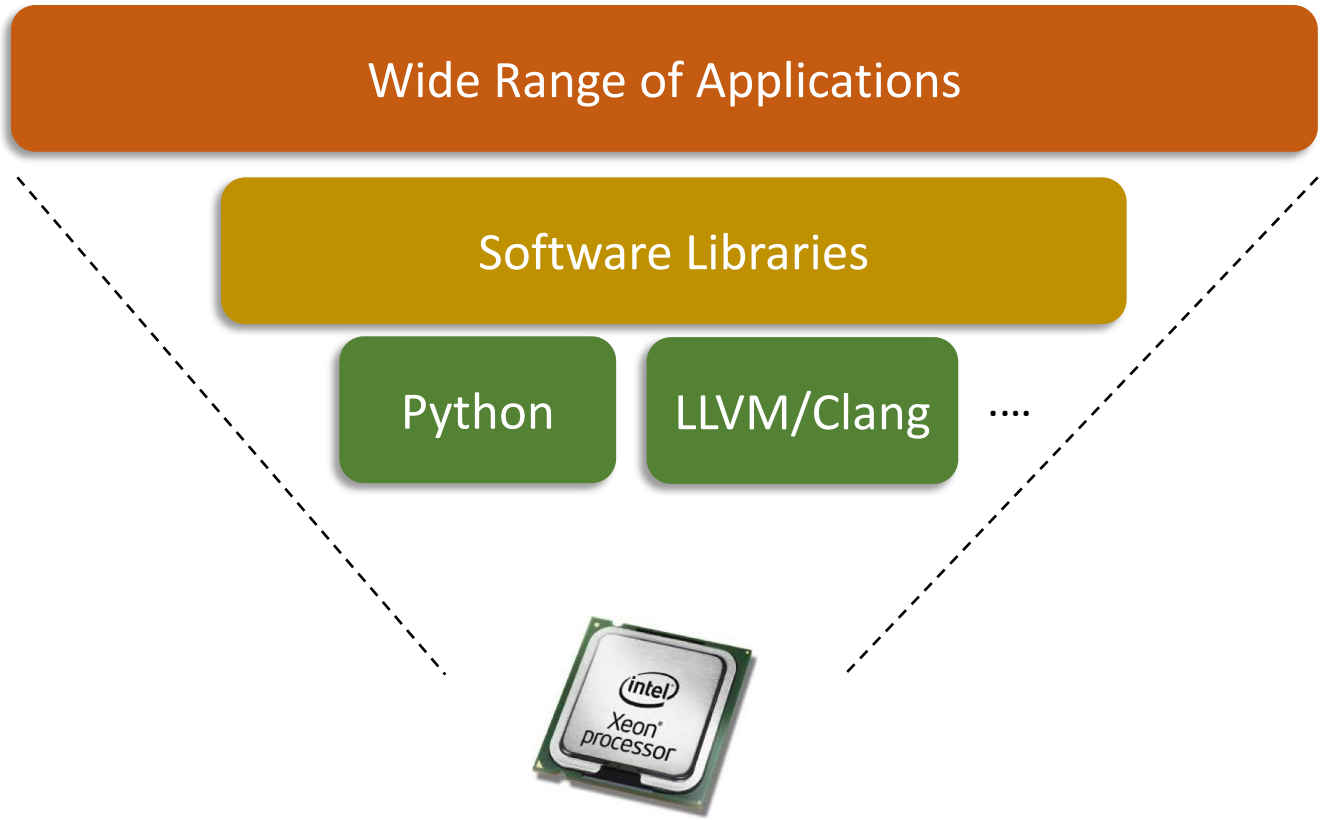
Tianqi Chen


# Outline

Why study machine learning compilation

Key elements in machine learning compilation

# Software Landscape




Broad coverage.

Compute performance is less critical.

Engineers handles most optimizations

# AI Software Landscape

ML Models

Transformer, ResNet, LSTM ...

Data

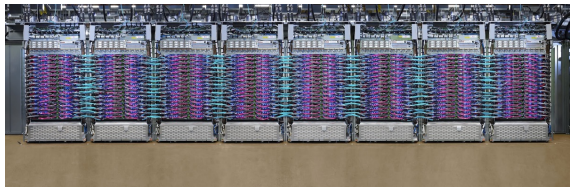
IMAGENET ... WIKIPEDIA The Free Encyclopedia

Diverse and fast evolving models

Big data

Specialized compute acceleration



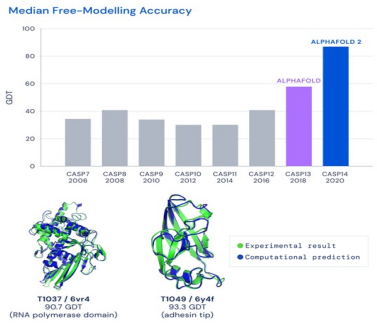



Compute

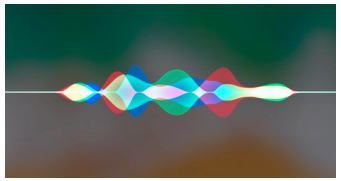
# Machine Learning Deployment Problem

## Intelligent Applications

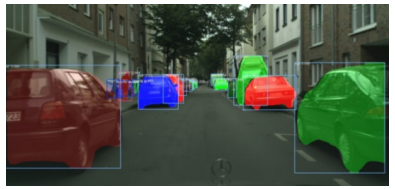
Protein folding




NLP and Speech



Vision

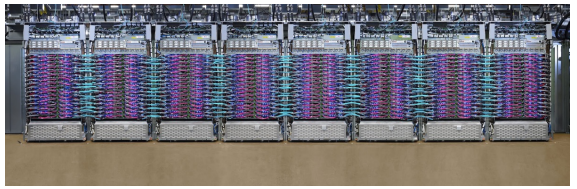


**Gap**

A lot of heavy lifting involved to bring intelligent applications to deployment environments.

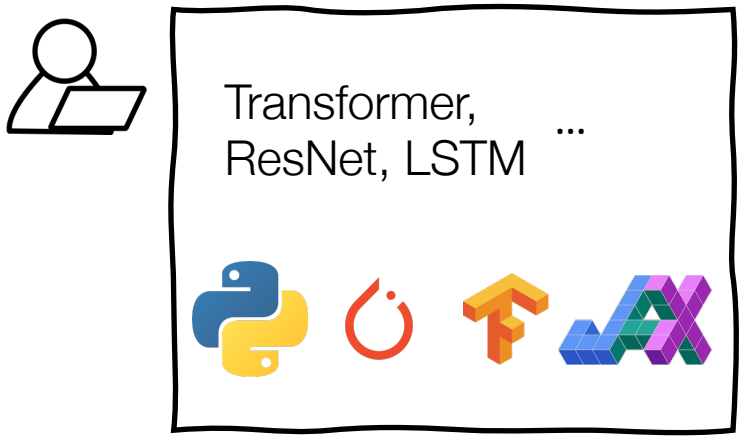
Factors include: hardware(ARM, x86, RISCV),operation system, container execution environment, runtime library variants, accelerator involved…

## Deployment Environments



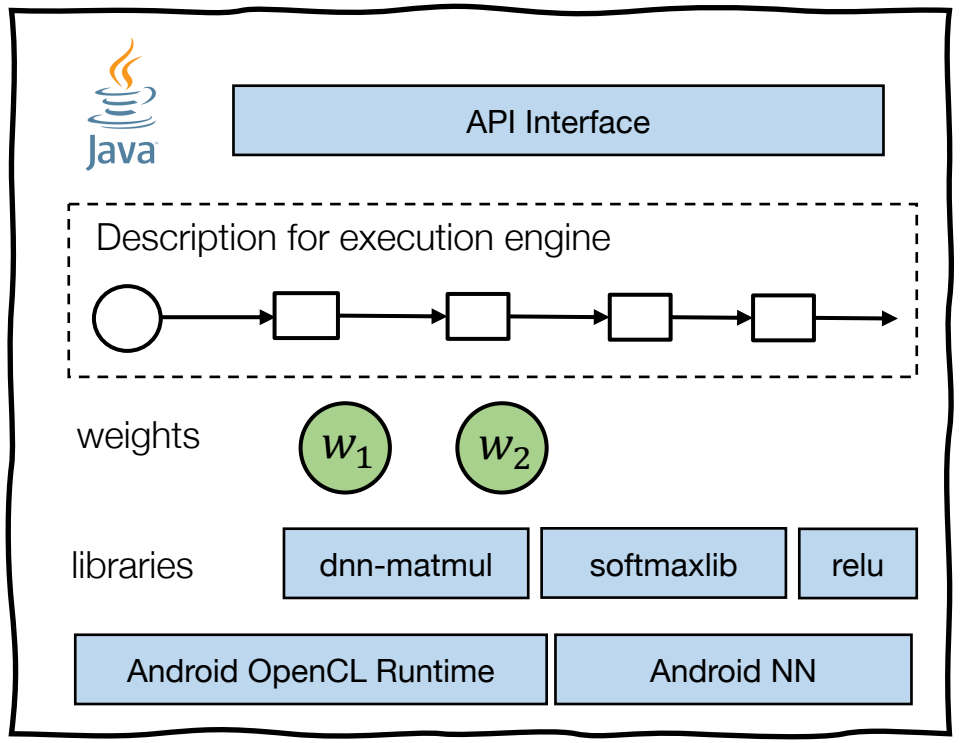
# Machine Learning Compilation

Development Form




**MLC Process**

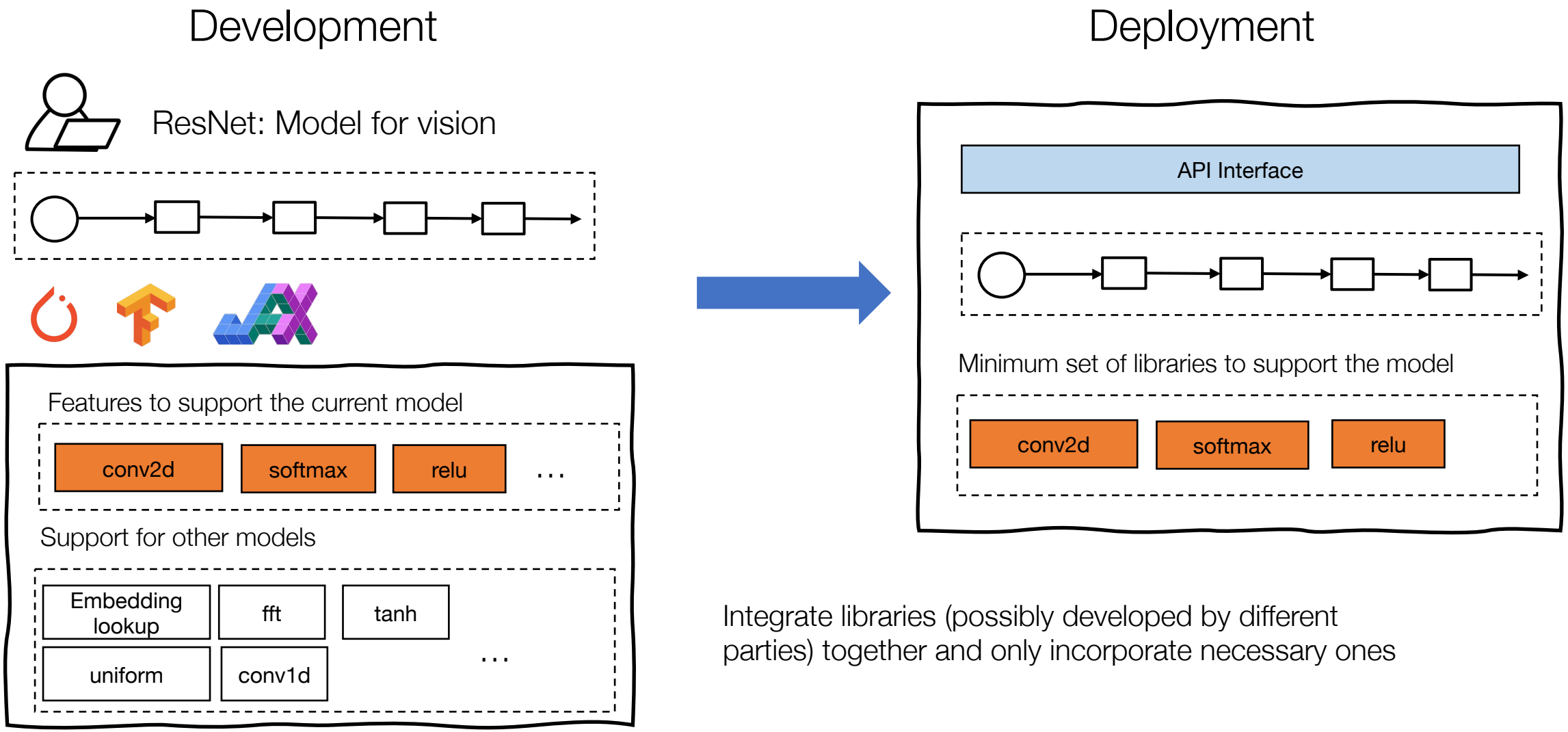
Deployment Form




Machine learning compilation (MLC) is the process to transform and optimize machine learning execution from its development form to deployment form.

An example instance of deployment form

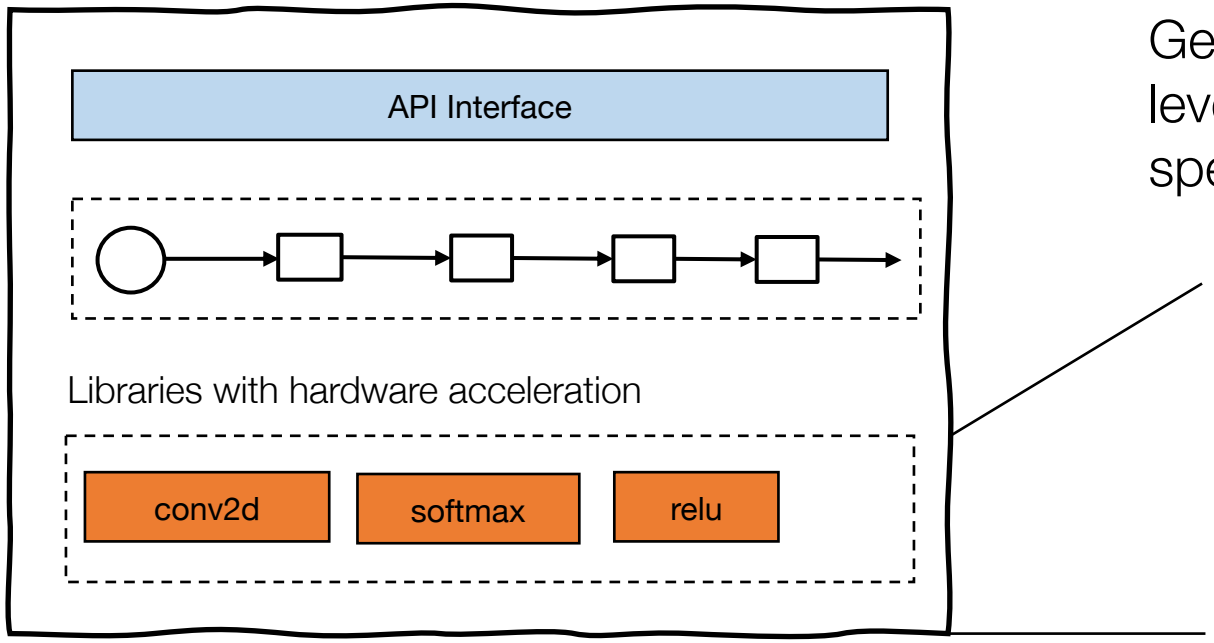
# MLC Goal: Integration and Dependency Minimization



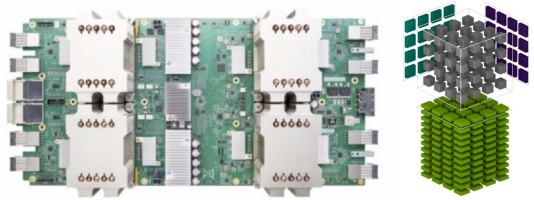

Integrate libraries (possibly developed by different parties) together and only incorporate necessary ones

# MLC Goal: Leverage Hardware Native Acceleration

Deployment




Generate deployment forms that offers leverages native accelerations from specialized hardware backends



# MLC Goal: Optimization in General

There are many equivalent ways to run the same model execution. The common theme of MLC is optimization in different forms:

Minimize memory usage.

Improve execution efficiency.

Scaling to multiple heterogeneous nodes.

# Machine Learning Compilation: Remarks

The term “compilation” make an analogy to traditional compilation (e.g. GCC). However, the techniques involved can be quite different for the two problems.

The process does not have to involve code generation – the deployment form can simply be a model description and an engine that execute it natively.

The development form and deployment form can be the same (in terms of framework interface).

Deployment form also applies to training besides inference needs (e.g., on-device intelligence for privacy reasons, scaling out).

# Reason #1 to Study MLC: Build ML Deployment Solutions

A lot of complexities are involved in bringing machine learning models to production or incorporating novel model customizations.

Machine learning compilation gives us a set of tools to solve problems such as memory size reduction, efficiency optimization, and dependency minimization

# Reason #2 : In-depth Understanding of Existing Frameworks

Machine learning compilation techniques are becoming increasingly relevant for machine learning frameworks.

Understanding the behind-scene techniques provides us super power to build models that can be co-optimized by the underlying techniques and enable us to push novel model customizations with good system support.

# Reason #3 : Build Software Stack for Emerging Hardware

We are at a golden age of hardware accelerations for machine learning, with waves of environments and innovations coming up.

Machine learning compilation provides tools to build software stacks that keeps up with new hardware acceleration features and model developments.

# Reason #4 : Machine Learning Compilation is Fun!

With the set of modern machine learning compilation tools, we can get into stages of machine learning model from high-level, code optimizations, to bare metal.

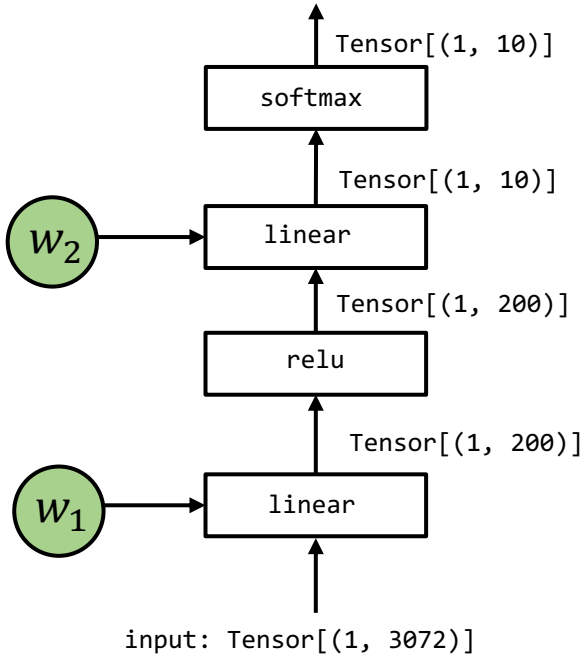
It is really fun to get end to end understanding of what is happening here and use them to solve our problems.

# Outline

Why study machine learning compilation

Key elements in machine learning compilation

# Key Elements in Machine Learning Compilation

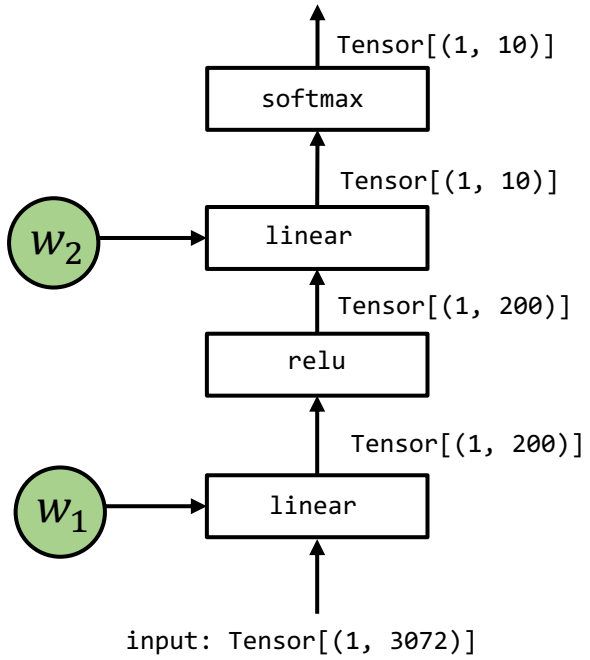



**Tensor** multi-dimensional array that stores the input, output and intermediate results of model executions.

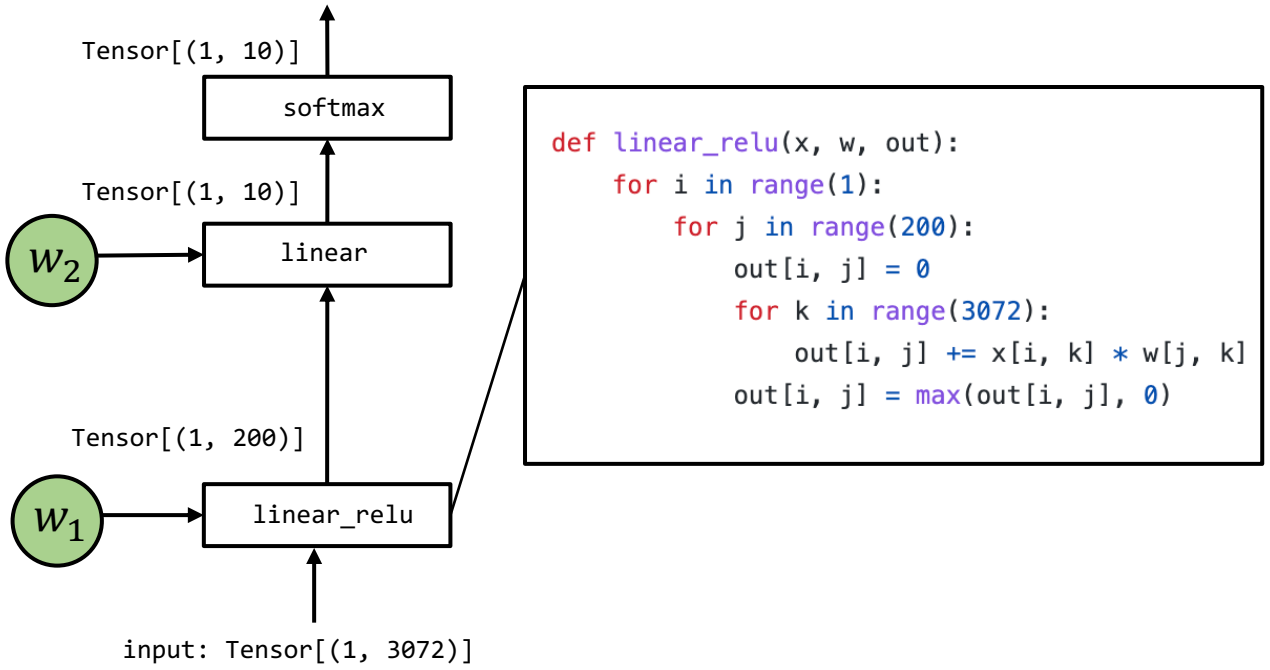
**Tensor Functions** that encodes computations among the input/output. Note that a tensor function can contain multiple operations

# Example Compilation Process

Development



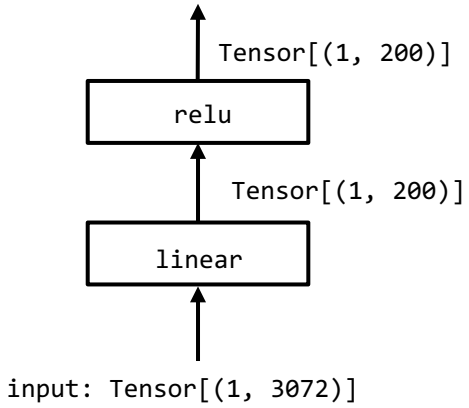

Deployment

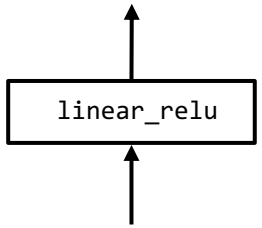



In this particular example, two tensor functions are folded into one (linear-relu). With a specialized implementation (in reality, they will be implemented using low-level primitives).

# Abstraction and Implementation

**Abstraction** refers to different ways to represent the same system interface (tensor function)



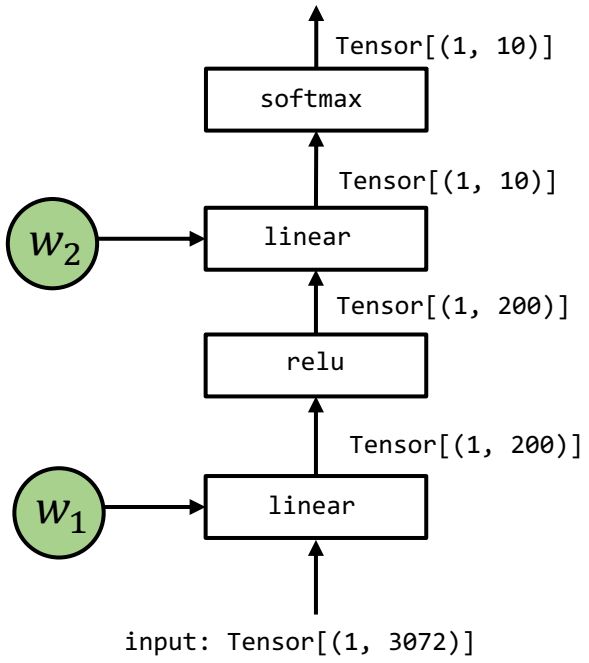




```
def linear_relu(x, w, out):
    for i in range(1):
        for j in range(200):
            out[i, j] = 0
            for k in range(3072):
                out[i, j] += x[i, k] * w[j, k]
            out[i, j] = max(out[i, j], 0)
```

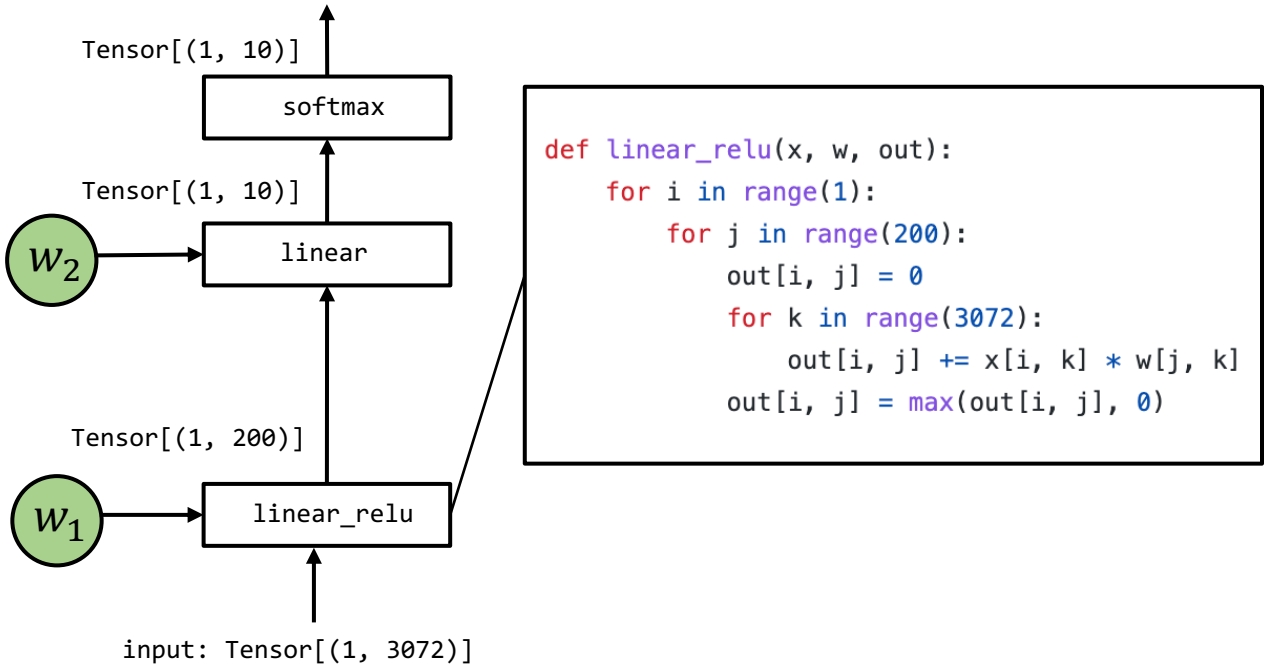
Three abstraction ways to represent the same tensor function (linear_relu), each providing a different level of details. In practice,  we usually say that the more specialized version is an **implementation** of higher-level abstraction.

# MLC as Tensor Function Transformation (with different abstractions)

Development



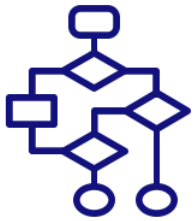

Deployment




Most MLC process can be viewed as transformation among tensor functions (that can be represented with different abstractions).

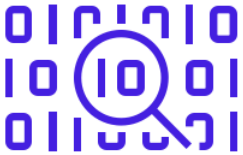
# Four Categories of Abstractions We will Visit Later

## Computational Graphs



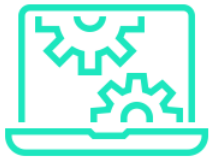
Computational graph and its extensions enable high level program rewriting and optimization.

## Tensor Programs



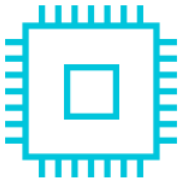
Tensor program abstractions focus on loop and layout transformation for fused operators.

## Libraries and Runtimes



Optimizing libraries are built by vendors and engineers to accelerate key operators of interest.

## Hardware Primitives



The hardware builders exposes novel primitives to provide native hardware acceleration.

# Disclaimer

This is the first course for ML compilation (likely) in the world.

The materials and content will likely adjust as we develop the course. Expect some issues in content and assignments.

We are developing this topic together. Let us explore this fun topic as a community.

# Logistics

Lectures: weekly recordings posted on the course webpage.

Code exercise: we will have code exercises for some episodes. They will be posted to Github with self-contained testcases that allow people to try and self-access.

Lecture notes: will be posted to mlc.ai after each lecture.

Discuss forum: links in the course webpage.

# Summary

- Goals of machine learning compilation
  - Integration and dependency minimization
  - Leveraging hardware native acceleration
  - Optimization in general
- Why study ML compilation
  - Build ML deployment solutions.
  - In-depth view of existing ML frameworks.
  - Build up software stack for emerging hardware.
  - Have fun.
- Key elements of ML compilation
  - Tensor and tensor functions.
  - Abstraction and implementation