

lec8

Creation Date: 10/01/2020 15:34

Last modified date: 10/01/2020 18:28

Lec 8 : Deep Learning Softwares

- CPU vs GPU

Today

- CPU vs GPU
- Deep Learning Frameworks
 - Caffe / Caffe2
 - Theano / TensorFlow
 - Torch / PyTorch

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 8 -4

University

April 27, 2017

CPU vs GPU

	# Cores	Clock Speed	Memory	Price
CPU	10	3.5 GHz	16 GB	\$1000
GPU	1000	1.5 GHz	16 GB	\$1000

CPU: Fewer cores, but each core is

-
- GPU cores cant run independently whereas CPU cores can.
- Because of the large number of cores in GPU, they are preferred for parallel tasks.
- CPU have cache, but they are relatively small in size.

Example: Matrix Multiplication



-
- Each dot product in the product matrix is independent of other entries.
- Since CPUs run instructions sequentially, matrix multiplication will be slower compared to the calculation done by GPUs.
- **Even Convolution operation can be parallelized on GPUs**

Programming GPUs

- CUDA (NVIDIA only)
 - Write C-like code that runs directly on the GPU

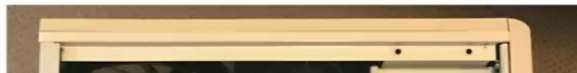
-

CPU vs GPU in practice

(CPU performance not well-optimized, a little unfair)

- **Note: USE cuDNN module, because it's almost 3x faster than 'unoptimized' CUDA.**

CPU / GPU Communication



- Deep Learning Frameworks

This year ...

Caffe
(UC Berkeley)



Caffe2
(Facebook)

Paddle
(Baidu)

CNTK

o

Recall: Computational Graphs

$$f = Wx \quad L = \sum \max(0, s_i - s_i + 1)$$

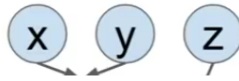
o

- o The point of deep learning frameworks:
 - Easily build big computational graphs
 - Easily compute gradients in computational graphs
 - Run it all efficiently on GPU (wrap around cuDNN, cuBLAS, etc)

Computational Graphs

Numpy

```
import numpy as np
```



o

Computational Graphs



TensorFlow

```
# Basic computational graph  
import numpy as np
```

o

- o The `tf.gradients` function call will compute the gradients at each node.
- o We can change between CPU and GPU by `tf.device` call.

Computational Graphs



PyTorch

```
import torch
from torch.autograd import Variable
```

-
- the Variable function creates a node in PyTorch
- the backward() function compute the gradients.
- We can use .cuda() or .cpu() to switch between gpu and cpu.

TensorFlow:

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
```

-
- tf.placeholders are just locations, no memory is allocated.
- until we create a session and run(), we are just creating the computational graph, there is no data in the system.
- We create the data and assign it to 'values'.
- in session.run() we pass the list of parameters whose value we want to compute.

TensorFlow: Distributed Version



Side Note: Theano

```
import theano
import theano.tensor as T

# Batch size input dim hidden dim num classes
```

PyTorch: Three Levels of Abstraction

TensorFlow equivalent

-
- the difference between numpy array and a tensor is that the tensor runs on a GPU.

PyTorch: Autograd

```
import torch
from torch.autograd import Variable
```

-

PyTorch: Autograd

```
import torch
from torch.autograd import Variable
```

-
- In PyTorch we create the graph everytime we do the forward pass.

PyTorch: New Autograd Functions

```
class PoH(torch.autograd.Function):
```

-

PyTorch: New Autograd Functions

```
class ReLU(torch.autograd.Function):
```

o

PyTorch: nn

```
import torch
from torch.autograd import Variable

class TwoLayerNet(torch.nn.Module):
    def __init__(self, D_in, H, D_out):
```

o

PyTorch: nn

```
import torch
from torch.autograd import Variable

class TwoLayerNet(torch.nn.Module):
    def __init__(self, D_in, H, D_out):
```

o

PyTorch: DataLoaders

```
import torch
from torch.autograd import Variable
from torch.utils.data import DataLoader, Dataset
```

o

Aside: Torch

```
require 'torch'
require 'nn'
require 'optim'

local N, D, H, C = 64, 256, 512, 10

local model = nn.Sequential()
model:add(nn.Linear(D, H))
```

o

Static vs Dynamic Graphs

TensorFlow: Build graph once, then

PvTorch: Each forward pass defines

o

Static vs Dynamic: Optimization

With static graph:

The graph you wrote

Equivalent graph with

o

Static vs Dynamic: Serialization

Static

Dvnamic

o

Static vs Dynamic: Conditional

Control flow

TensorFlow: Special TF

o

Static vs Dynamic: Loops

$$v_i = (v_{i-1} + x_i) * w$$



o

Dynamic Graphs in TensorFlow

TensorFlow Fold make dynamic

o

Dynamic Graph Applications

- Recurrent networks

o

Dynamic Graph Applications

- Recurrent networks



o

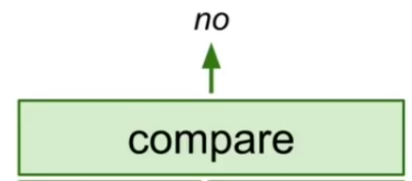
Dynamic Graph Applications

- Recurrent networks



o

Dynamic Graph Applications



o

Caffe Overview

- Core written in C++
- Has Python and MATLAB bindings

o

Caffe step 1: Convert Data

- Data loading from LMDB is the easiest

o

Caffe step 2: Define Network (prototxt)

```
name: "LogisticRegressionNet"
```

o

Caffe step 2: Define Network (prototxt)

```
6747 layer {
6748     bottom: "res5c"
```

o

Caffe step 3: Define Solver (prototxt)

o

Caffe Pros / Cons

- (+) Good for feedforward networks

◦

Caffe2 Overview

- Very new - released a week ago =)

◦

Google:
TensorFlow

Facebook:
PyTorch + Caffe2

o

My Advice:

Subtitle track: Disable

TensorFlow is a safe bet for most projects. Not perfect but has

o