Accelerating Research to Production with PyTorch 1.0 and ONNX

Marat Dukhan
SOFTWARE ENGINEER
Deep Learning Frameworks

Theano
MXNet
CNTK
Chainer
Tensorflow
Caffe
HIPS Autograd
Dynet
Deep Learning Frameworks

DECLARATIVE

Tensorflow
Caffe
Theano
MXNet
CNTK

IMPERATIVE

PyTorch
Chainer
Dynet
HIPS Autograd
Deep Learning Frameworks

**DECLARATIVE**
- Python script
- Building IR in Python
- Python-independent execution
- Framework’s VM
  - Operator implementations
  - Execution engine

**IMPERATIVE**
- Python interpreter
- Code
- Operator implementations
- Regular python extension

**Deep Learning Frameworks**

- Caffe2
- PyTorch
Production quality

First-class distributed support

Cross-platform availability

Mobile deployment tools
Caffe2 in Production

200T+ predictions / day
Figure 6. ResNet 50 FP32 performance on NVIDIA optimized deep learning frameworks and DGX-1 with Pascal GPUS.

Source: https://devblogs.nvidia.com/parallelforal/training-self-driving-vehicles-challenge-scale

4/30/2017: DGX-1 with Batch Size=64 per GPU. Chainer numbers are preliminary.
Caffe2 on Mobile

PRIVACY

LOCALITY

USER EXP
Mobile Fragmentation

- 20+ CHIPSET VENDORS
- 25+ CPU MICROARCHITECTURES
- 15+ GPU ARCHITECTURES

- TWO MAJOR OPERATING SYSTEMS
  - ANDROID
  - IOS

- THREE MAJOR GRAPHICS APIS
  - OPENGL
  - VULKAN
  - METAL

- TWO MAJOR COMPUTE APIS
  - OPENCL
  - Renderscript
Caffe2

- ARM
- Metal™/MPS CNN
- Qualcomm Snapdragon NPE
- iOS
- Android NN API
Caffe2 Mobile in Production

Deployed to 1B+ Phones

FASTER INference  
NO FRAGMENTATION  
COMPRESSED MODELS
What is PyTorch?

Automatic differentiation engine
Ndarray library with GPU support
Gradient based optimization package
Utilities (data loading, etc.)

Deep Learning
Reinforcement Learning

numpy-alternative
import torch
from torch.autograd import Variable

trX = torch.linspace(-1, 1, 101)
trY = 2 * trX + torch.randn(*trX.size()) * 0.33

w = Variable(trX.new([0.0]), requires_grad=True)

for i in range(100):
    for (x, y) in zip(trX, trY):
        X = Variable(x)
        Y = Variable(y)

        y_model = X * w.expand_as(X)
        cost = (Y - y_model) * 2
        Cost.backward(torch.ones(*cost.size()))

    w.data = w.data + 0.01 * w.grad.data

print(w)
Writing Data Loaders

PyTorch Solution: Use regular Python to write datasets, leverage existing Python code.

ParlAI (pronounced “par-lay”) is a framework for dialog AI research, implemented in Python.

Its goal is to provide researchers:

- a unified framework for training and testing dialog models
- multi-task training over many datasets at once
- seamless integration of Amazon Mechanical Turk for data collection and human evaluation

Over 20 tasks are supported in the first release, including popular datasets such as SQuAD, bAbI tasks, MCTest, WikiQA, WebQuestions, SimpleQuestions, WikiMovies, QACNN & QADailyMail, CBT, BookTest, bAbI Dialog tasks, Ubuntu Dialog, OpenSubtitles, Cornell Movie and VQA-COCO2014.
Interfacing with environments

Pretty much every environment provides a Python API
PyTorch in Research

GitHub repositories that use PyTorch

NUMBER

7/1/16 1/1/17 7/1/17 1/1/18
Open ecosystem for interoperable AI models.
Seamless Interoperability
ONNX enables models to be trained in one framework and transferred to another for inference.

Any tools exporting ONNX models can benefit from ONNX-compatible runtimes and libraries designed to maximize performance on some of the best hardware in the industry.
Within Facebook

Reimplementation for many models (weeks or months)

**RESEARCH**
- PyTorch
  - Focus on flexibility
  - Define-then-run
  - “Doesn’t get in the way”

**PRODUCTION**
- Caffe2
  - Focus on performance
  - Define-then-run
  - Mobile & server backends
Within Facebook

**RESEARCH**
- Focus on flexibility
- Define-then-run
- “Doesn’t get in the way”

**INTEROP**
- Share possible components

**PRODUCTION**
- Focus on performance
- Define-then-run
- Mobile & server backends
ONNXIFI

ONNX Interface for Framework Integration

TensorRT
Deep Learning Frameworks

**IMPERATIVE**

- ✗ Full graph optimization
- ✗ Production deployment
- ✔ Intuitive programming model
- ✔ Easy debugging

**DECLARATIVE**

- ✔ Full graph optimization
- ✔ Production deployment
- ✗ Intuitive programming model
- ✗ Easy debugging

*PyTorch*  
*Caffe2*
Deep Learning Frameworks

IMPERATIVE & DECLARATIVE

☑ Full graph optimization
☑ Production deployment
☑ Intuitive programming model
☑ Easy debugging

PyTorch 1.0
PyTorch 1.0 Features

- Python ecosystem
- Hybrid Front-End
- Graph optimizations
- Graph-Based IR
- Efficient execution
  - Graph Executor
  - Mobile Deployment
  - ONNX Export
- RNNs without CPython overhead
- TensorRT
@traced
def foo(y, w, t):
    y = x.mm(w)
    print(y)  # still works!
    return y + t

@script
def bar(x, w):
    y = torch.zeros(1, 2)
    for t in x:
        y = foo(y, w, t)
    return y

DEFINE-BY-RUN WITH ONNX EXPORT

[0,0]

For i = range(X.shape[0]):

MatMul

w

X[i]

Add

y

ONNX
PyTorch 1.0

AI Models

Research

Performance Optimizations

Production
Thank You

Learn more at pytorch.org