Learning PyTorch

Qiyang Hu
UCLA Office of Advanced Research Computing
Nov 15, 2021

What is PYTORCH

- An open-source Python-based deep learning framework
  - Primarily developed by Facebook's AI Research lab (FAIR)
  - Replacement for Numpy with supporting GPUs, ROCm, TPUs
  - A full set of deep learning libraries
- History
  - Lua-based Torch (2002 - 2011)
  - PyTorch 0.1 (2016): THNN
  - PyTorch 1.0 (2018): merging Caffe2
  - PyTorch 1.10 (Oct 21, 2021)
- PyTorch as a backend building block
  - Used in Tesla Autopilot, Uber's Pyro, Hugging Face's Transformers
  - Keras-like: PyTorch Lightening, PyTorch Ignite, tensorlayers, fast.ai
  - For specific domains: NiftyTorch, Flair, Skorch, ELF, Detectron2

Why PYTORCH

- Simplicity
  - Feels like Numpy
  - Consistent & great APIs
- Flexibility
  - Defining the model
  - Modifying the model
- Dynamic compute graphs
  - Immediate forward execution
  - Tape-based autograd
  - Destroyed immediately after backprop
- Model serialization and quantization
  - JIT, TorchScript, Tracing, FX
    - Seamlessly switch between Modes, Distributed training, Mobile deployment

A graph is created on the fly


```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))
```

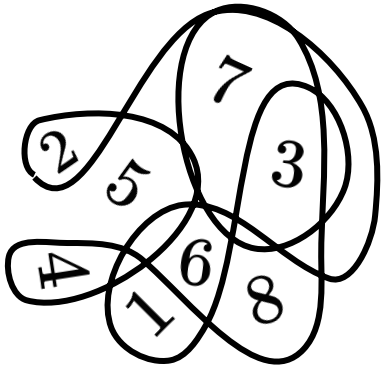
Tensors as building blocks

$$1$$

$$\begin{bmatrix} 2 \\ 5 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 & 3 \\ 5 & 7 & 9 \\ 4 & 8 & 6 \end{bmatrix}$$

$$\begin{bmatrix} \begin{bmatrix} 4 & 6 & 9 \\ 1 & 2 & 5 \\ 8 & 7 & 3 \end{bmatrix} \end{bmatrix}$$



Scalar	Vector	Matrix	Tensor	Tensor
0-D	*1*-D	*2*-D	*3*-D	*n*-D
$X = 1$	$X[1] = 5$	$X[2, 1] = 8$	$X[0, 1, 2] = 5$	$X[2, 3, \ldots, 1] = 6$
				N indices

Tensor, Storage and Views

$$M(i,j) = \text{offset} + \text{stride}[0] \cdot i + \text{stride}[1] \cdot j$$

- Data and Metadata
- Contiguous & Unboxed
- Offset, size, stride

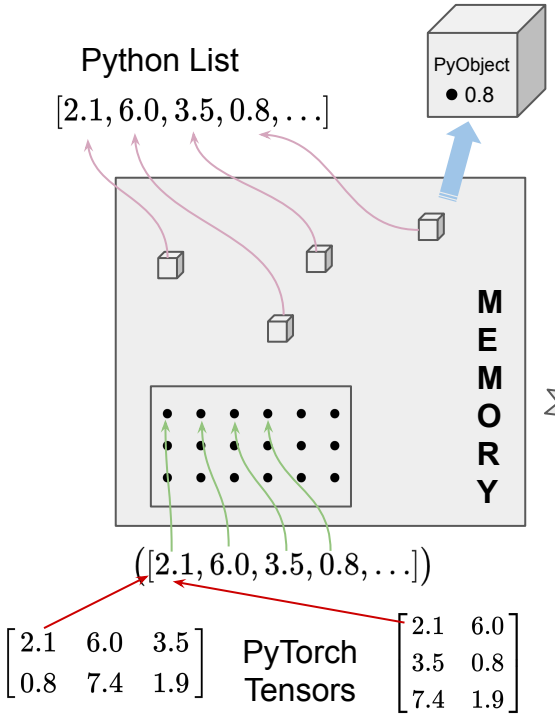
depth

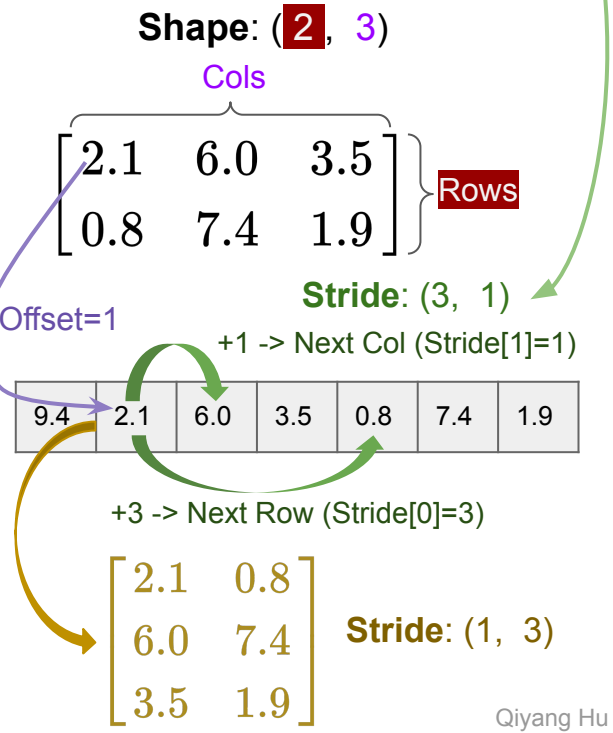
height

$$\begin{bmatrix} 4 & 6 & 9 \\ 1 & 2 & 5 \\ 8 & 7 & 3 \end{bmatrix}$$

width

sizes	(D,H,W)
dtype	integer
device	cuda:0
layout	strided
strides	(H*W,W,1)

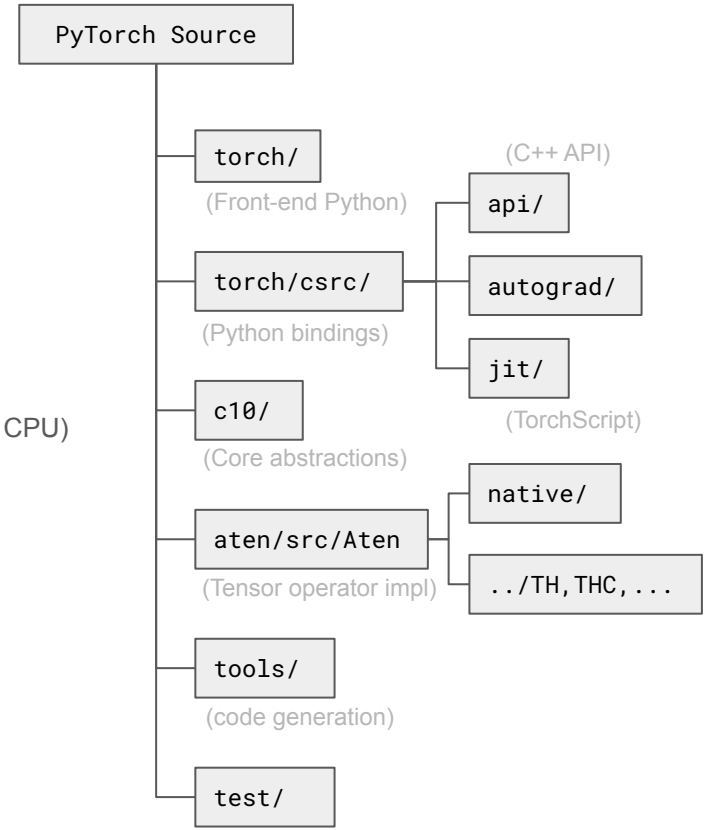




“Py” and “Non-Py” in PyTorch

- Tensor extensions
  - Beyond strided tensors:

    sparse, quantized, encrypted, MKLDNN, TPU tensors etc.
  - Tensor wrapper: device ✖ layout ✖ dtype
- Works with Numpy arrays
  - Easy conversion
  - Zero copy: share their underlying memory locations (if on CPU)
- PyTorch = Python + C/C++ + CUDA
  - Python extension objects in C/C++
  - Code base components:
    - The core Torch libraries: TH, THC, THNN, THCUNN
    - Vendor libraries: CuDNN, NCCL
    - Python Extension libraries
    - Additional 3rd-party libraries: NumPy, MKL, LAPACK, DLPack



Colab Hands-on

bit.ly/learning_pytorch

Automatic differentiation

- Autograd package
  - Track all operations of tensors
  - Compute derivatives analytically via back-prop
  - Natively loaded in torch module
  - Can be used in other scientific domains
- Simple usage
  - Set tensor's .requires_grad as TRUE
  - Call .backward()
    - Gradient accumulated into .grad attribute
    - Tensor's creation function recorded in .grad_fn attribute
- Stop a tensor from tracking history
  - .detach()
  - Wrap the code block in with torch.no_grad()

Optimizers in PyTorch

- torch.optim package
  - Provides various optimization algorithms
  - Need to move model to GPU before constructing optimizers
  - Must zero the gradient explicitly:
    - optimizer.zero_grad()
  - Take an optimization step:
    - optimizer.step() in GD method
    - optimizer.step(closure) in CG or LBFGS method
  - Optional: adjust the learning rate based on the number of epochs.
    - optimizer.lr_scheduler

Neural Networks in PyTorch

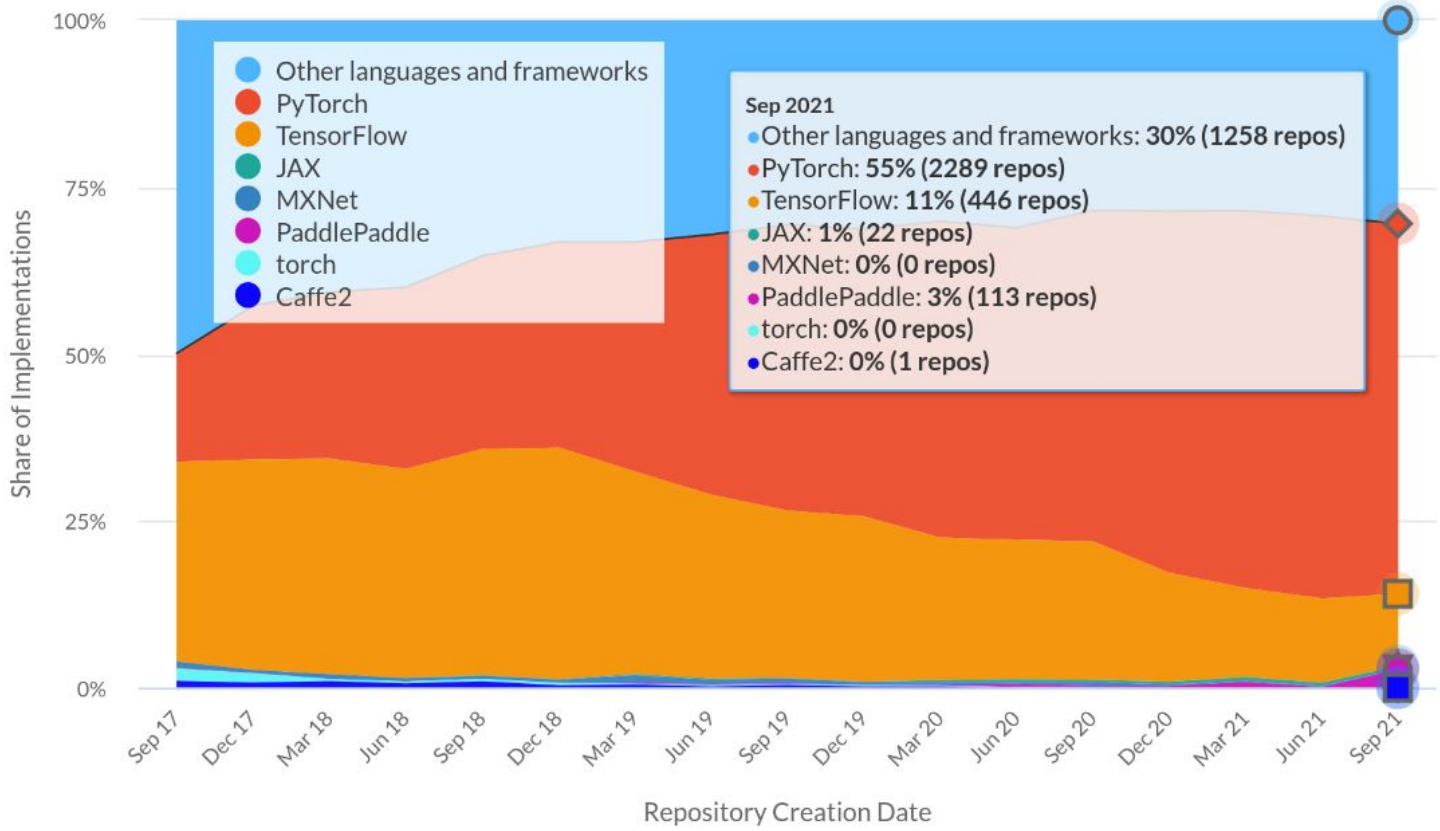
- torch.nn package
  - Contains all building blocks for NN architectures
  - All blocks subclassed from nn.Module (e.g. nn.Linear)
- Define a network
  - For simple networks: concatenate modules through a nn.Sequential container
  - For complex networks: Subclassing nn.Module
- nn.Module package expects first index as first batch size of samples
  - Need to reshape the input by .unsqueeze()
  - Use Dataset and DataLoader
- Loss functions in torch.nn:
  - MSELoss (regression), BCELoss (binary classification), CrossEntropyLoss (multiclass classification)

PyTorch

Tensorflow

JAX

Frameworks used in paper implementations on GitHub



OARC Workshop Survey

https://forms.gle/nbWgNP45qCwZhLRh9