Getting started with Torch

Alison B Lowndes

Deep Learning Solutions Architect & Community Manager| EMEA

AGENDA

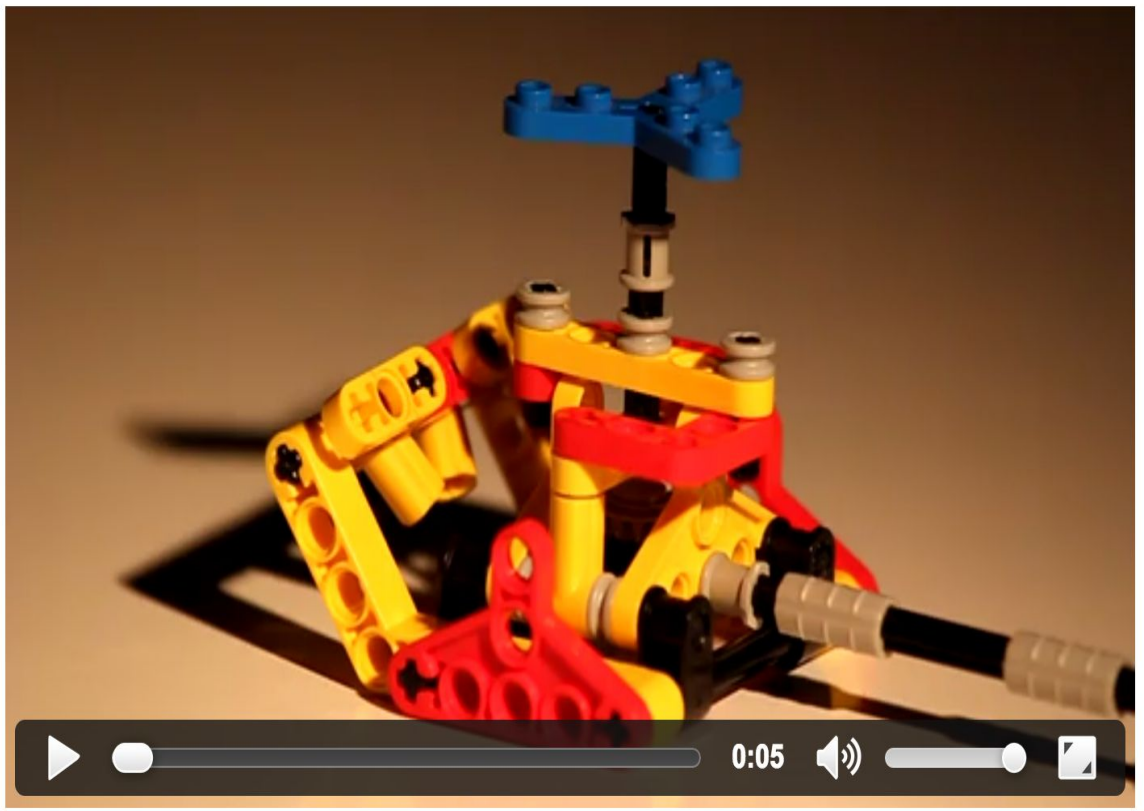
- Previous classes
- iTorch
- Why Deep Learning
- About Torch
- CheatSheet
- Benefits of Torch
- Installation
- LuaRocks
- Tensors
- Qwiklabs
- Software releases
- RoadMap
- Advanced

iTorch

- Inline help via ?
- Great visualization functions for images, video, audio, html.
- Can plot to screen in notebook mode, or save to disk as a html file.
- git clone https://github.com/facebook/iTorch.git
- $ itorch notebook

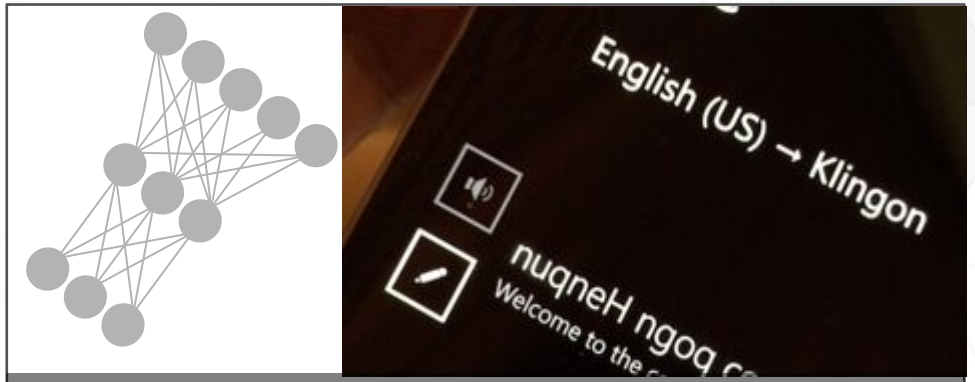
Demo: HERE

```
itorch.video('small.mp4')
```

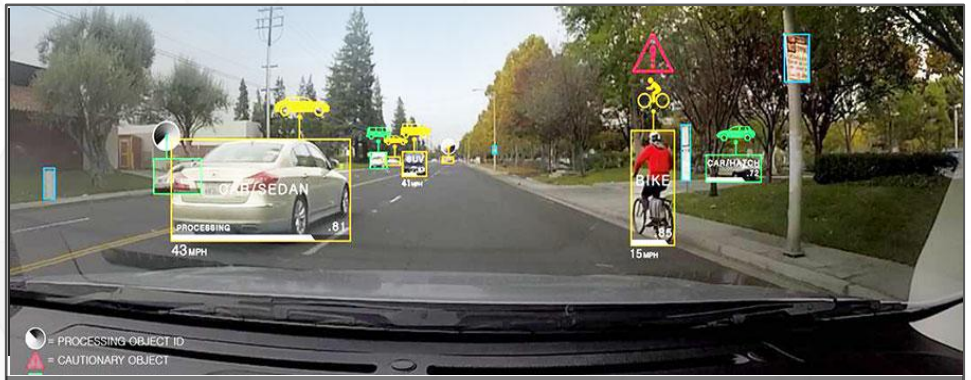


PRACTICAL DEEP LEARNING EXAMPLES

Image Classification, Object Detection, Localization,
Action Recognition, Scene Understanding



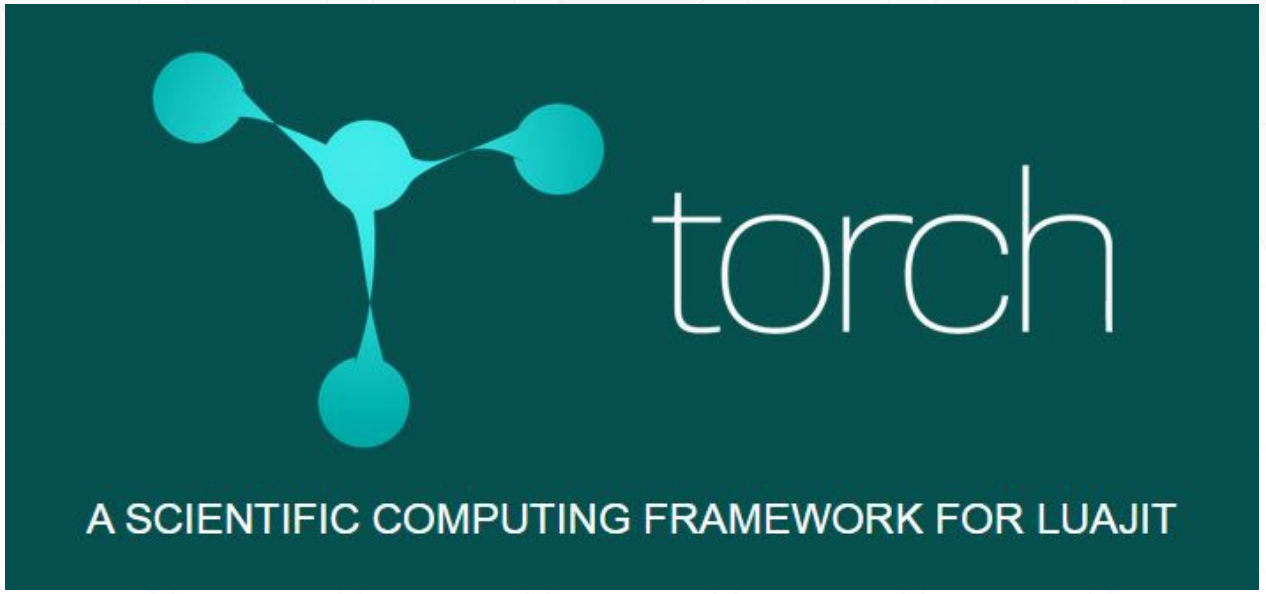
Speech Recognition, Speech Translation,
Natural Language Processing



Pedestrian Detection
Traffic Sign Recognition

Breast Cancer Cell Mitosis Detection,
Volumetric Brain Image Segmentation

*Credited to Yann LeCun, Facebook AI Research & Center for Data Science, NYU



Circa 2000 - now Torch7 - 4th (using odd numbers only 1,3,5,7)
Web-scale learning in speech, image and video applications

Maintained by top researchers:
Ronan Collobert - Research Scientist @ Facebook
Clement Farabet - Senior Software Engineer @ Twitter
Koray Kavukcuoglu - Research Scientist @ Google DeepMind
Soumith Chintala - Research Engineer @ Facebook

Cheatsheet

Cheatsheet: https://github.com/torch/torch7/wiki/Cheatsheet

Github: https://github.com/torch/torch7

Google Group for new users and installation queries:
https://groups.google.com/forum/embed/?place=forum%2Ftorch7#!forum/torch7

Advanced only https://gitter.im/torch/torch7

Why use Torch?

Recasting a pre-defined model as a CUDA model for use on GPU's is as simple as :

```
model:cuda()
```

```
--define input as a CUDA Tensor
input = torch.CudaTensor(100)
```

Run-time speed is a priority ~ Convenience of a scripting interface ~ multiGPU

The Benefits

- An efficient tensor library (similar to NumPy) with CUDA for neural network computations
- Dedicated Neural Networks (nn.) package to build DAGs directed acyclic computation graphs with automatic differentiation
- Really great community and industry support - several hundred community-built and maintained packages - active contribution from web scale companies
- In active use for state-of-the-art neural network research.
- Fast, efficient, easy to use multi-GPU support and distribution of neural network training
- Awesome interface to C via the Lua scripting language - very fast due to Just In Time (JIT) compilation (similar to Javascript in syntax).
- Very easy to learn.
- Embeddable, with ports to iOS, Android and FPGA backends

Torch Core

The torch core consists of the following packages:

- torch : tensors, class factory, serialization, BLAS
- nn : neural network Modules and Criterions
- optim : SGD, LBFGS and other optimization functions
- gnuplot : ploting and data visualization
- paths : make directories, concatenate file paths, and other filesystem utilities
- image : save, load, crop, scale, warp, translate images and such
- trepl : the torch LuaJIT interpreter
- cwrap : used for wrapping C/CUDA functions in Lua

Lua

Lua is a powerful, fast, lightweight, embeddable scripting language combining simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics.

The language is maintained by a team at PUC-Rio, the Pontifical Catholic University of Rio de Janeiro in Brazil - born and raised in Tecgraf, formerly the Computer Graphics Technology Group of PUC-Rio. Lua is now housed at LabLua, a laboratory of the Department of Computer Science of PUC-Rio.

- Tables: the only data structure
- Associative arrays indexed numerically or with strings, other values of the language
- Tables are unlimited in size and allow virtual tables

LuaRocks

```
$ luarocks install image    # an image library for Torch7
$ luarocks install nnx      # lots of extra neural-net modules
$ luarocks install camera   # a camera interface for Linux/MacOS
$ luarocks install ffmpeg   # a video decoder for most formats
$ ...
```

Many many packages (rocks) available including: torch (core), randomkit, signal.

Dataformats supported inc. CSV, HDF5, mattorch, JSON, image, audio, video

ML: nn, dp, dpnn, unsup, reinforcement learning …..

CV: fex - SIFT, NLP, Sensor I/O for drones, parallel processing and distributed computing

LuaRocks

- List of Packages by Category

Core Math	**Visualization**	**Utility libraries**
Data formats I/O	**Sensor I/O**	**Databases**
Machine Learning	**Computer Vision**	NLP
Parallel Processing	**CUDA**	**OpenCL**
Images	**Videos**	**Audio**
Asynchronous	**Networking**	**Security**
Alternative REPLs	**Interfaces to third-party libs**	Reinforcement Learning
Miscellaneous		

https://github.com/torch/torch7/wiki/Cheatsheet#list-of-packages-by-category

Docker Images

```
docker pull kaixhin/torch
```

CUDA 7.5 version - see repo for requirements

```
docker pull kaixhin/cuda-torch
```

CUDA 7.0 version - see repo for requirements

```
docker pull kaixhin/cuda-torch:7.0
```

CUDA 6.5 version - see repo for requirements

```
docker pull kaixhin/cuda-torch:6.5
```

Installation

http://torch.ch/docs/getting-started.html#_

For Matlab users, UCLA's Ata Mahjoubfar, PhD written: http://atamahjoubfar.github.io/Torch_for_Matlab_users.pdf

For Numpy users: https://github.com/torch/torch7/wiki/Torch-for-Numpy-users

For advanced users: refer to the "gotchas" https://luapower.com/luajit-notes

Command line install

 # in a terminal, run the commands

```
curl -s https://raw.githubusercontent.com/torch/ezinstall/master/install-deps
| bash
git clone https://github.com/torch/distro.git ~/torch --recursive
cd ~/torch; ./install.sh
```

Once installed run torch with "th" to start the TREPL (torch read-eval-print loop).

```
$ th main.lua -data /disk1/imagenet -nGPU 2 -
backend cudnn -netType alexnet
```

Screenshot

In [ ]:

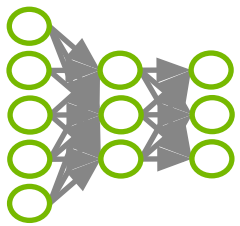
```
net = nn.Sequential()
net:add(nn.SpatialConvolution(1, 6, 5, 5)) -- 1 input image channel, 6 output channels, 5x5 convol
ution kernel
net:add(nn.SpatialMaxPooling(2,2,2,2))     -- A max-pooling operation that looks at 2x2 windows an
d finds the max.
net:add(nn.SpatialConvolution(6, 16, 5, 5))
net:add(nn.SpatialMaxPooling(2,2,2,2))
net:add(nn.View(16*5*5))                    -- reshapes from a 3D tensor of 16x5x5 into 1D tensor
of 16*5*5
net:add(nn.Linear(16*5*5, 120))             -- fully connected layer (matrix multiplication betwee
n input and weights)
net:add(nn.Linear(120, 84))
net:add(nn.Linear(84, 10))                   -- 10 is the number of outputs of the network (in thi
s case, 10 digits)
net:add(nn.LogSoftMax())                     -- converts the output to a log-probability. Useful f
or classification problems

print('Lenet5\n' .. net:__tostring());
```

cuDNN

Deep Learning Primitives

IGNITING ARTIFICIAL INTELLIGENCE

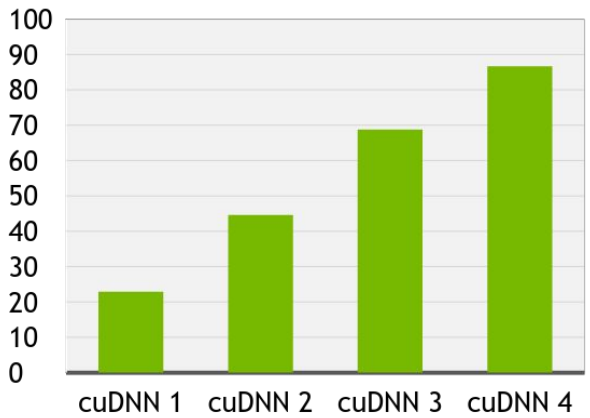


developer.nvidia.com/cudnn

- GPU-accelerated Deep Learning subroutines
- High performance neural network training
- Accelerates Major Deep Learning frameworks: Caffe, Theano, Torch
- Up to 3.5x faster AlexNet training in Caffe than baseline GPU

Tiled FFT up to 2x faster than FFT

Millions of Images Trained Per Day

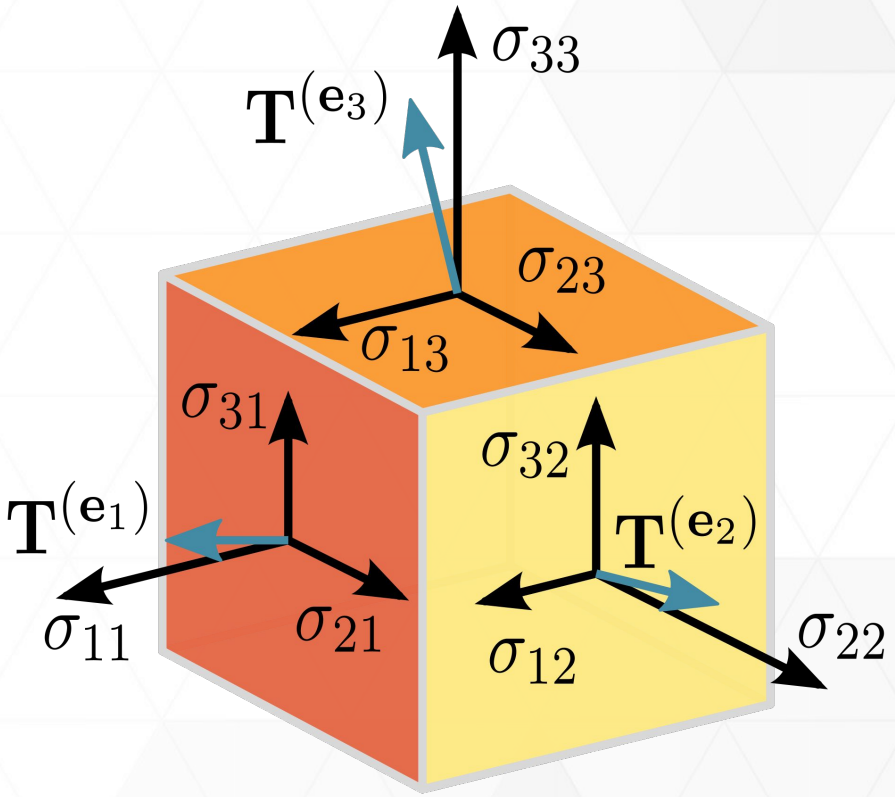


TensorFlow
torch
Caffe
Microsoft CNTK
Purine
K KERAS
theano
Chainer
MINERVA
DL4J Deeplearning4j
MatConvNet

Tensors

The Tensor class is the most important class in Torch.
A Tensor is a serializable, potentially multi-dimensional matrix.
The number of dimensions is unlimited - created using LongStorage.

A Tensor is a particular way of viewing a Storage: a Storage only represents a chunk of memory, while the Tensor interprets this chunk of memory as having dimensions



Tensor tutorial

by Georg Ostrovski, DeepMind [GitHub]

--- creation of a 4D-tensor 4x5x6x2

```
z = torch.Tensor(4,5,6,2)
```

--- for more dimensions, use LongStorage:

```
s = torch.LongStorage(6)
 s[1] = 4; s[2] = 5; s[3] = 6; s[4] = 2; s[5] = 7; s[6] = 3;
 x = torch.Tensor(s)
```

The number of dimensions of a Tensor can be queried by nDimension() or dim(). Size of the i-th dimension is returned by size(i). A LongStorage containing all the dimensions can be returned by size().

https://github.com/torch/torch7/blob/master/doc/tensor.md

CUDA

Call `require "cutorch"` on a CUDA-capable machine

- cutorch - Torch CUDA Implementation
- cunn - Torch CUDA Neural Network Implementation
- cunnx - Experimental CUDA NN implementations
- cudnn - NVIDIA CuDNN Bindings

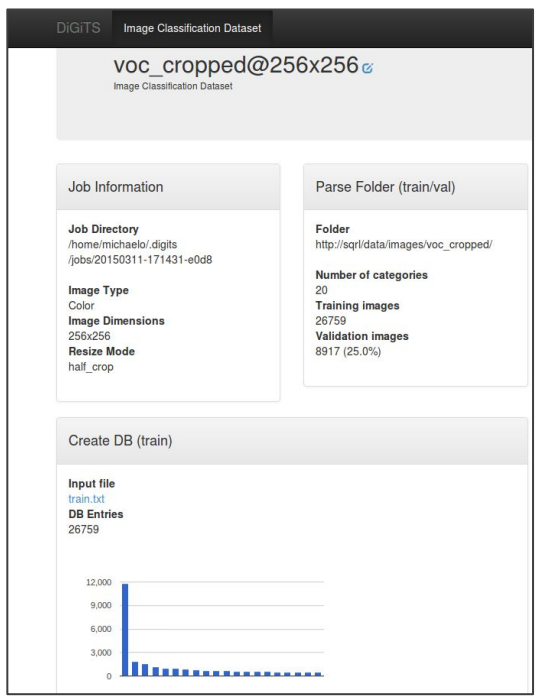
You get an additional tensor type torch.CudaTensor (just like torch.FloatTensor).
CUDA double precision is not supported.

NVIDIA Jetson TK1
Installation and usage instructions for Torch + CuDNN https://github.com/e-lab/torch-toolbox/blob/master/Tutorials/Setup-Torch-cuDNN-on-Jetson-TK1.md

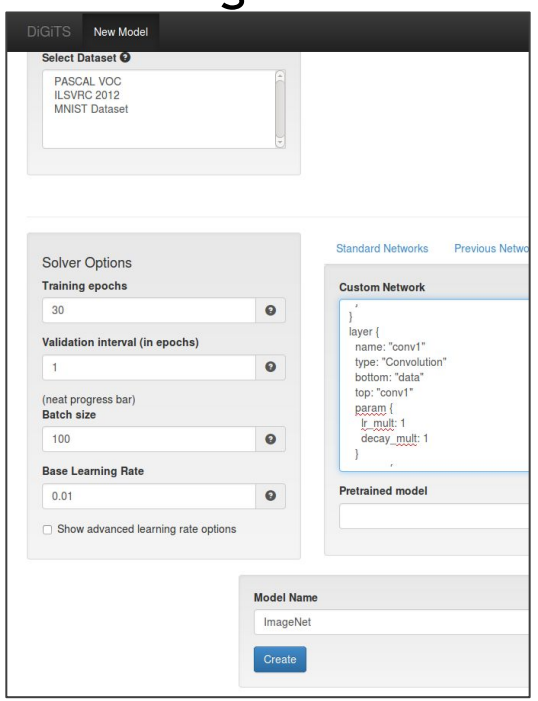
NVIDIA DIGITS with Torch7 - preview available!

Interactive Deep Learning GPU Training System

Process Data

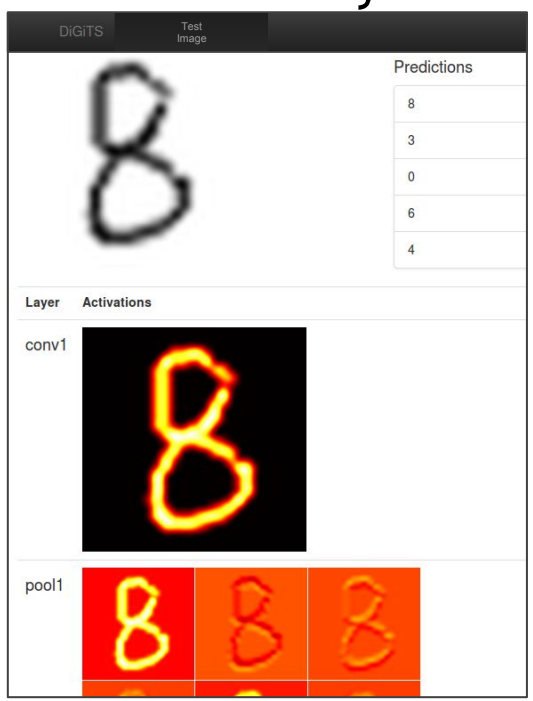


Configure DNN



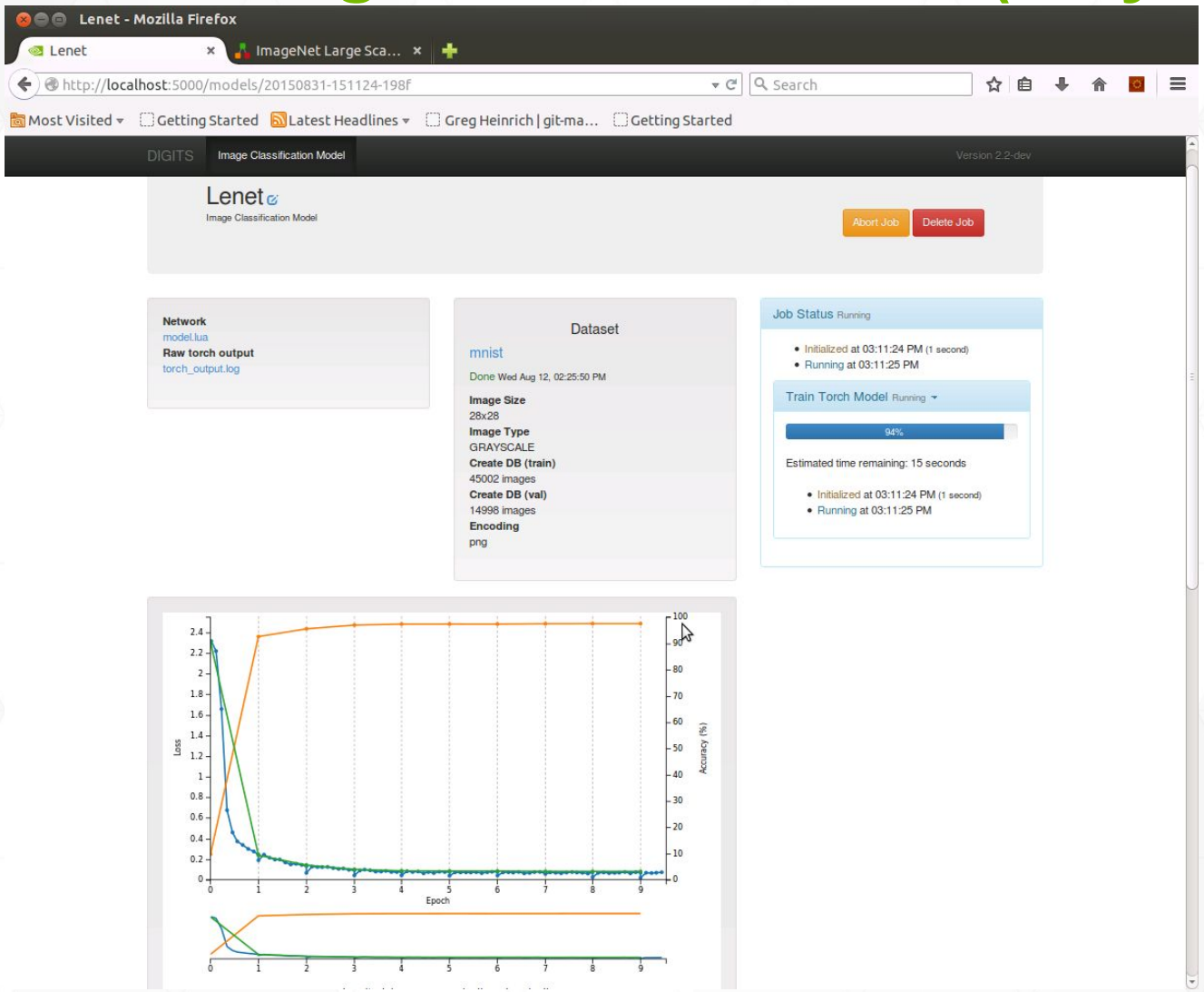
Monitor Progress

Visualize Layers



http://developer.nvidia.com/digits

DiGiTs full integration with Torch (very soon)



Stdout Logs

015-08-31 15:11:26 [INFO] Loading network definition from /fast-scratch/…./ws/digits/digits/jobs/20150831-151124-198f/model

2015-08-31 15:11:26 [INFO] Loading mean tensor from /fast-scratch/.../ws/digits/digits/jobs/20150812-142520-8414/mean.jpg file

2015-08-31 15:11:26 [INFO] Loading label definitions from /fast-scratch/.../ws/digits/digits/jobs/20150812-142520-8414/labels.txt file

2015-08-31 15:11:26 [INFO] found 10 categories

2015-08-31 15:11:26 [INFO] Network definition:

nn.Sequential {

[input -> (1) -> (2) -> (3) -> (4) -> (5) -> (6) -> (7) -> (8) -> (9) -> (10) -> output]

(1): nn.MulConstant

(2): nn.SpatialConvolution(1 -> 20, 5x5)

(3): inn.SpatialMaxPooling(2,2,2,2)

(4): nn.SpatialConvolution(20 -> 50, 5x5)

(5): inn.SpatialMaxPooling(2,2,2,2)

(6): nn.View

(7): nn.Linear(800 -> 500)

(8): nn.ReLU

(9): nn.Linear(500 -> 10)

(10): nn.LogSoftMax

….

Stdout Logs

2015-08-31 15:11:26 [INFO] switching to CUDA

2015-08-31 15:11:26 [INFO] opening train lmdb file: /fast-scratch/.../ws/digits/digits/jobs/20150812-142520-8414/train_db

2015-08-31 15:11:26 [INFO] Loaded train image details from the mean file: Image channels are 1, Image width is 28 and Image height is 28

2015-08-31 15:11:26 [INFO] found 45002 images in train db/fast-scratch/.../ws/digits/digits/jobs/20150812-142520-8414/train_db

2015-08-31 15:11:26 [INFO] loading all the keys from train db

2015-08-31 15:11:26 [INFO] opening validation lmdb file: /fast-scratch/.../ws/digits/digits/jobs/20150812-142520-8414/val_db

2015-08-31 15:11:26 [INFO] Loaded train image details from the mean file: Image channels are 1, Image width is 28 and Image height is 28

2015-08-31 15:11:26 [INFO] found 14998 images in train db/fast-scratch/.../ws/digits/digits/jobs/20150812-142520-8414/val_db

2015-08-31 15:11:26 [INFO] loading all the keys from validation db

2015-08-31 15:11:26 [INFO] initializing the parameters for learning rate policy: step

2015-08-31 15:11:26 [INFO] initializing the parameters for Optimizer

2015-08-31 15:11:26 [INFO] During training. details will be logged after every 5000 images

2015-08-31 15:11:26 [INFO] Training epochs to be completed for each validation : 1

2015-08-31 15:11:26 [INFO] Training epochs to be completed before taking a snapshot : 1

2015-08-31 15:11:26 [INFO] While logging, epoch value will be rounded to 4 significant digits

2015-08-31 15:11:26 [INFO] Model weights will be saved as snapshot_<EPOCH>_Weights.t7

2015-08-31 15:11:26 [INFO] started training the model

2015-08-31 15:11:30 [INFO] Validation (epoch 0): loss = 2.3070402155553, accuracy = 0.097279637284971

2015-08-31 15:11:30 [INFO] Training (epoch 0.0007): loss = 2.3174071311951, lr = 0.01

Stdout Logs

2015-08-31 15:11:32 [INFO] Training (epoch 0.1124): loss = 2.2207900915936, lr = 0.01

2015-08-31 15:11:35 [INFO] Training (epoch 0.224): loss = 1.6579078219499, lr = 0.01

2015-08-31 15:11:38 [INFO] Training (epoch 0.3356): loss = 0.67513685943974, lr = 0.01

2015-08-31 15:11:40 [INFO] Training (epoch 0.4473): loss = 0.45962502422986, lr = 0.01

2015-08-31 15:11:43 [INFO] Training (epoch 0.5589): loss = 0.3724656421098, lr = 0.01

2015-08-31 15:11:45 [INFO] Training (epoch 0.6705): loss = 0.3372980668951, lr = 0.01

2015-08-31 15:11:48 [INFO] Training (epoch 0.7822): loss = 0.29915498543507, lr = 0.01

2015-08-31 15:11:51 [INFO] Training (epoch 0.8938): loss = 0.27473048248868, lr = 0.01

2015-08-31 15:11:57 [INFO] Validation (epoch 1): loss = 0.23767776332939, accuracy = 0.92659021202827

2015-08-31 15:11:57 [INFO] Snapshotting to /fast-scratch/.../ws/digits/digits/jobs/20150831-151124-198f/snapshot_1_Weights.t7

2015-08-31 15:11:57 [INFO] Snapshot saved - /fast-scratch/.../ws/digits/digits/jobs/20150831-151124-198f/snapshot_1_Weights.t7

2015-08-31 15:11:57 [INFO] Training (epoch 1): loss = 0.2430327351888, lr = 0.01

2015-08-31 15:11:57 [INFO] Training (epoch 1.0007): loss = 0.18871998786926, lr = 0.01

2015-08-31 15:12:00 [INFO] Training (epoch 1.1124): loss = 0.24316110186706, lr = 0.01

2015-08-31 15:12:02 [INFO] Training (epoch 1.224): loss = 0.21386161679105, lr = 0.01

2015-08-31 15:12:05 [INFO] Training (epoch 1.3356): loss = 0.19611184303738, lr = 0.01

2015-08-31 15:12:08 [INFO] Training (epoch 1.4473): loss = 0.19667462947642, lr = 0.01

2015-08-31 15:12:10 [INFO] Training (epoch 1.5589): loss = 0.16826266747941, lr = 0.01

2015-08-31 15:12:13 [INFO] Training (epoch 1.6705): loss = 0.14886514187618, lr = 0.01

Example NN in Lua

```
 1: -- parameters
 2: nstates = {16,256,128}
 3: fanin = {1,4}
 4: filtsize = 5
 5: poolsize = 2
 6: normkernel = image.gaussian1D(7)
 7:
 8: -- Container:
 9: model = nn.Sequential()
10:
11: -- stage 1 : filter bank -> squashing -> L2 pooling -> normalization
12: model:add(nn.SpatialConvolutionMap(nn.tables.random(nfeats, nstates[1], fanin[1]), filtsize, filtsize))
13: model:add(nn.Tanh())
14: model:add(nn.SpatialLPPooling(nstates[1],2,poolsize,poolsize,poolsize,poolsize))
15: model:add(nn.SpatialSubtractiveNormalization(16, normkernel))
16:
17: -- stage 2 : filter bank -> squashing -> L2 pooling -> normalization
18: model:add(nn.SpatialConvolutionMap(nn.tables.random(nstates[1], nstates[2], fanin[2]), filtsize, filtsize))
19: model:add(nn.Tanh())
20: model:add(nn.SpatialLPPooling(nstates[2],2,poolsize,poolsize,poolsize,poolsize))
21: model:add(nn.SpatialSubtractiveNormalization(nstates[2], normkernel))
22:
23: -- stage 3 : standard 2-layer neural network
24: model:add(nn.Reshape(nstates[2]*filtsize*filtsize))
25: model:add(nn.Linear(nstates[2]*filtsize*filtsize, nstates[3]))
26: model:add(nn.Tanh())
27: model:add(nn.Linear(nstates[3], noutputs))
```

Sample of y-channel data from SVHN dataset

TEST ON SVHN dataset with 1x K40

$ th -i doall.lua -size small -type cuda // testing on 10,000 images

```
==> processing options

==> switching to CUDA ..............

==> here is the model:

nn.Sequential {

  [input -> (1) -> (2) -> (3) -> (4) -> (5) -> (6) -> (7) -> (8) -> (9) -> (10) -> (11) -
> output]

  (1): nn.SpatialConvolutionMM

  (2): nn.ReLU

  (3): nn.SpatialMaxPooling

  (4): nn.SpatialConvolutionMM

  (5): nn.ReLU

  (6): nn.SpatialMaxPooling

  (7): nn.View

  (8): nn.Dropout

  (9): nn.Linear(1600 -> 128)

  (10): nn.ReLU

  (11): nn.Linear(128 -> 10)
```

Stopped at 92.95% test accuracy.

1x 2.66GHz CPU hyperthreaded to 12 == 19 minutes

1x Tesla K40 == 4 mins

4.75x speedup

Torch

Further help

- http://torch.ch/support.html
- **Torch** is largely community-maintained but for beginner or install help please use the Google Group https://groups.google.com/forum/embed/?place=forum%2Ftorch7#!forum/torch7
- The **GitHub issues** page https://github.com/torch/torch7/issues
- Advanced question via Gitter https://gitter.im/torch/torch7

HANDS-ON LAB

https://nvidia.qwiklab.com/tags/free

1. Create an account at nvidia.qwiklab.com
2. Go to “Getting started with Torch7” lab
3. Start the lab and enjoy ;)

Only requires a supported browser

No code runs on your machine

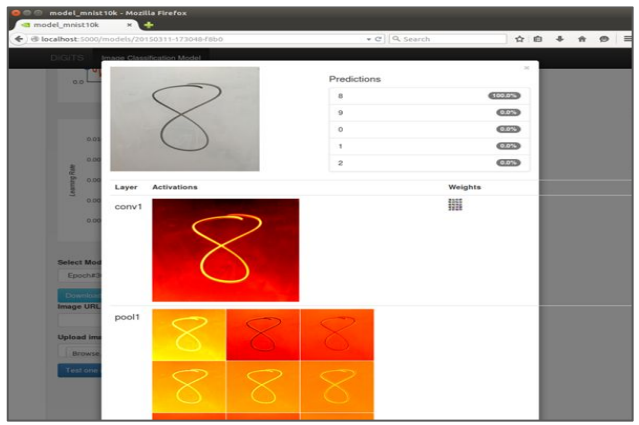
NVIDIA GPU supplied!

Free until end of Deep Learning Lab series.

Deep Learning Performance Doubles

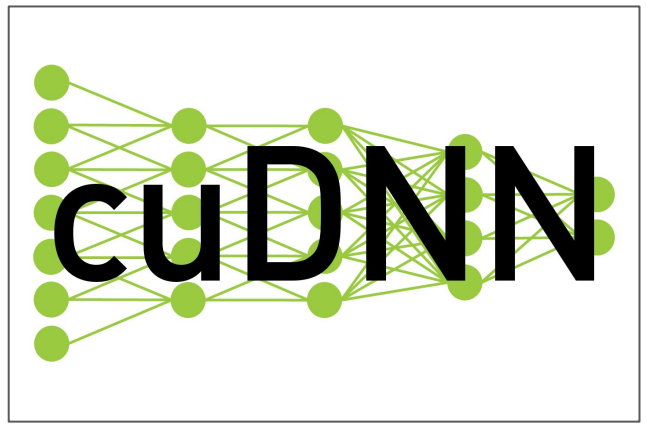
For Data Scientist and Researchers

Train Models up to 2x Faster with Automatic Multi-GPU Scaling



DIGITS 3

2x Faster Single GPU Training Support for Larger Models



cuDNN 4

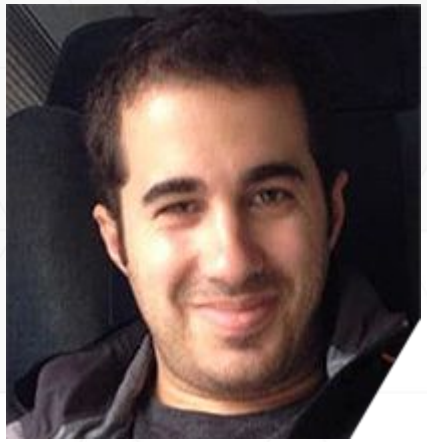
2x Larger Datasets Instruction-level Profiling

CUDA 7.5

TORCH RoadMap

- 'nn' container unified interface between containers and graph
- 'rnn' package
- Split nn into THNN and nn - THNN == NN package using TH as backend + nn == lua layer
- THNN can be used as a standalone C library
- Same for cunn
- Publishing more tutorials & a universal dataset API to support both CPU & GPU
- Model Zoo like Caffe

Advanced



Applied Torch7

FOR VISION AND NATURAL LANGUAGE

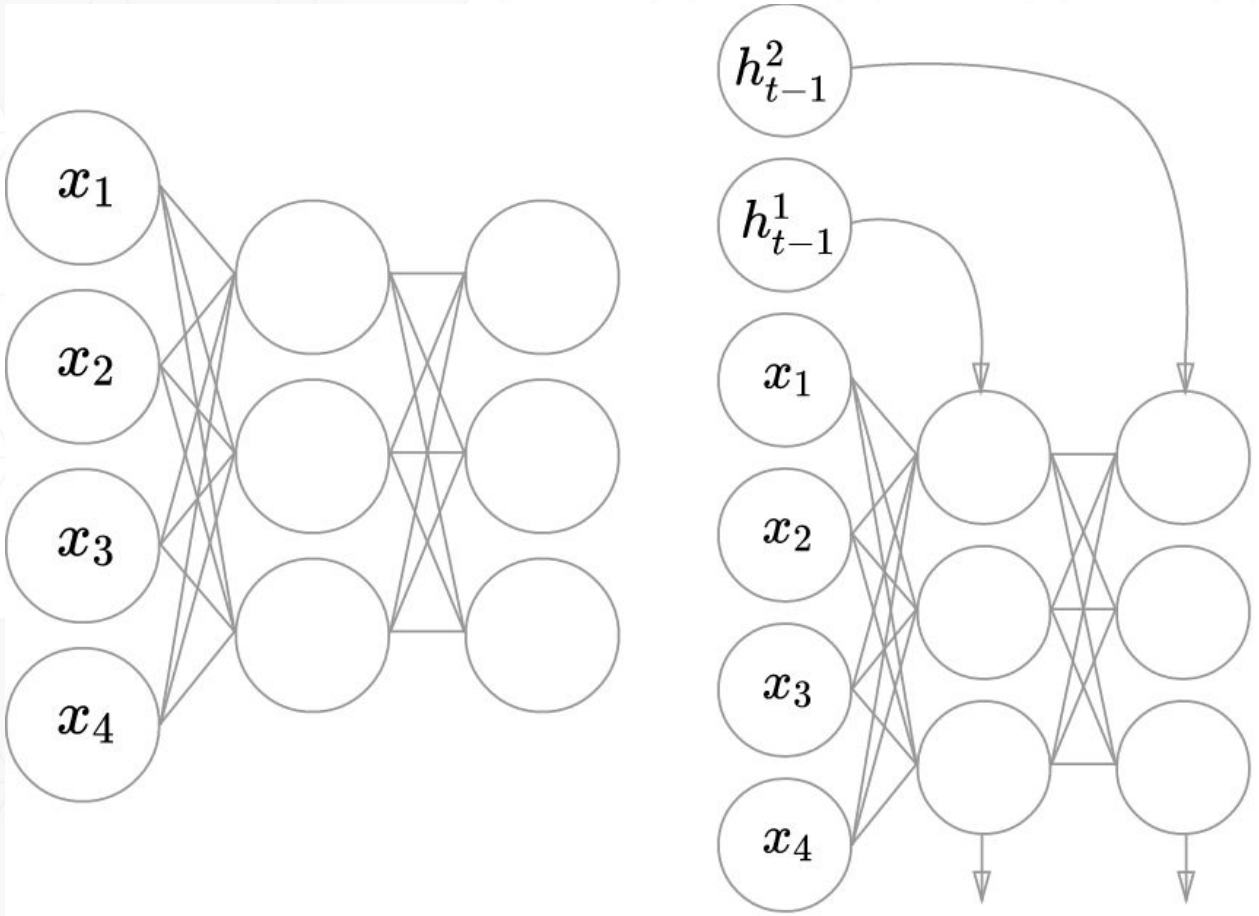
Nicholas Léonard, Element Inc, Research Engineer

Webinar targeted at ML enthusiasts and researchers and covers applying deep learning techniques on classifying images and building language models, including convolutional and recurrent neural networks.

WATCH NOW at http://www.gputechconf.com/gtc-webinars

Torch implementation of LSTM

by Adam Paszke http://apaszke.github.io/lstm-explained.html



In their most simplest form LSTM units consist of a cell state & three gates:

- **input**
- **forget**
- **output + a cell unit.**

Gates use a sigmoid activation.

Input and cell state are transformed with tanh.

Sample code

by Adam Paszke http://apaszke.github.io/lstm-explained.html

The network will be implemented as a `nngraph.gModule` with the following layers:

- `nn.Identity()` - passes on the input
- `nn.Dropout(p)` - standard dropout module
- `nn.Linear(in, out)` - an affine transform
- `nn.Narrow(dim, start, len)` - selects a subvector along dim dimension
- `nn.Sigmoid()` - applies sigmoid element-wise
- `nn.Tanh()` - applies tanh element-wise
- `nn.CMulTable()` - outputs the product of tensors in forwarded table
- `nn.CAddTable()` - outputs the sum of tensors in forwarded table

Reinforcement learning with RL-Glue

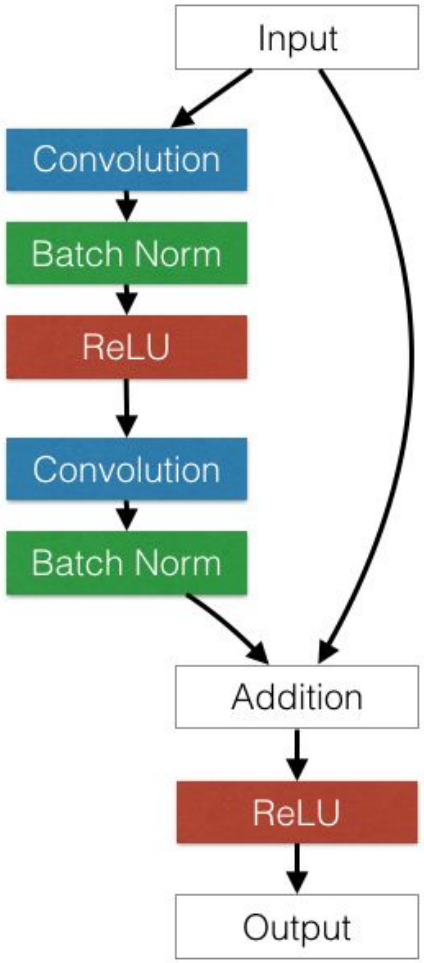
http://glue.rl-community.org

RL-Glue (Reinforcement Learning Glue) is an interface that allows you to connect reinforcement learning agents, environments, and experiment programs together.

First install the **RL-Glue Core**

rlenvs in Torch provide the environments; **rlenvs**.Atari .Blackjack, .CartPole, .CliffWalking, .GridWorld, MountainCar, MultiArmedBandit, .WindyWorld etc.

Training ResNets on Torch: ImageNet15 winning network architecture



Sam Gross (FAIR) and Michael Wilber (Cornell)
http://torch.ch/blog/2016/02/04/resnets.html

Code + pretrained models: https://github.com/facebook/fb.resnet.torch

Pre-trained models: https://github.com/facebook/fb.resnet.torch/tree/master/pretrained

Instructions to fine-tune: https://github.com/facebook/fb.resnet.torch/tree/master/pretrained#fine-tuning-on-a-custom-dataset

Cheatsheet

Cheatsheet: https://github.com/torch/torch7/wiki/Cheatsheet

Github: https://github.com/torch/torch7

Google Group for new users and installation queries:
https://groups.google.com/forum/embed/?place=forum%2Ftorch7#!forum/torch7

Advanced only https://gitter.im/torch/torch7

INTERESTED IN LEARNING MORE

Check out our Free Deep Learning Courses https://developer.nvidia.com/deep-learning-courses

Date	Class
#1	Introduction to Deep Learning
#2	Getting Started with DIGITS interactive training system for image classification
#3	Getting Started with the Caffe Framework
#4	Getting Started with the Theano Framework
#5	Getting Started with the Torch Framework