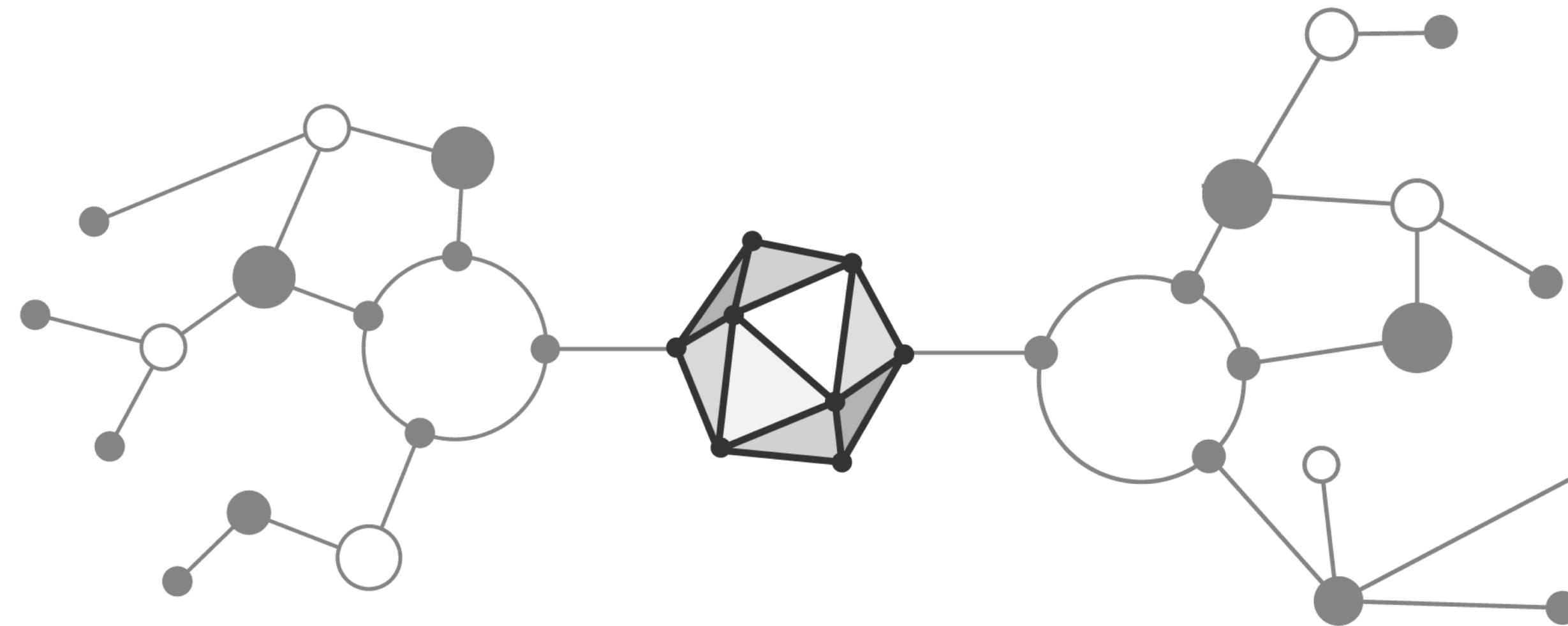


facebook

ONNX

Sarah Bird, Dmytro Dzhulgakov

Facebook



Deep Learning Frameworks



Tensors and Dynamic neural networks in Python with strong GPU acceleration

Flexible Development

- Research-oriented imperative model
- Python flow-control constructs
- Dynamic graph support with autograd

<http://pytorch.org>

Released Jan 18th

100,000+ downloads

950+ community repos

8200+ user posts

245 contributors





Caffe2

A New Lightweight, Modular, and Scalable Deep Learning Framework

**RUN ANYWHERE,
FAST**

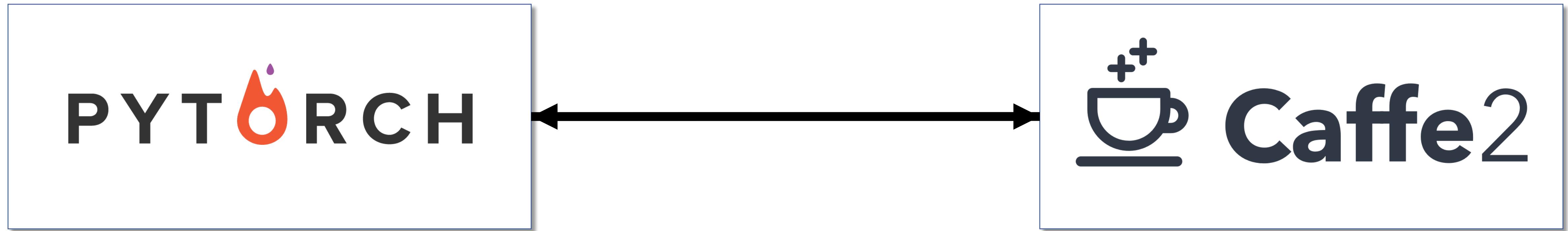
Your favorite deep learning technology,
now from zero to scale, cloud to mobile.

Train ImageNet in 1 hour

Production Powerhouse

- Scalable from small devices to large GPUs in DC
- Strong distributed training support
- Highly optimized mobile device support
- Based on ahead-of-time static graph
 - no interpreter needed in prod

Research to Production



Reimplementation
takes weeks or months

Merge Frameworks?

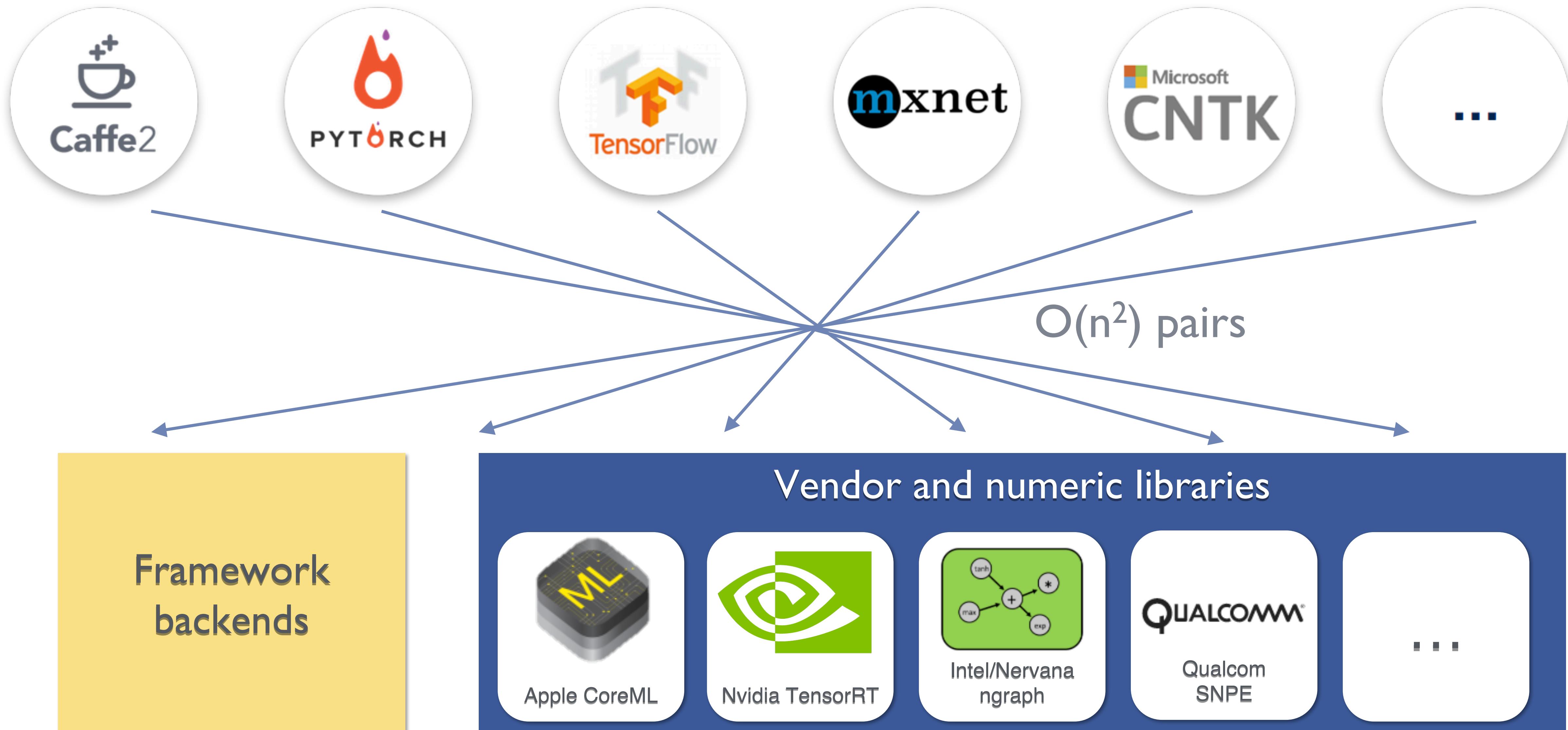


- Model transfer is important, but less common
- Difficult to optimize the tools for all cases
- Separate but interoperable tools is more efficient

Shared Model Format



Deep Learning Frameworks Zoo



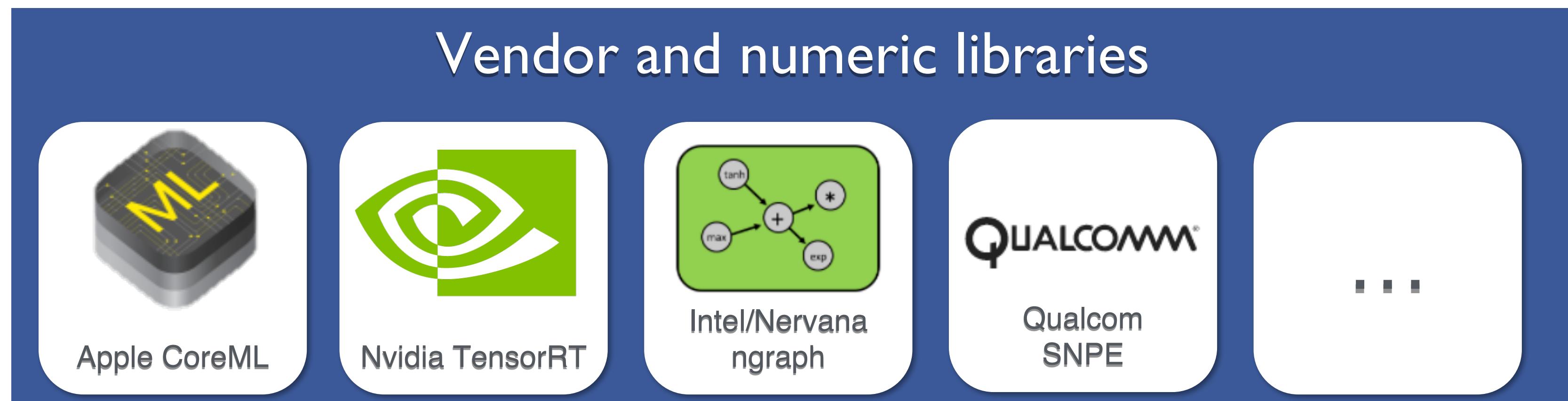
Open Neural Network Exchange



Shared model and operator representation

From $O(n^2)$ to $O(n)$ pairs

Framework
backends



Standard?

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)

SITUATION:
THERE ARE
14 COMPETING
STANDARDS.

14?! RIDICULOUS!
WE NEED TO DEVELOP
ONE UNIVERSAL STANDARD
THAT COVERS EVERYONE'S
USE CASES.



SOON:

SITUATION:
THERE ARE
15 COMPETING
STANDARDS.

Open community

- Framework agnostic
- GitHub from the beginning
- Close partnerships and OSS contributions



Facebook
Open Source

Microsoft



NVIDIA

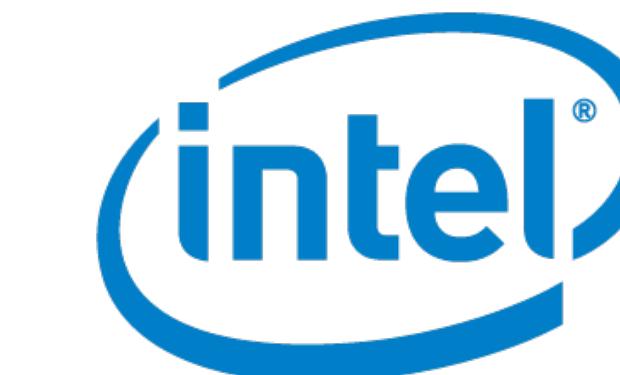


HUAWEI

AMD

arm

QUALCOMM®



Unframeworks

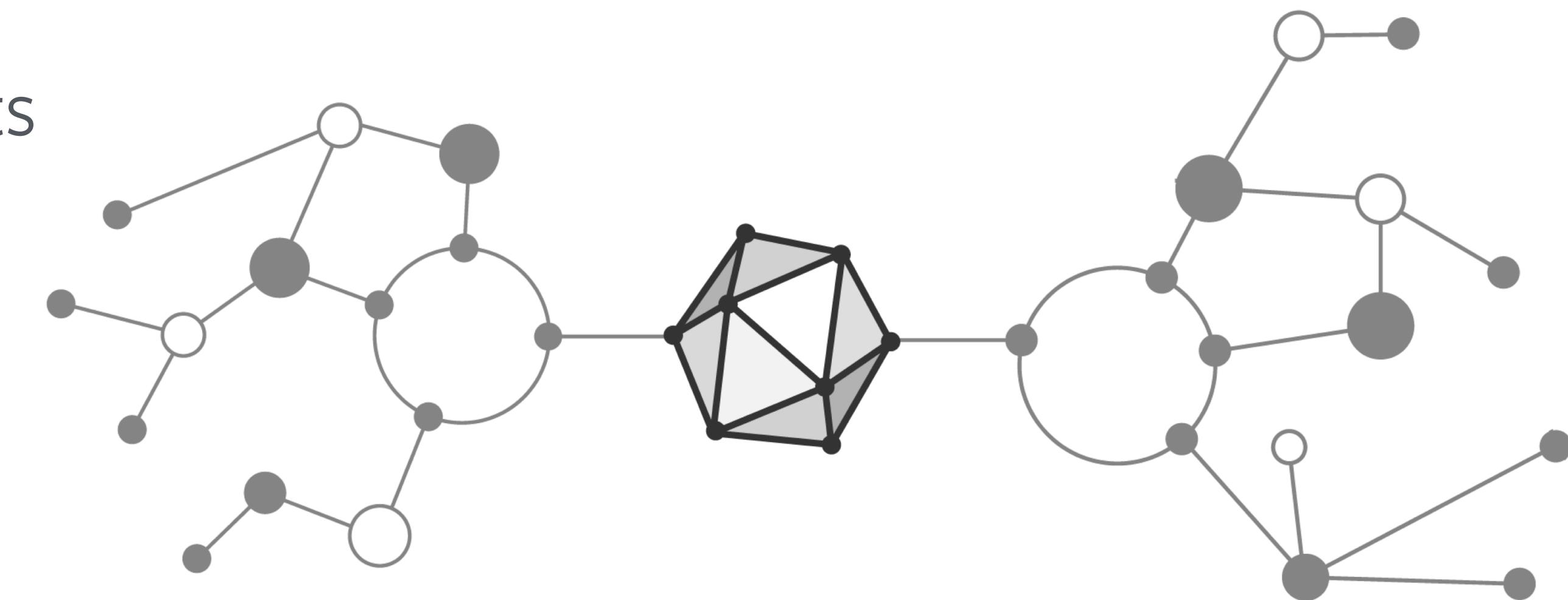
Unframeworks

Vision: Interoperable Tools

- Accelerate research to production
- Developers can use the best combination of tools for them
- Enables more people to contribute

Approach:

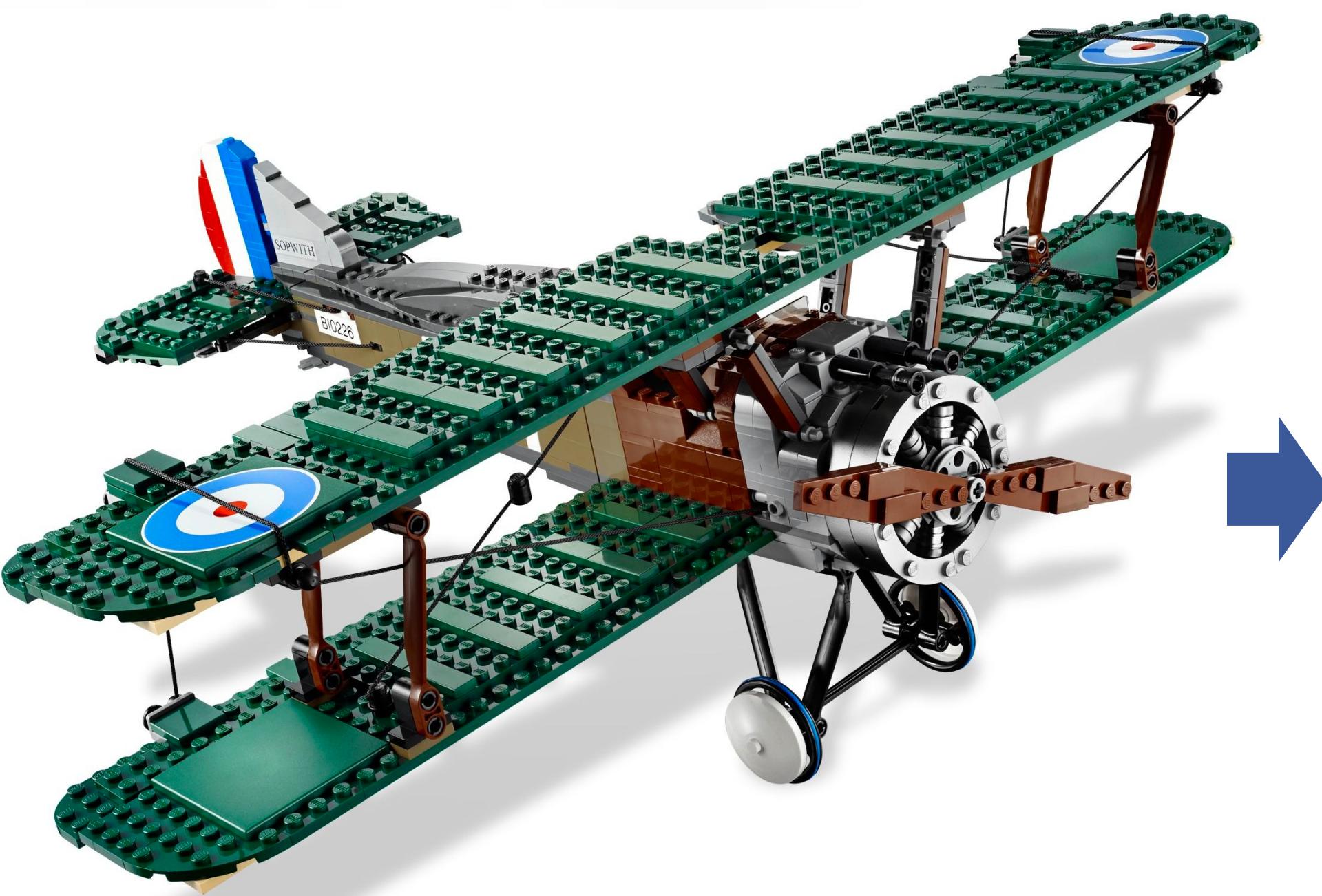
- Split toolchain into smaller components



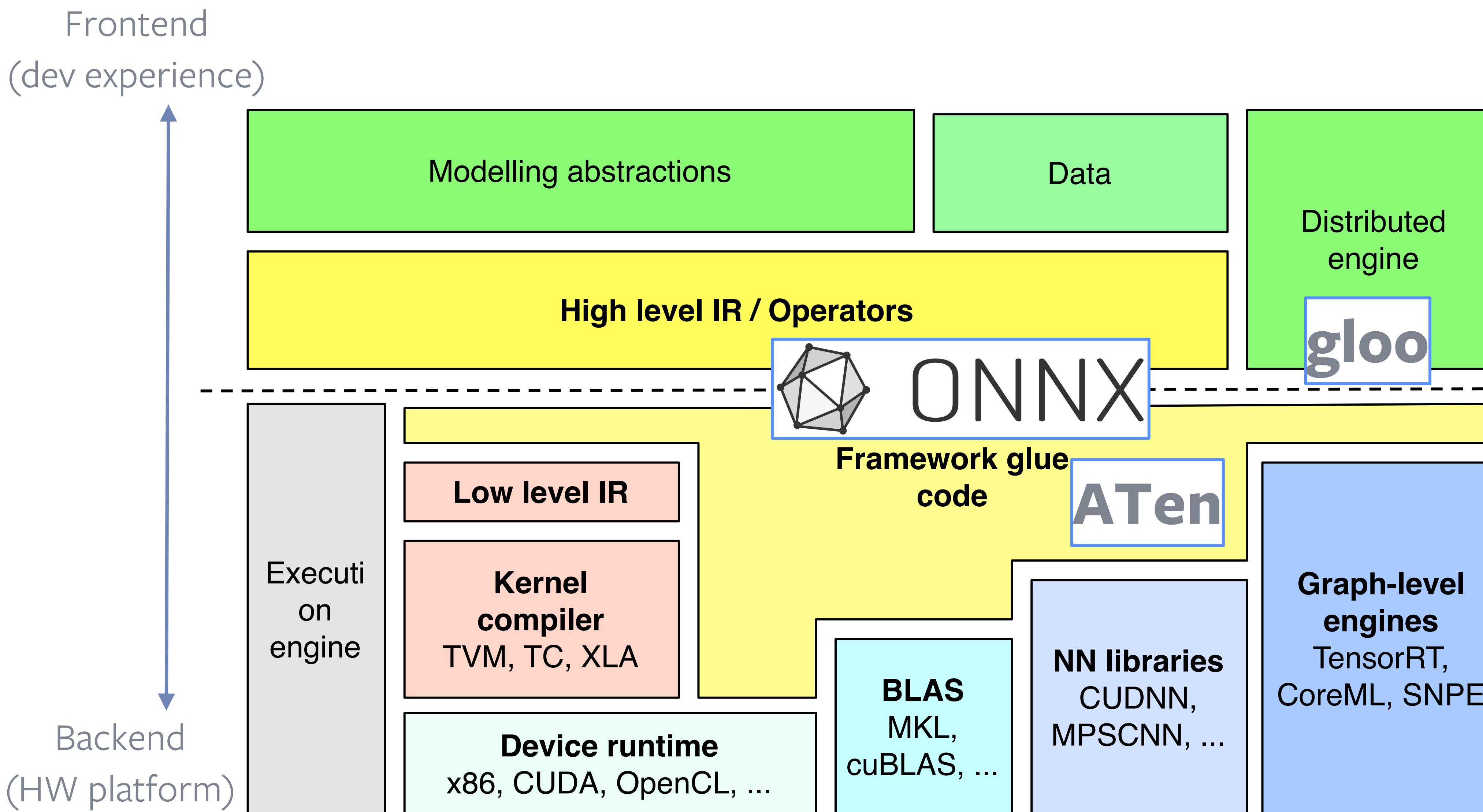


UNIX philosophy for deep learning frameworks

Build reusable components
that work well together
(across frameworks)

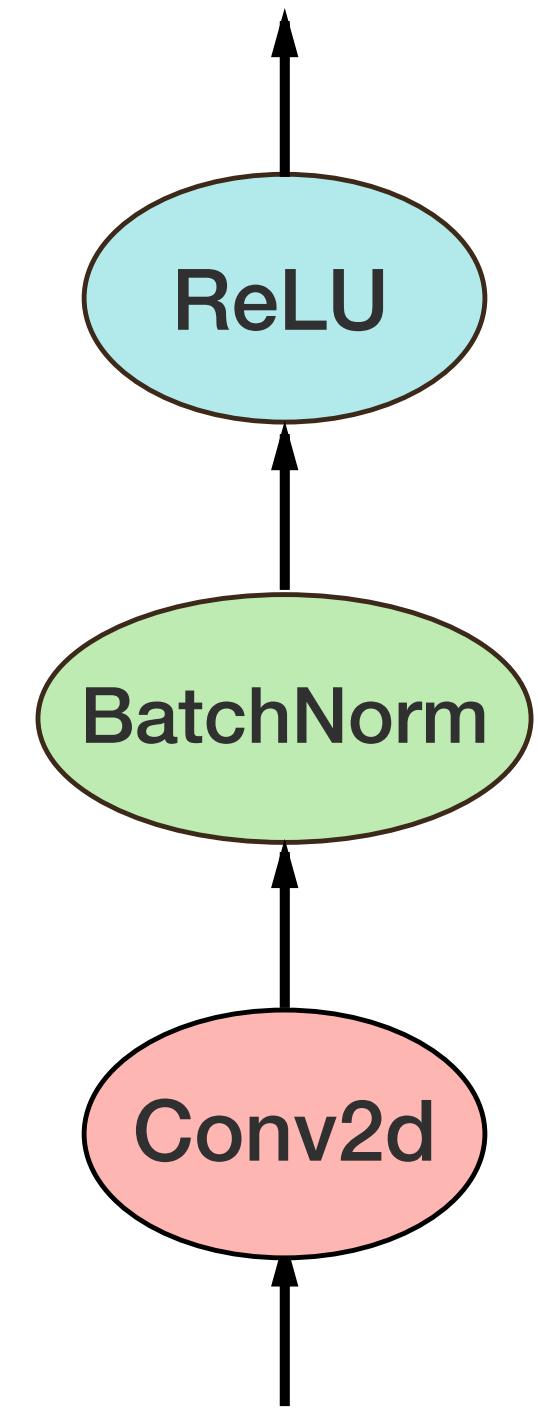


Framework anatomy



ONNX high-level IR

- Initial focus on exchange for inference
- **SSA graph structure, serializable**
 - Support for structured control flow
- **Standard operator definitions**
 - Striking balance on granularity
 - Codified semantics in tests/ref
- **Common optimization passes**



PRelu

PRelu takes input data (Tensor) and slope tensor as input, and produces one output data (Tensor) where the function $f(x) = \text{slope} * x$ for $x < 0$, $f(x) = x$ for $x \geq 0$, is applied to the data tensor elementwise.

Inputs

`x : T`
Input tensor

`Slope : T`
Slope tensor. If `Slope` is of size 1, the value is shared across different channels

Outputs

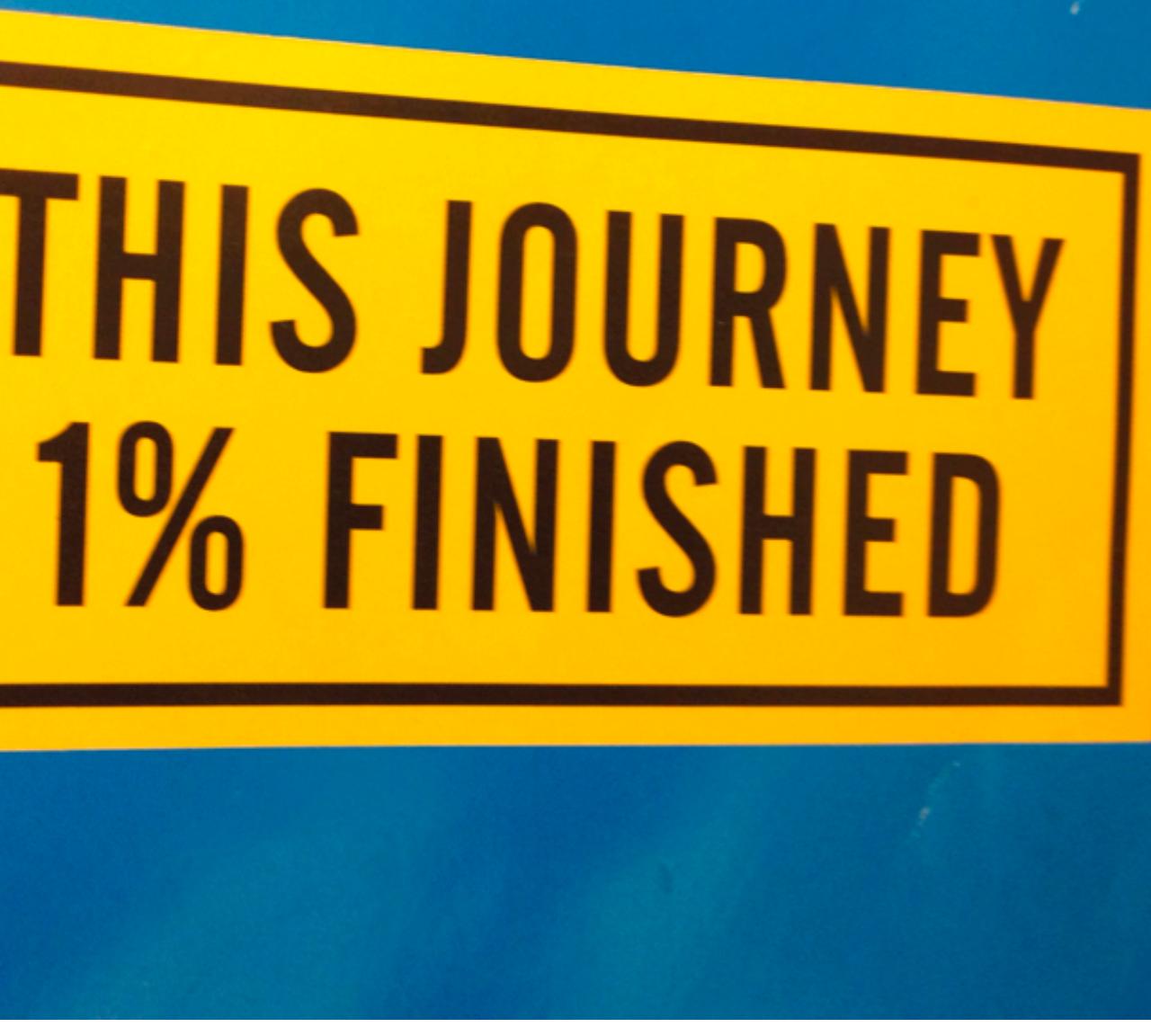
`y : T`
Output tensor

Type Constraints

`T : tensor(float16), tensor(float), tensor(double)`
Constrain input and output types to float tensors.

Current status

- **ONNX IR spec is V1.0**
- Good coverage for **vision** models
- Iterating on:
 - Optimization-friendly RNNs
 - Control Flow
 - More hardware backends



THIS JOURNEY
1% FINISHED

Beyond static graphs:
Capturing dynamic behavior

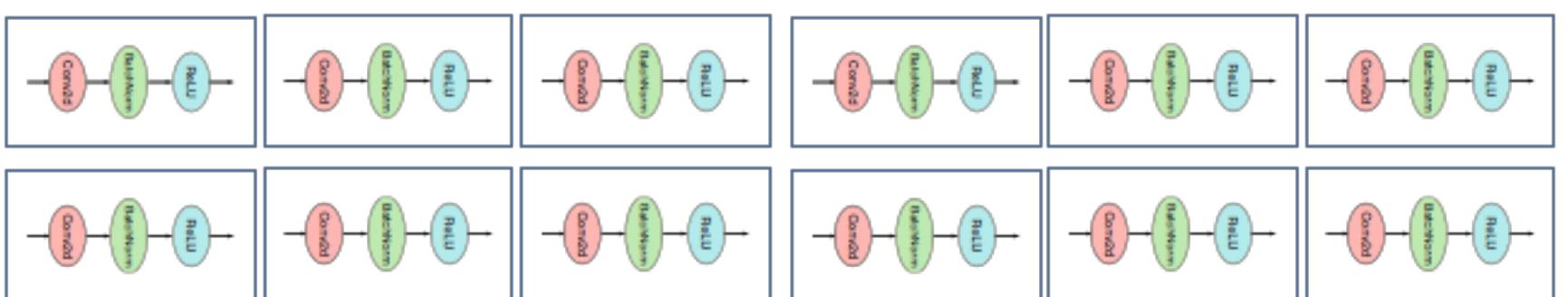
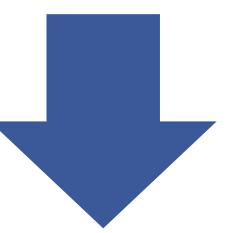
Declarative

vs

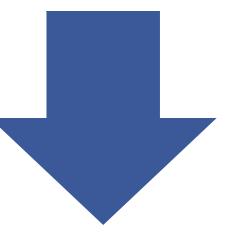
Eager mode

Python script

Building IR
in Python



Python-independent
execution



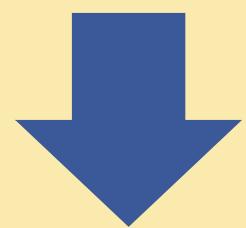
Framework's VM

Operator
implementations

Execution
engine

Python interpreter

Code



Operator
implementations

Regular python extension

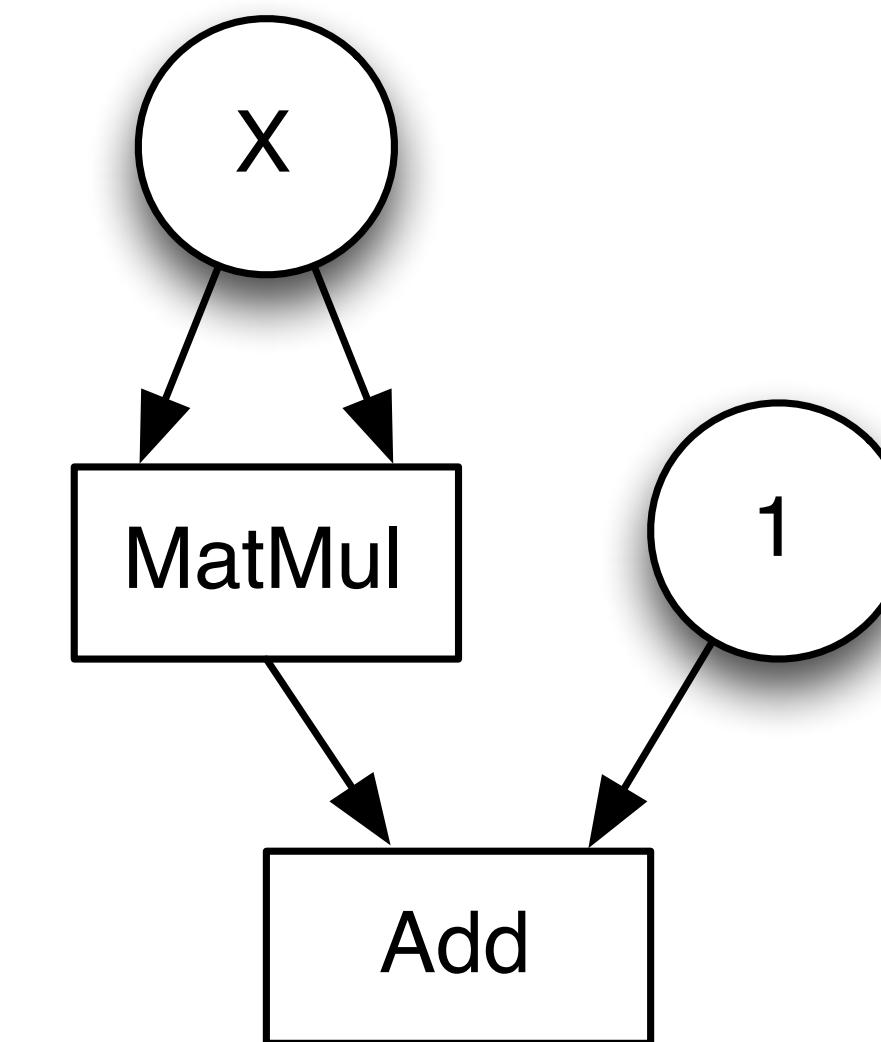


Tracing for static graph

Record which operators were invoked

```
def foo(x):  
    y = x.mm(x)  
    print(y) # still works!  
    return y + 1
```

```
x = torch.Tensor([[1,2],[3,4]])  
foo(x)
```



Enough to cover CNNs and static sections

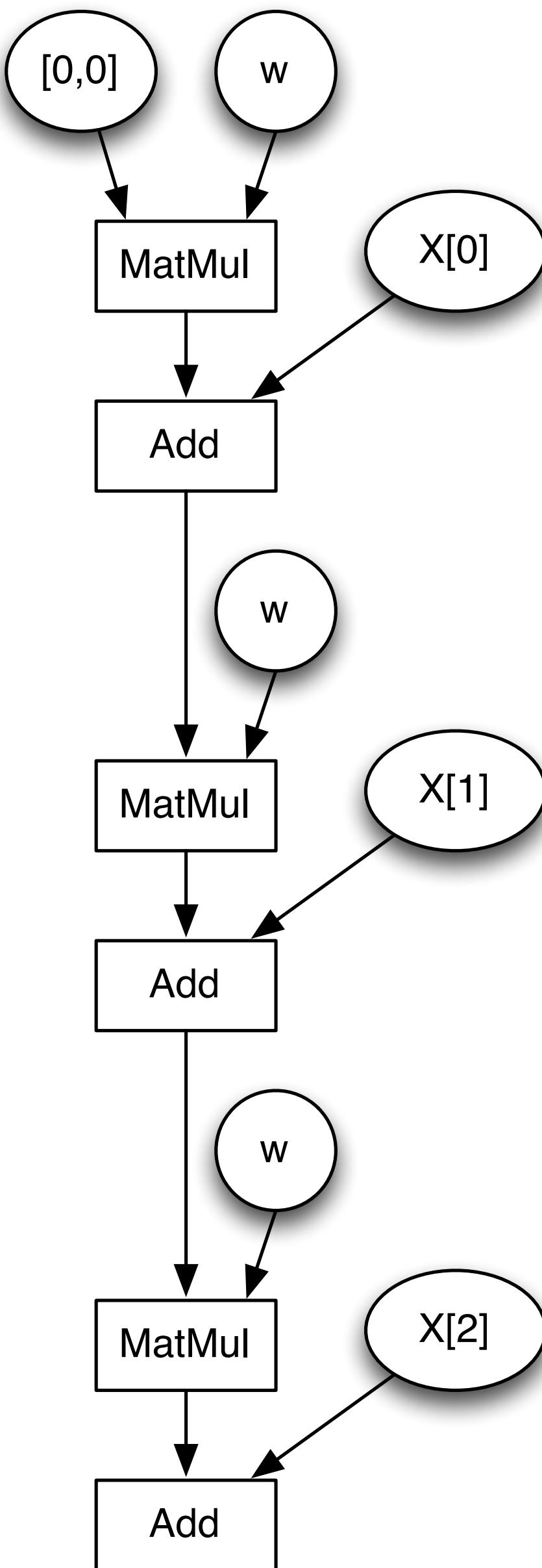
Tracing for dynamic graphs

```
def foo(x, w):
    y = torch.zeros(1, 2)
    for t in x:
        y = y.mm(w) + t
    return y

w = torch.Tensor([[0.5, 0.2], [0.1, 0.4]])
x = torch.Tensor([[1, 2], [3, 4], [5, 6]])
foo(x, w)

x2 = torch.Tensor([[7, 8], [9, 10]])
foo(x2, w)
```

Doesn't do what you want!

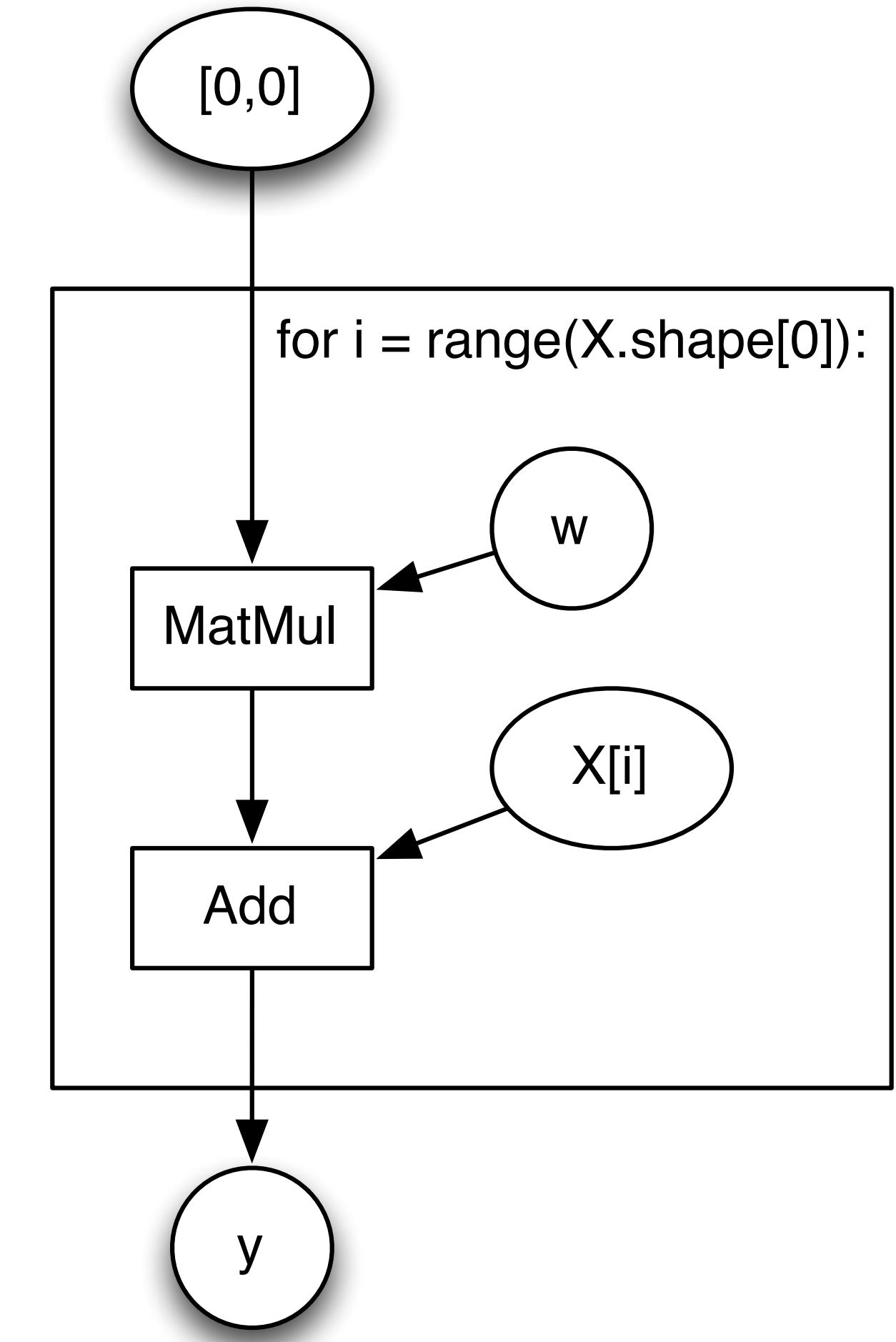


Tracing for dynamic graphs

```
def foo(x, w):
    y = torch.zeros(1, 2)
    for t in x:
        y = y.mm(w) + t
    return y
```

```
w = torch.Tensor([[0.5, 0.2], [0.1, 0.4]])
x = torch.Tensor([[1, 2], [3, 4], [5, 6]])
foo(x, w)

x2 = torch.Tensor([[7, 8], [9, 10]])
foo(x2, w)
```

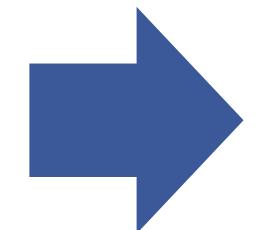


Capture control flow from python?

Approaches for dynamic graphs

- Parse or compile Python (tricky)
- Use special primitives (annoying)

```
for t in x:  
    y = y.mm(w) + t
```



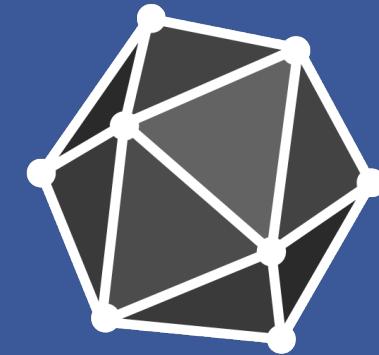
```
lib.For(x, y, lambda y, t:  
        y.mm(w) + t)
```

- Capture common patterns like RNN
- Build DSL for subset of Python
- Make it easy to embed C++ calling back to framework

Putting it together

Capturing dynamic behavior

- Trace static portions
- Minimum rewrites for dynamic parts
- Establish tooling for step-by-step code migration



ONNX

Get Involved!

ONNX is a community project.

<https://onnx.ai>

<https://github.com/onnx>



Facebook
Open Source

Microsoft

facebook