



Growing an open source ecosystem

Soumith Chintala
Facebook AI Research

Overview

- What is Torch?
- The Community
- Common use
- Core Philosophy
- Key drivers of adoption
- The Future



What is torch ?

- Interactive Scientific computing framework

Strings, numbers, tables - a tiny introduction

```
In [ ]: a = 'hello'  
In [ ]: print(a)  
In [ ]: b = {}  
In [ ]: b[1] = a  
In [ ]: print(b)  
In [ ]: b[2] = 30  
In [ ]: for i=1,#b do -- the # operator is the length operator in Lua  
           print(b[i])  
      end
```



What is torch ?

- Interactive Scientific computing framework

Tensors

```
In [ ]: a = torch.Tensor(5,3) -- construct a 5x3 matrix, uninitialized
```

```
In [ ]: a = torch.rand(5,3)
print(a)
```

```
In [ ]: b=torch.rand(3,4)
```

```
In [ ]: -- matrix-matrix multiplication: syntax 1
a*b
```

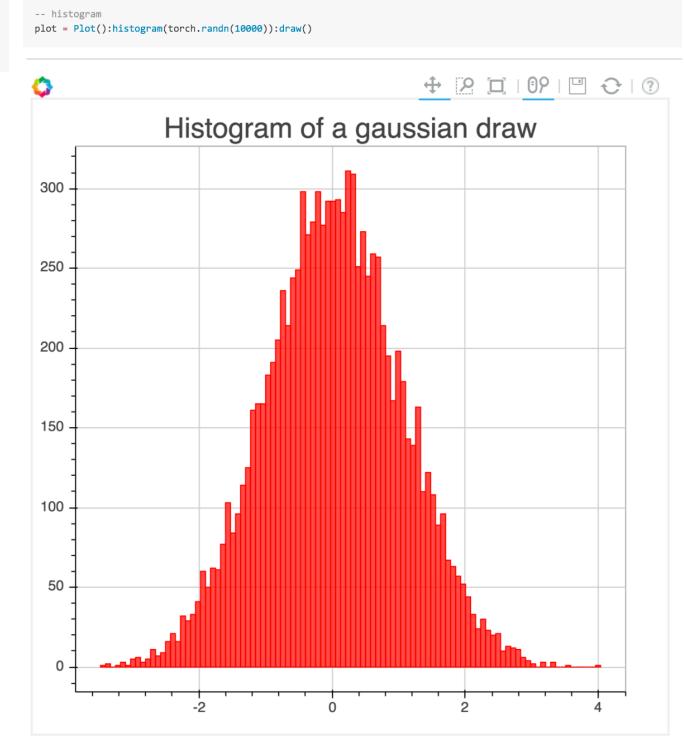
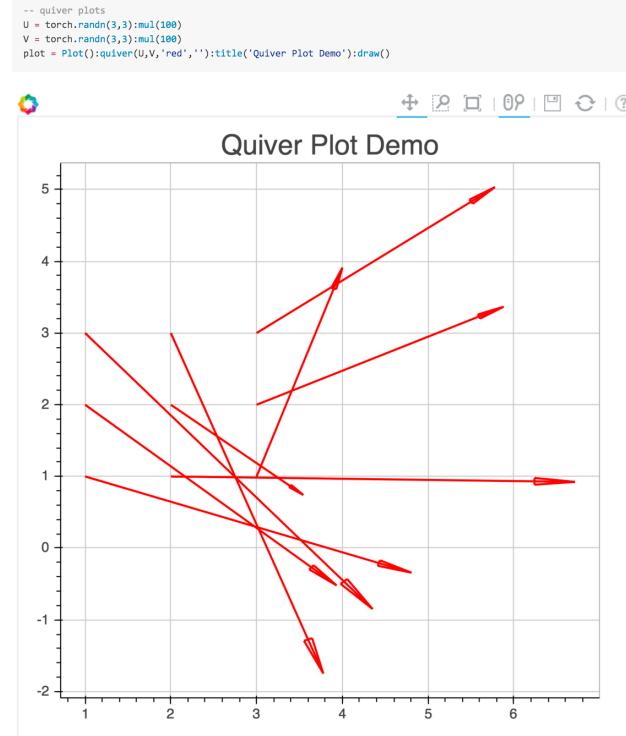
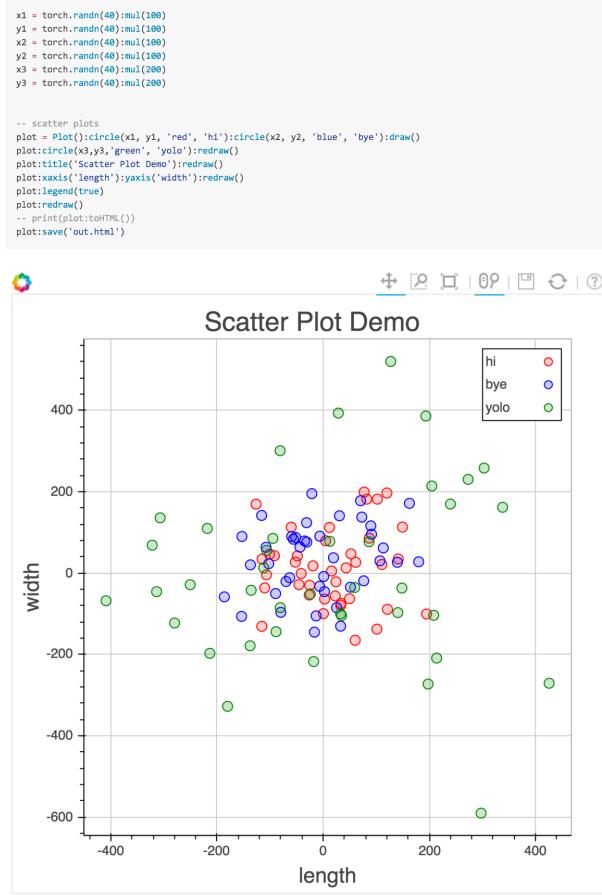
```
In [ ]: -- matrix-matrix multiplication: syntax 2
torch.mm(a,b)
```

```
In [ ]: -- matrix-matrix multiplication: syntax 3
c=torch.Tensor(5,4)
c:mm(a,b) -- store the result of a*b in c
```



What is torch ?

- Similar to Matlab / Python+Numpy



What is torch ?

- Little language overhead compared to Python / Matlab
- JIT compilation via LuaJIT
 - Fearlessly write for-loops

Code snippet from a core package

```
function NarrowTable:updateOutput(input)
    for k,v in ipairs(self.output) do self.output[k] = nil end
    for i=1,self.length do
        self.output[i] = input[self.offset+i-1]
    end
    return self.output
end
```



What is torch ?

- Easy integration into and from C
- Example: using CuDNN functions

```
for g = 0, self.groups - 1 do
    errcheck('cudnnConvolutionForward', cudnn.getHandle(),
              one:data(),
              self.iDesc[0], input:data() + g*self.input_offset,
              self.weightDesc[0], self.weight:data() + g*self.weight_offset,
              self.convDesc[0], self.fwdAlgType[0],
              self.extraBuffer:data(), self.extraBufferSizeInBytes,
              zero:data(),
              self.oDesc[0], self.output:data() + g*self.output_offset);
end
```



What is torch ?

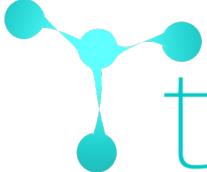
- Strong GPU support

CUDA Tensors

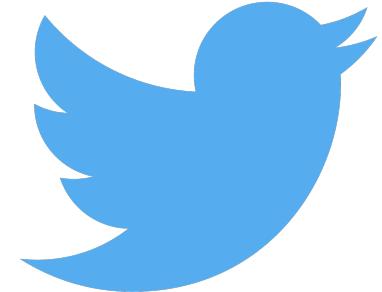
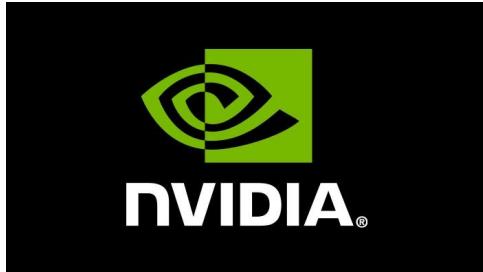
Tensors can be moved onto GPU using the :cuda function

```
In [ ]: require 'cutorch';
          a = a:cuda()
          b = b:cuda()
          c = c:cuda()
          c:mm(a,b) -- done on GPU
```





torch Community



canary

Inria

 **Stanford**
University

PURDUE
UNIVERSITY



Etsy

Yandex

element™

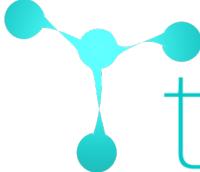


AMD

IBM®

MULTICORE
WARE

 **Moodstocks**



torch Community

 [szagoruyko / loadcaffe](#)  Watch ▾ 19  Unstar 203  Fork 69

 [Code](#)  [Issues 10](#)  [Pull requests 1](#)  [Wiki](#)  [Pulse](#)  [Graphs](#)

Load Caffe networks in Torch7

 [facebook / fb.resnet.torch](#)  Unwatch ▾ 62  Unstar 421  Fork 85

 [Code](#)  [Issues 4](#)  [Pull requests 0](#)  [Wiki](#)  [Pulse](#)  [Graphs](#)

Torch implementation of ResNet from <http://arxiv.org/abs/1512.03385> and training scripts

 [Moodstocks / inception-v3.torch](#)  Watch ▾ 11  Unstar 48  Fork 8

 [Code](#)  [Issues 1](#)  [Pull requests 0](#)  [Wiki](#)  [Pulse](#)  [Graphs](#)

Rethinking the Inception Architecture for Computer Vision <http://arxiv.org/abs/1512.00567>



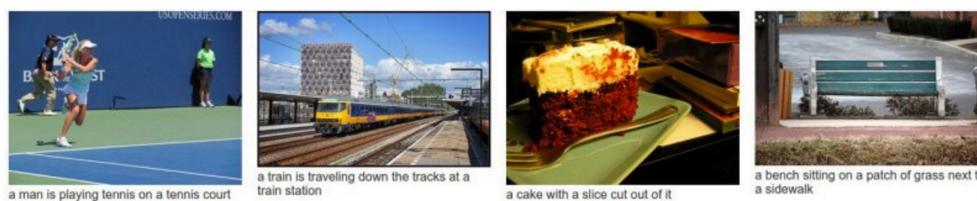


NeuralTalk2

Recurrent Neural Network captions your images. Now much faster and better than the original [NeuralTalk](#). Compared to the original NeuralTalk this implementation is **batched, uses Torch, runs on a GPU, and supports CNN finetuning**. All of these together result in quite a large increase in training speed for the Language Model (~100x), but overall not as much because we also have to forward a VGGNet. However, overall very good models can be trained in 2-3 days, and they show a much better performance.

This is an early code release that works great but is slightly hastily released and probably requires some code reading of inline comments (which I tried to be quite good with in general). I will be improving it over time but wanted to push the code out there because I promised it to too many people.

This current code (and the pretrained model) gets ~0.9 CIDEr, which would place it around spot #8 on the [codalab leaderboard](#). I will submit the actual result soon.

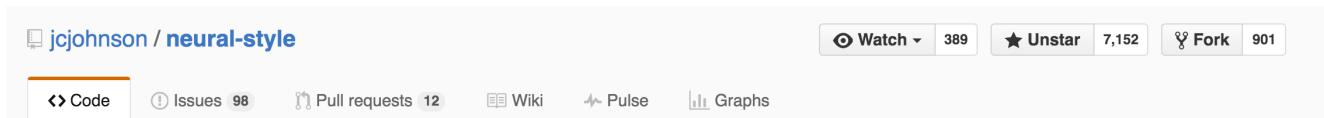


You can find a few more example results on the [demo page](#). These results will improve a bit more once the last few bells and whistles are in place (e.g. beam search, ensembling, reranking).

There's also a [fun video](#) by [@kcimc](#), where he runs a neuraltalk2 pretrained model in real time on his laptop during a walk in Amsterdam.



torch Community

A screenshot of a GitHub repository card for `jcjohnson / neural-style`. The card shows basic repository statistics: Watch (389), Unstar (7,152), Fork (901). Below the stats are navigation links for Code, Issues (98), Pull requests (12), Wiki, Pulse, and Graphs.

Torch implementation of neural style algorithm

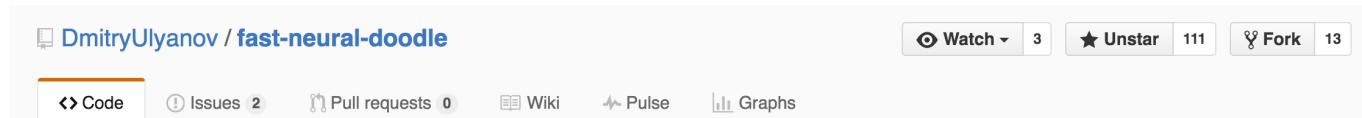
neural-style

This is a torch implementation of the paper [A Neural Algorithm of Artistic Style](#) by Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge.

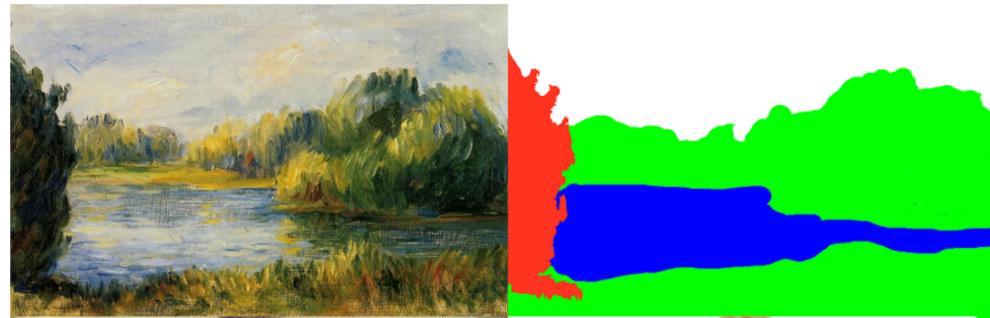
The paper presents an algorithm for combining the content of one image with the style of another image using convolutional neural networks. Here's an example that maps the artistic style of [The Starry Night](#) onto a night-time photograph of the Stanford campus:



torch Community



Faster neural doodle

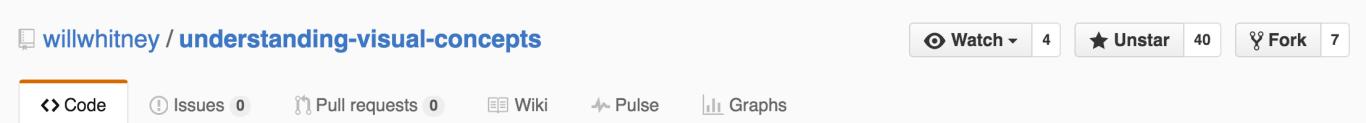


A screenshot of a GitHub repository page. At the top left, it says "facebook / UETorch". On the top right, there are buttons for "Unwatch" (37), "Unstar" (135), and "Fork" (20). Below the header, there's a navigation bar with tabs: "Code" (which is selected and highlighted in orange), "Issues" (1), "Pull requests" (0), "Wiki", "Pulse", and "Graphs".

A Torch plugin for Unreal Engine 4.



torch Community

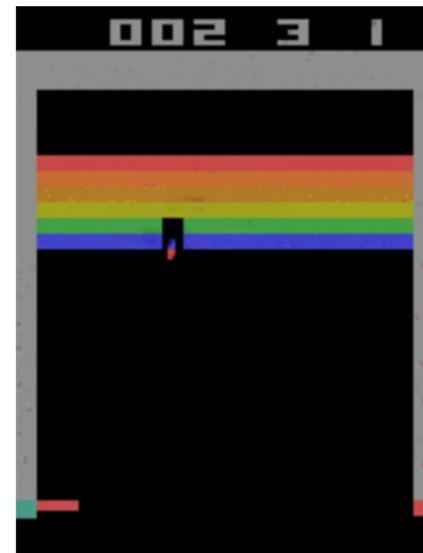
[willwhitney / understanding-visual-concepts](#) Watch 4 Unstar 40 Fork 7

Code Issues 0 Pull requests 0 Wiki Pulse Graphs

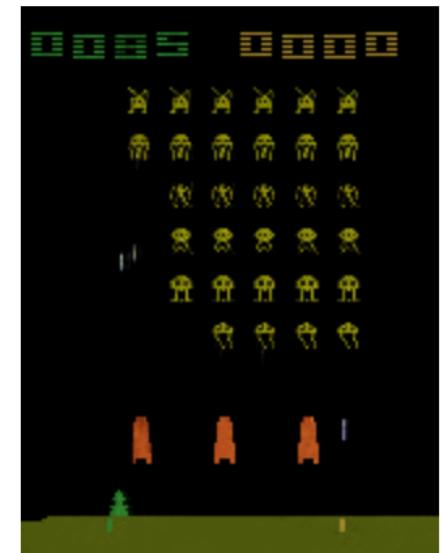
Unsupervised learning of visual concepts from video <http://willwhitney.github.io/understanding-visual-concepts/>



Moving the paddle.



Counting remaining lives.



Animating the aliens.



A screenshot of a GitHub repository page. At the top, it shows the repository name 'Kaixin / rlenvs'. Below the name are buttons for 'Code', 'Issues 0', 'Pull requests 0', 'Wiki', 'Pulse', and 'Graphs'. To the right of these buttons are links for 'Watch 6', 'Unstar 32', 'Fork 4', and a 'Settings' gear icon. The main content area below the header contains the text 'Reinforcement learning environments for Torch7'.

Reinforcement learning environments for Torch7, inspired by RL-Glue [1]. Supported environments:

- rlenvs.Acrobot [2]
- rlenvs.Atari (Arcade Learning Environment)* [3]
- rlenvs.Blackjack [4]
- rlenvs.CartPole [5]
- rlenvs.Catch [6]
- rlenvs.CliffWalking [7]
- rlenvs.DynaMaze [8]
- rlenvs.GridWorld [9]
- rlenvs.JacksCarRental [7]
- rlenvs.MountainCar [10]
- rlenvs.MultiArmedBandit [11, 12]
- rlenvs.RandomWalk [13]
- rlenvs.Taxi [14]
- rlenvs.WindyWorld [7]





Neural Conversational Model in Torch

This is an attempt at implementing [Sequence to Sequence Learning with Neural Networks \(seq2seq\)](#) and reproducing the results in [A Neural Conversational Model](#) (aka the Google chatbot).

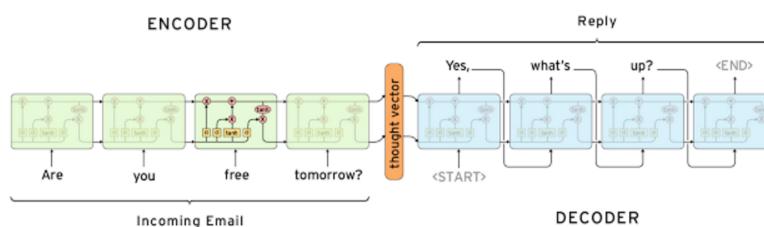
The Google chatbot paper [became famous](#) after cleverly answering a few philosophical questions, such as:

Human: What is the purpose of living?

Machine: To live forever.

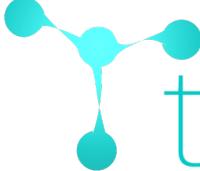
How it works

The model is based on two [LSTM](#) layers. One for encoding the input sentence into a "thought vector", and another for decoding that vector into a response. This model is called Sequence-to-sequence or seq2seq.



Source: <http://googleresearch.blogspot.ca/2015/11/computer-respond-to-this-email.html>

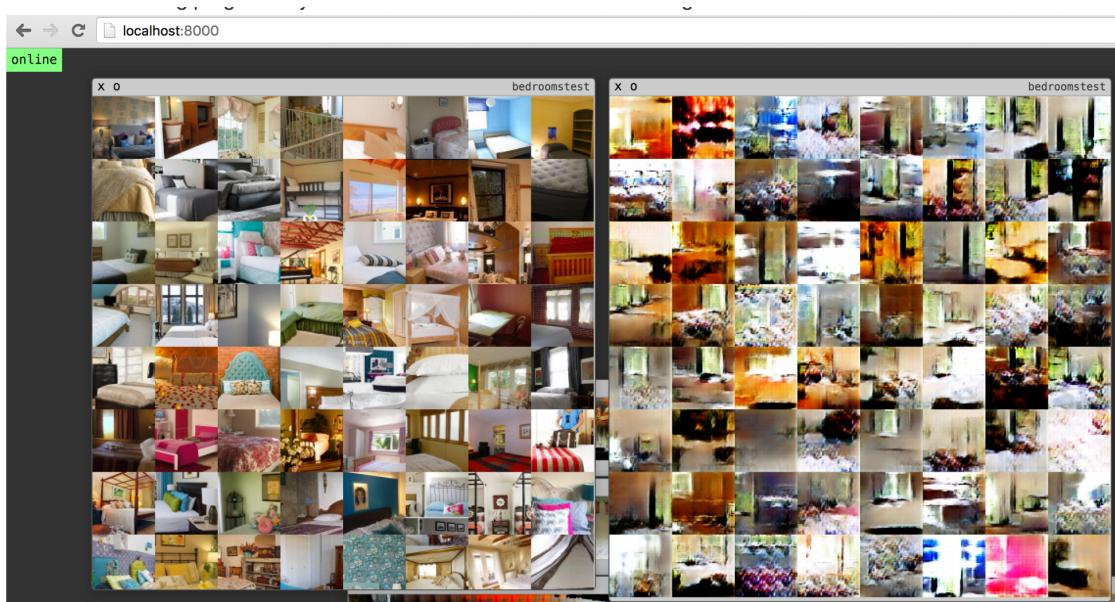




torch Community



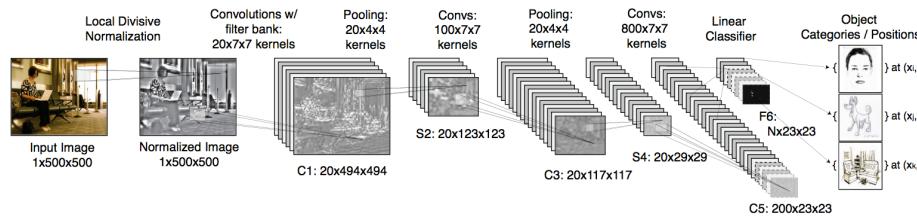
A torch implementation of <http://arxiv.org/abs/1511.06434> — Edit



Neural Networks

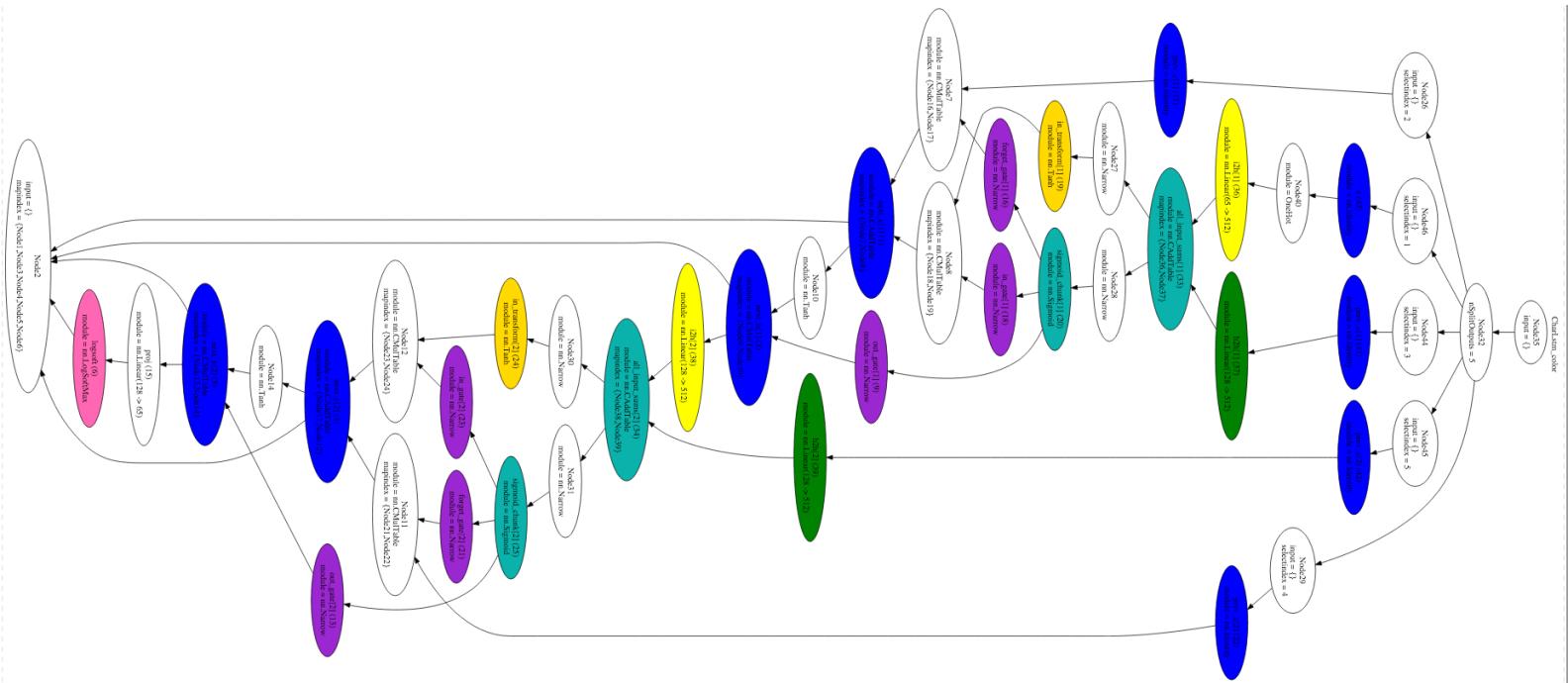
- nn: neural networks made easy
- building blocks of differentiable modules
 - define a model with pre-normalization, to work on raw RGB images:

```
01 model = nn.Sequential()  
02  
03 model.add(nn.SpatialConvolution(3,16,5,5))  
04 model.add(nn.Tanh())  
05 model.add(nn.SpatialMaxPooling(2,2,2,2))  
06 model.add(nn.SpatialContrastiveNormalization(16, image.gaussian(3)))  
07  
08 model.add(nn.SpatialConvolution(16,64,5,5))  
09 model.add(nn.Tanh())  
10 model.add(nn.SpatialMaxPooling(2,2,2,2))  
11 model.add(nn.SpatialContrastiveNormalization(64, image.gaussian(3)))  
12  
13 model.add(nn.SpatialConvolution(64,256,5,5))  
14 model.add(nn.Tanh())  
15 model.add(nn.Reshape(256))  
16 model.add(nn.Linear(256,10))  
17 model.add(nn.LogSoftMax())
```



Advanced Neural Networks

- nngraph
 - easy construction of complicated neural networks



autograd by

- Write imperative programs
- Backprop defined for every operation in the language

```
neuralNet = function(params, x, y)
    local h1 = t.tanh(x * params.W[1] + params.b[1])
    local h2 = t.tanh(h1 * params.W[2] + params.b[2])
    local yHat = h2 - t.log(t.sum(t.exp(h2)))
    local loss = - t.sum(t.cmul(yHat, y))
    return loss
end

-- gradients:
dneuralNet = grad(neuralNet)

-- some data:
x = t.randn(1,100)
y = t.Tensor(1,10):zero() y[1][3] = 1

-- compute loss and gradients wrt all parameters in params:
dparams, loss = dneuralNet(params, x, y)
```



Distributed Learning

- in-built multi-GPU (data and model parallel)
- distlearn by 
- multi-node parallelism

```
-- Use a tau of 10 and an alpha of 0.2
local allReduceEA = require 'distlearn.AllReduceEA'(tree, 10, 0.2)
-- Make sure all the nodes start with the same parameter values
allReduceEA.synchronizeParameters(params)
for _ = 1,epochs do
    for _ = 1,steps
        -- Compute your gradients as normal
        local grads = computeYourGrads(...)
        -- Do your SGD as normal
        SGD(params, grads)
        -- Average the params
        allReduceEA.averageParameters(params)
    end
    -- Make sure the center's haven't drifted too far due to
    -- floating point precision error build up
    allReduceEA.synchronizeCenter(params)
    -- Validate...
end
```



Core Philosophy

- Interactive computing
 - No compilation time
- Imperative programming
 - Write code like you always did, not computation graphs in a hacked up DSL
- Minimal abstraction
 - Thinking linearly
- Maximal Flexibility
 - No constraints on interfaces or classes



Key Drivers of Adoption

- Tutorials and support
 - Pre-trained models
 - High-quality open-source projects
- Deeply integrated GPU goodness
- Minimal abstractions
- Imperative programming
- Zero compile-time



Key Drivers of Adoption

- Tutorials and support
 - Pre-trained models
 - High-quality open-source projects
- Deeply integrated **COMMUNITY!**
- Minimal abstractions
- Imperative programming
- Zero compile-time



The Future

- Fusing computation / compiler-stuff
- Continue breaking down barriers of entry
- Keeping focus on the long tail



The Future

