What's new in PyTorch 1.0

Key changes researchers should be aware of

DEEP BIO

June Oh
15 December 2018

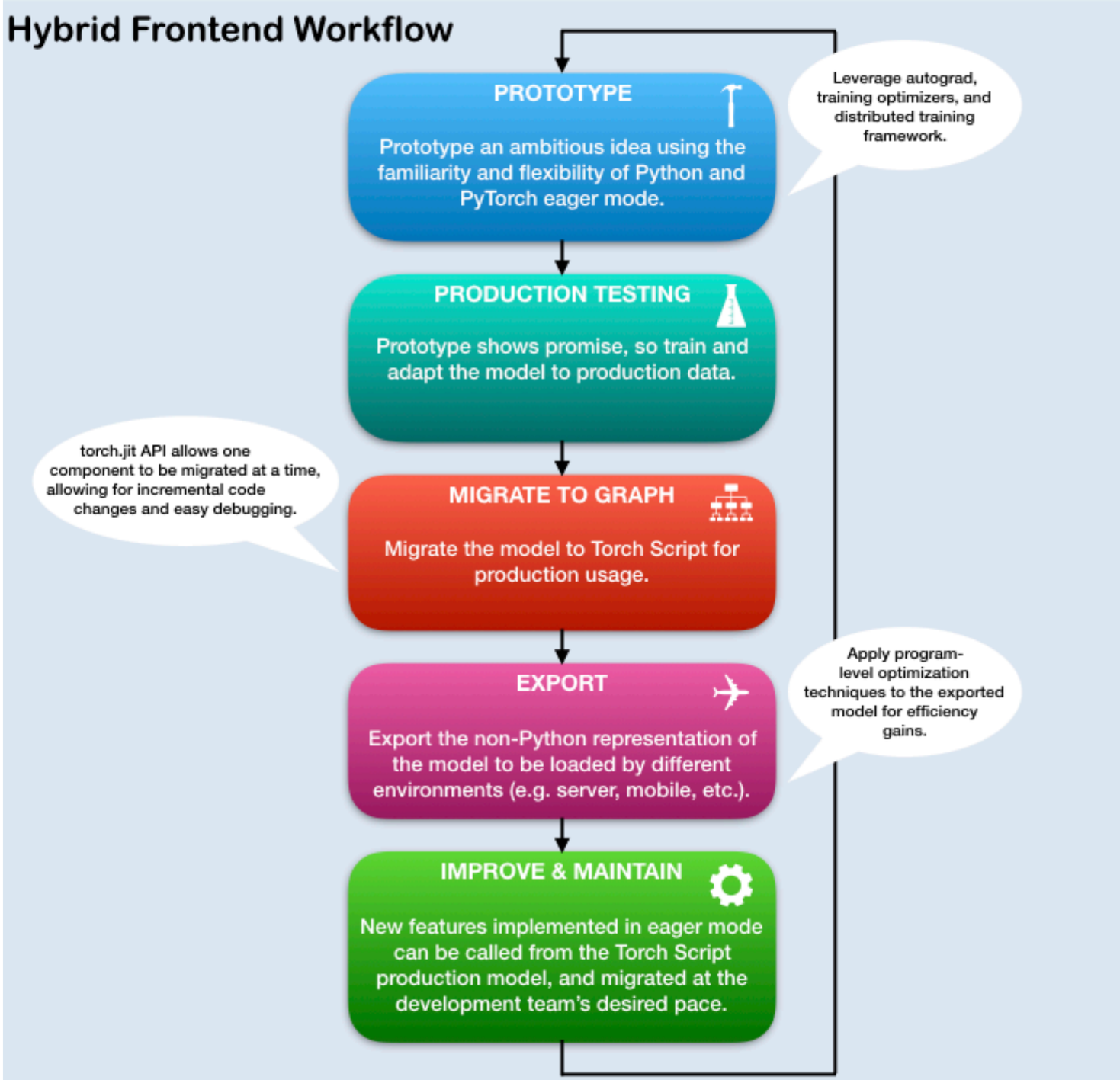
Contents

The Hybrid Frontend

- Graph-based model representation:
  - allows HW and SW optimizations
  - supports framework-agnostic model exports
  - suitable for production
- Imperative interface:
  - more Pythonic
  - easier debugging
  - suitable for research

DEEP BIO

Hybrid Frontend Workflow



DEEP BIO

The Hybrid Frontend

Tracing

- module or function + example inputs = graph-based function
- the model moves out of Python: no data-dependent control flow, no debugging

```
import torch
import torchvision

model = torchvision.models.resnet18()
example = torch.rand(1, 3, 224, 224)

traced_script_module = torch.jit.trace(model, example)
```

The Hybrid Frontend

Scripting

- Python → Torch Script: supports data-dependent control flow and debugging
- Torch Script does not support all Python features

```
class MyModule(torch.jit.ScriptModule):
    def __init__(self, N, M):
        super(MyModule, self).__init__()
        self.weight = torch.nn.Parameter(torch.rand(N, M))
    @torch.jit.script_method
    def forward(self, input):
        if input.sum() > 0:
            output = self.weight.mv(input)
```

The Hybrid Frontend

Related tutorials:

- Loading a PyTorch Model in C++
- Deploying a seq2seq model with the Hybrid Frontend
- ONNX Live Tutorial
- Transferring a model from PyTorch to Caffe2 and mobile using ONNX

Redesigned Distributed Package

- **torch.distributed**: base package with Gloo, NCCL, MPI backend support
- **torch.nn.parallel.DistributedDataParallel**: ready-to-go utility for data parallelism
- related tutorials:
  - PyTorch 1.0 Distributed Trainer with Amazon Web Service
  - Writing Distributed Applications with PyTorch

Torch Hub

Use GitHub as a hub for pretrained models!

- Define entrypoints in **hubconf.py**

```
dependencies = ['torch', 'math']

def resnet18(pretrained=False, *args, **kwargs):
    from torchvision.models.resnet import resnet18 as _resnet18

    model = _resnet18(*args, **kwargs)
    if pretrained:
        model.load_state_dict("pretrained.pth")

    return model
```

Torch Hub

- load entrypoints with **torch.hub.load**

```
hub_model = torch.hub.load(
    'pytorch/vision:master',
    'resnet18',
    1234,
    pretrained=True)
```

C++ Frontend (API unstable)

```python
import torch

model = torch.nn.Linear(5, 1)
optimizer = torch.optim.SGD(model.parameters(), lr=0.1)
prediction = model.forward(torch.randn(3, 5))
loss = torch.nn.functional.mse_loss(prediction, torch.ones(3, 1))
loss.backward()
optimizer.step()
```

```cpp
#include <torch/torch.h>

torch::nn::Linear model(5, 1);
torch::optim::SGD optimizer(model->parameters(), /*lr=*/0.1);
torch::Tensor prediction = model->forward(torch::randn({3, 5}));
auto loss = torch::mse_loss(prediction, torch::ones({3, 1}));
loss.backward();
optimizer.step();
```

DEEPBIO

Bug fixes and deprecations

- torch.nn.functional.softmin was using the incorrect formula in 0.4.1 (#10066)
- DataLoader fixed a couple of issues resulting in hangs (#11985, #12700)
- torch.masked_fill_ now works properly on non-contiguous tensor inputs (#12594)
- Tensor.__delitem__: fixed a segmentation fault on (#12726)
- Removed support for C extensions. Please use cpp extensions (#12122)
- Deleted torch.utils.trainer (#12487)
- torch.distributed.deprecated and torch.nn.parallel.deprecated.DistributedDataParallel are deprecated but still available.

Peformance improvements

- torch.clamp no longer does unnecessary copying (#10352)
- torch.add, torch.sub, torch.mul, torch.div are much faster for non-contiguous tensors on GPU (#8919)
- torch.nn.RNN and related Modules have been ported to C++ and are more performant (#10305, #10481)

Read the full release notes on

https://github.com/pytorch/pytorch/releases/tag/v1.0.0