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
500,000+ downloads
2700+ community repos
17,200+ user posts
351 contributors
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

PYTORCH — ONNX — Caffe2
Deep Learning Frameworks Zoo

Framework backends

Vendor and numeric libraries

- Apple CoreML
- Nvidia TensorRT
- Intel/Nervana ngraph
- Qualcomm SNPE

O(n^2) pairs
Open Neural Network Exchange

Shared model and operator representation

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

Vendor and numeric libraries

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. Yeah!

Soon:

Situation: There are 15 competing standards.
Open community

• Framework agnostic
• GitHub from the beginning
• Close partnerships and OSS contributions
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)
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
Current status

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

Caffe2

Python interpreter Code

Operator implementations

Regular python extension

PyTorch
Tracing for static graph

Record which operators were invoked

```python
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
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!
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)

```python
for t in x:
    y = 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 is a community project.

https://onnx.ai
https://github.com/onnx