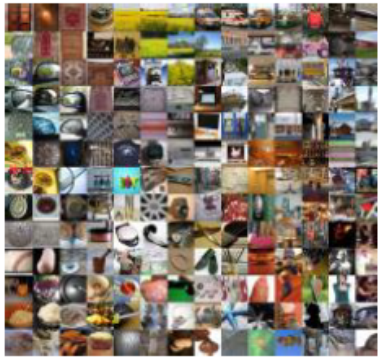
Louisiana State University

Introduction to Deep Learning and Its Applications

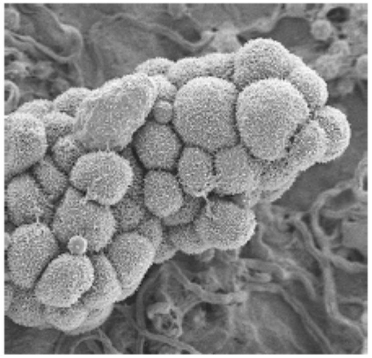
Mingxuan Sun
Assistant Professor in Computer Science
Louisiana State University
11/09/2016

DEEP LEARNING EVERYWHERE



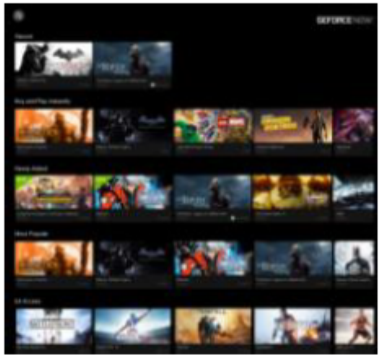
INTERNET & CLOUD

Image Classification
Speech Recognition
Language Translation
Language Processing
Sentiment Analysis
Recommendation



MEDICINE & BIOLOGY

Cancer Cell Detection
Diabetic Grading
Drug Discovery

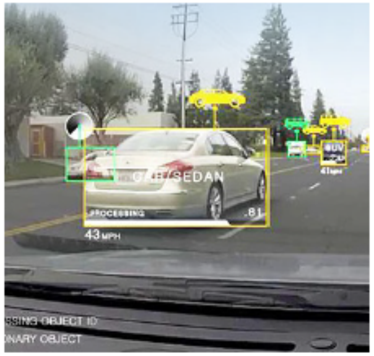


MEDIA & ENTERTAINMENT

Video Captioning
Video Search
Real Time Translation

SECURITY & DEFENSE

Face Detection
Video Surveillance
Satellite Imagery



AUTONOMOUS MACHINES

Pedestrian Detection
Lane Tracking
Recognize Traffic Sign

source:developer.nvidia.com/deep-learning-courses

Machine Learning

Input: X

Output: Y

Label"motorcycle"

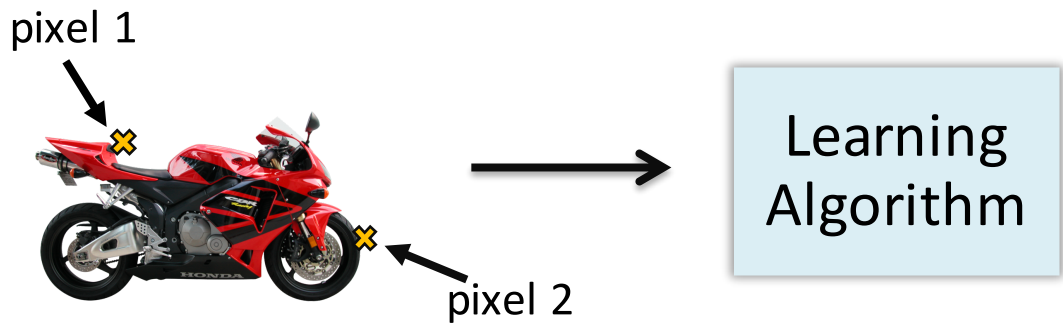
Why is it hard?

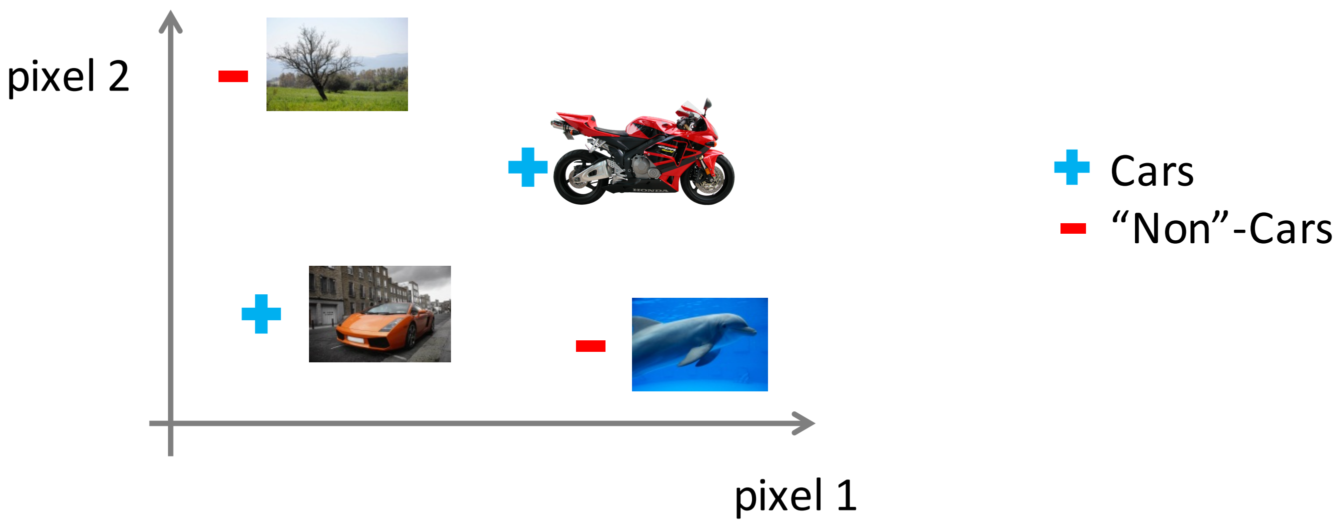
You see this

But the camera sees this:

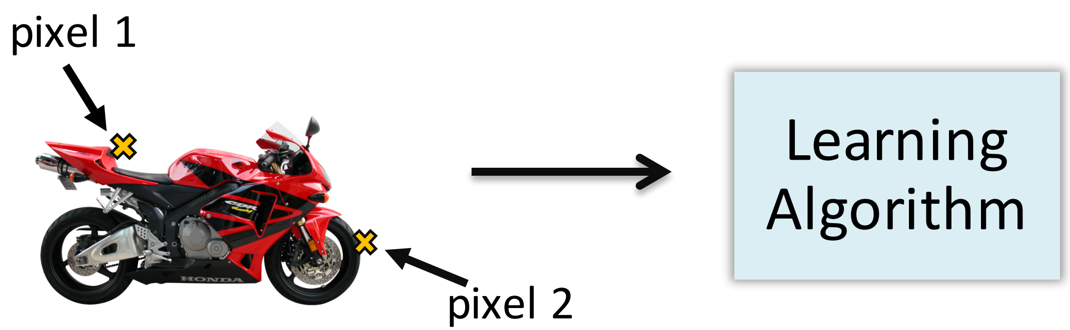
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

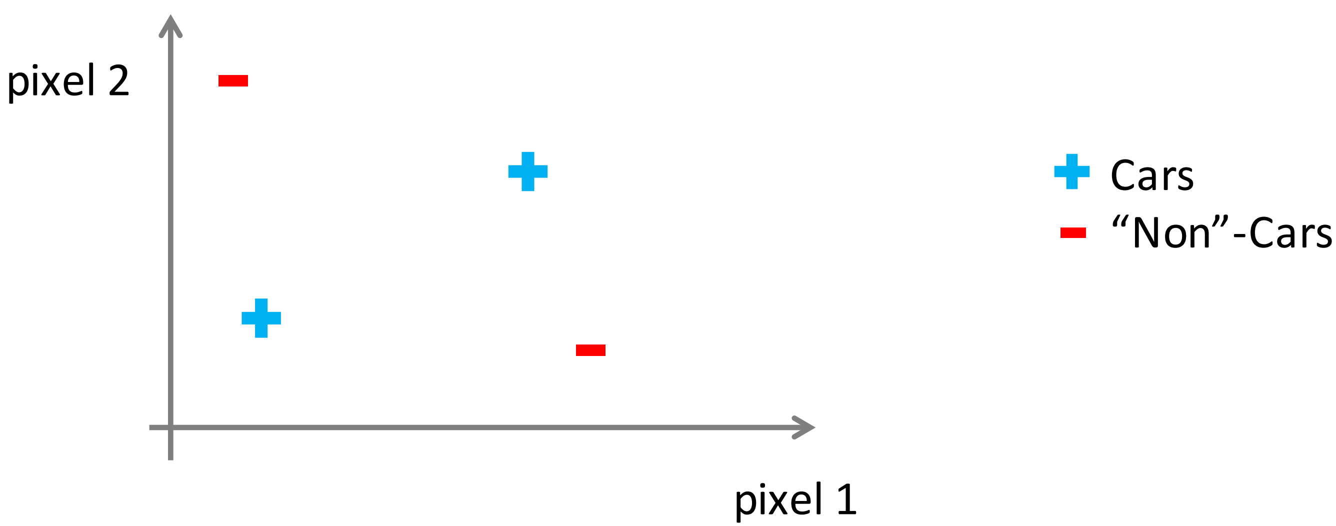
Raw Image Representation



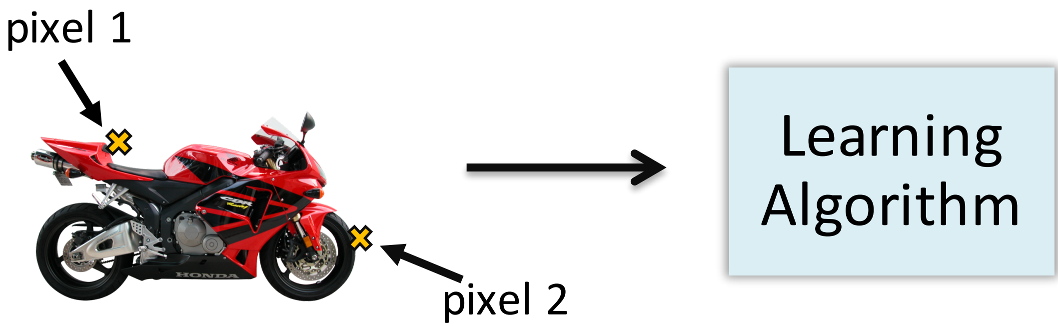


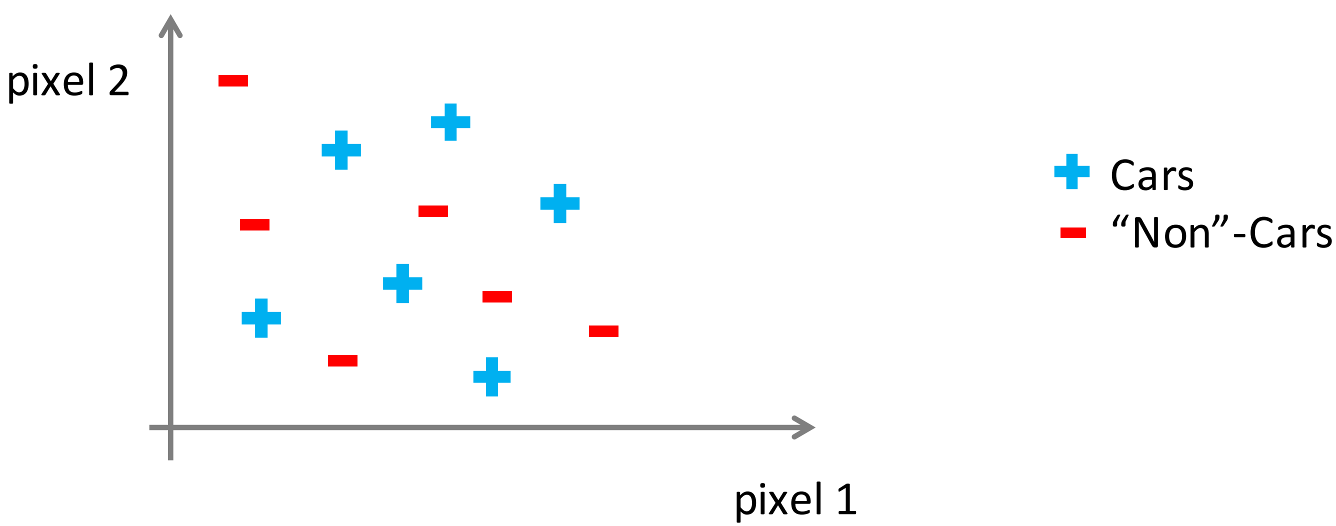
Raw Image Representation



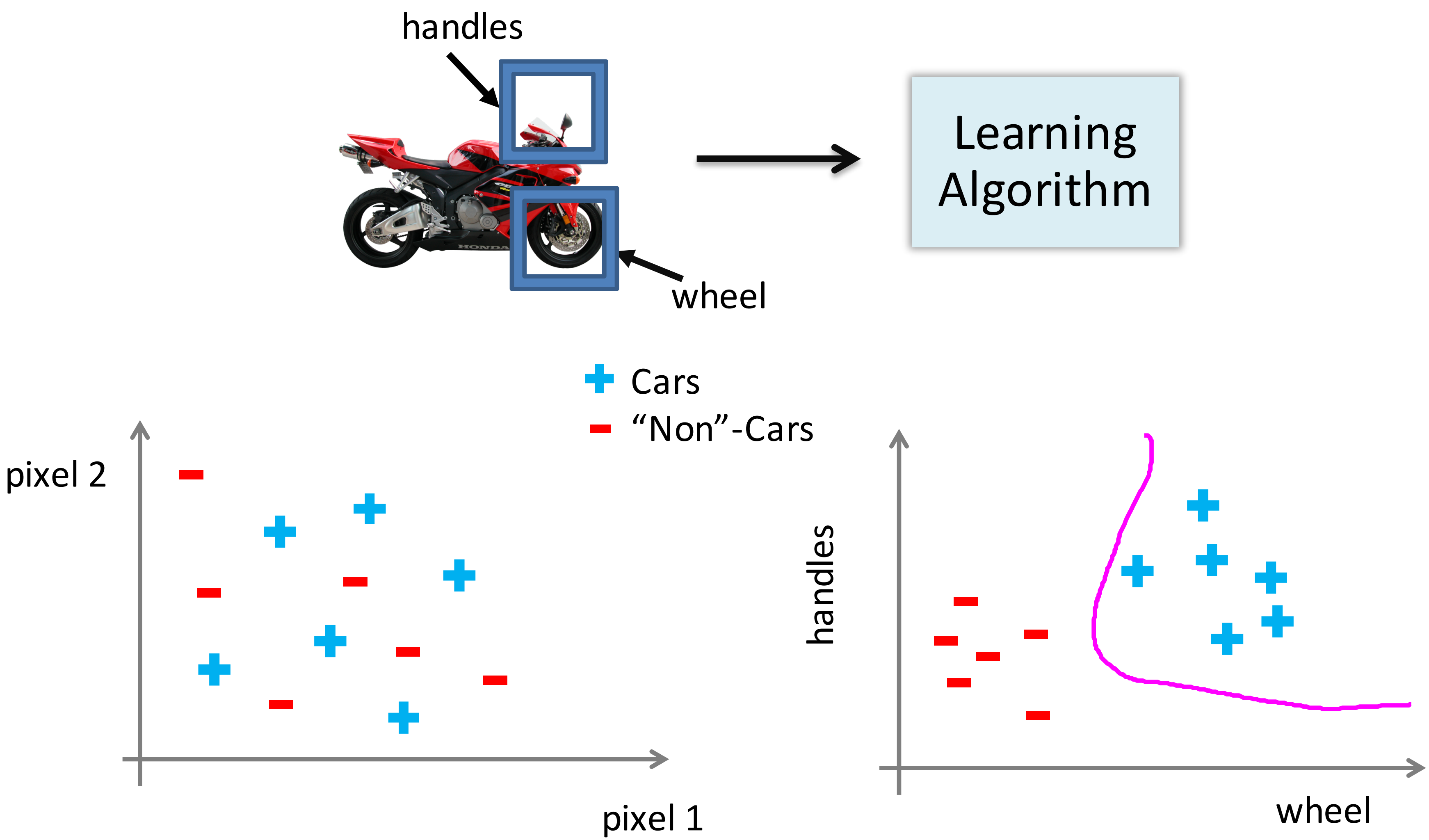


Raw Image Representation

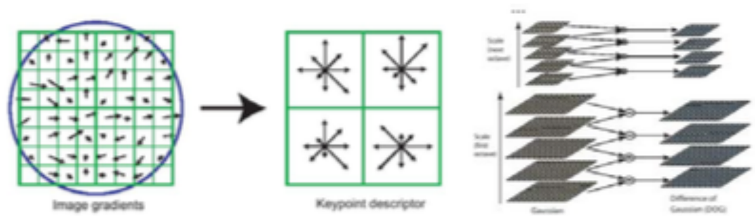




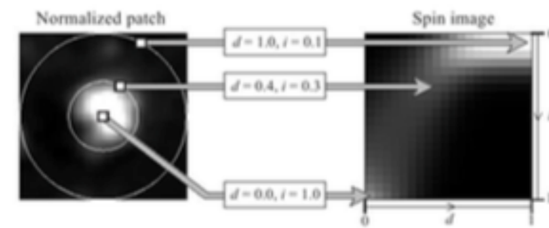
Better Feature Representation?



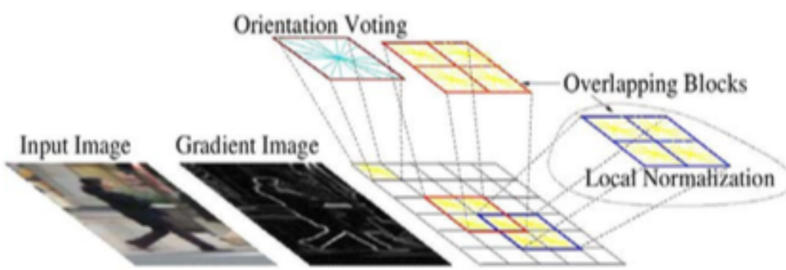
Feature Representations



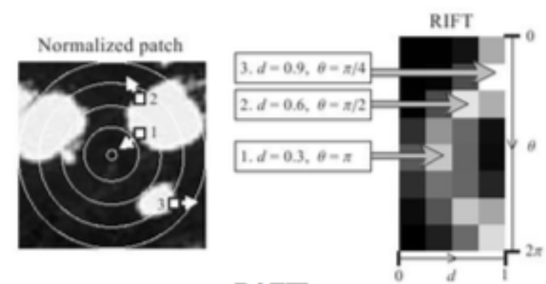
SIFT



Spin image

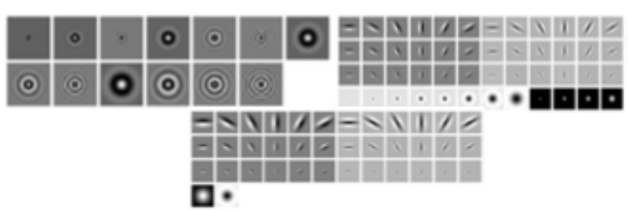


HoG

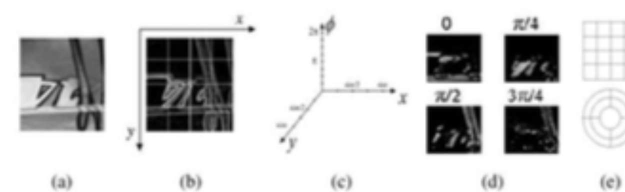


RIFT

Expert Knowledge!



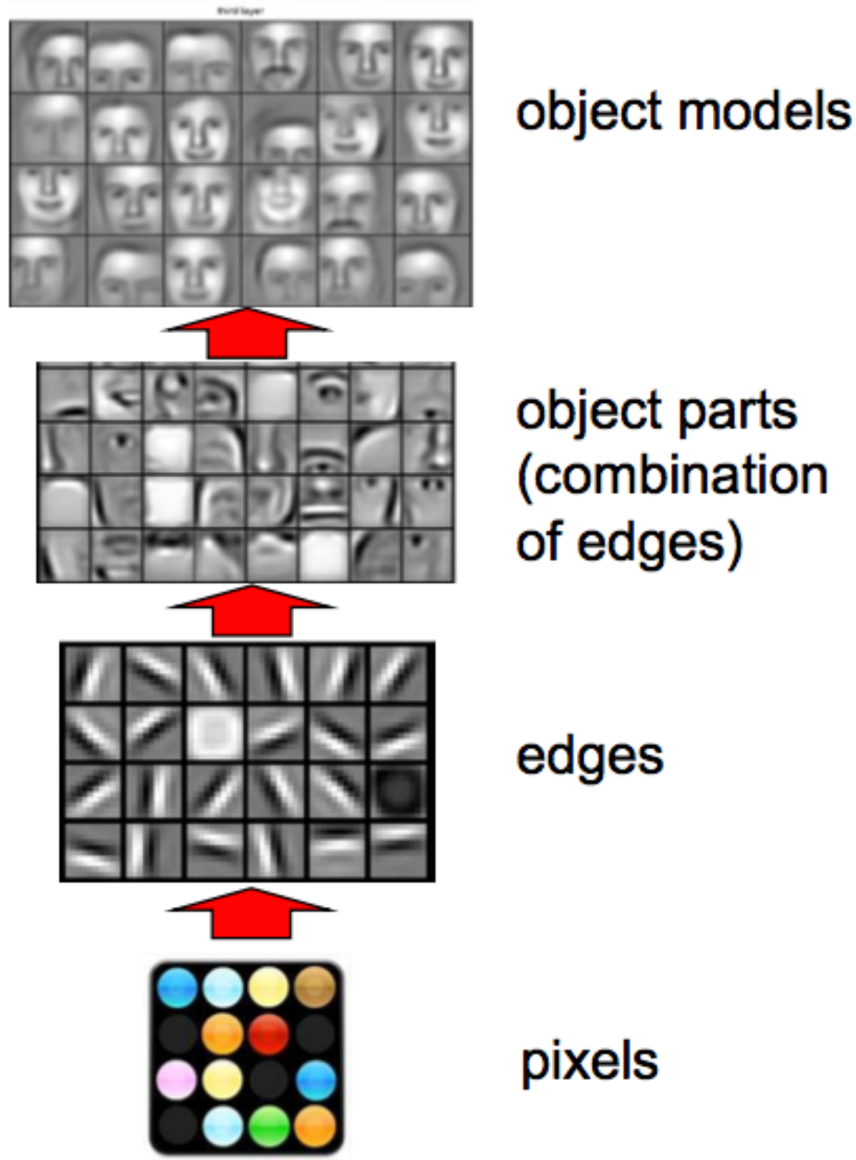
Textons



GLOH

Source: feature representations in computer vision(Honglak lee)

Deep Learning: learn representations!



Source:Lee et.al. ICML2009

So, 1. **what exactly is deep learning** ?

And, 2. **why is it generally better** than other methods on image, speech and certain other types of data?

The short answers

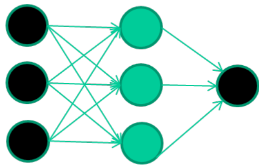
1. **'Deep Learning' means** using a **neural network** with **several layers of nodes** between input and output

2. **the series of layers between input & output do feature identification and processing in a series of stages, just as our brains seem to.**

Sources: https://www.macs.hw.ac.uk/~dwcorne/Teaching/**intro**dl.**ppt** hmmm… OK, but:

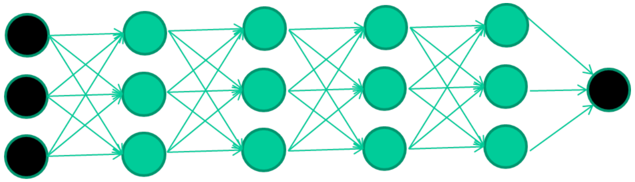
3. **multilayer neural networks have been around for 25 years. What's actually new?**

we have always had good algorithms for learning the weights in networks with 1 hidden layer



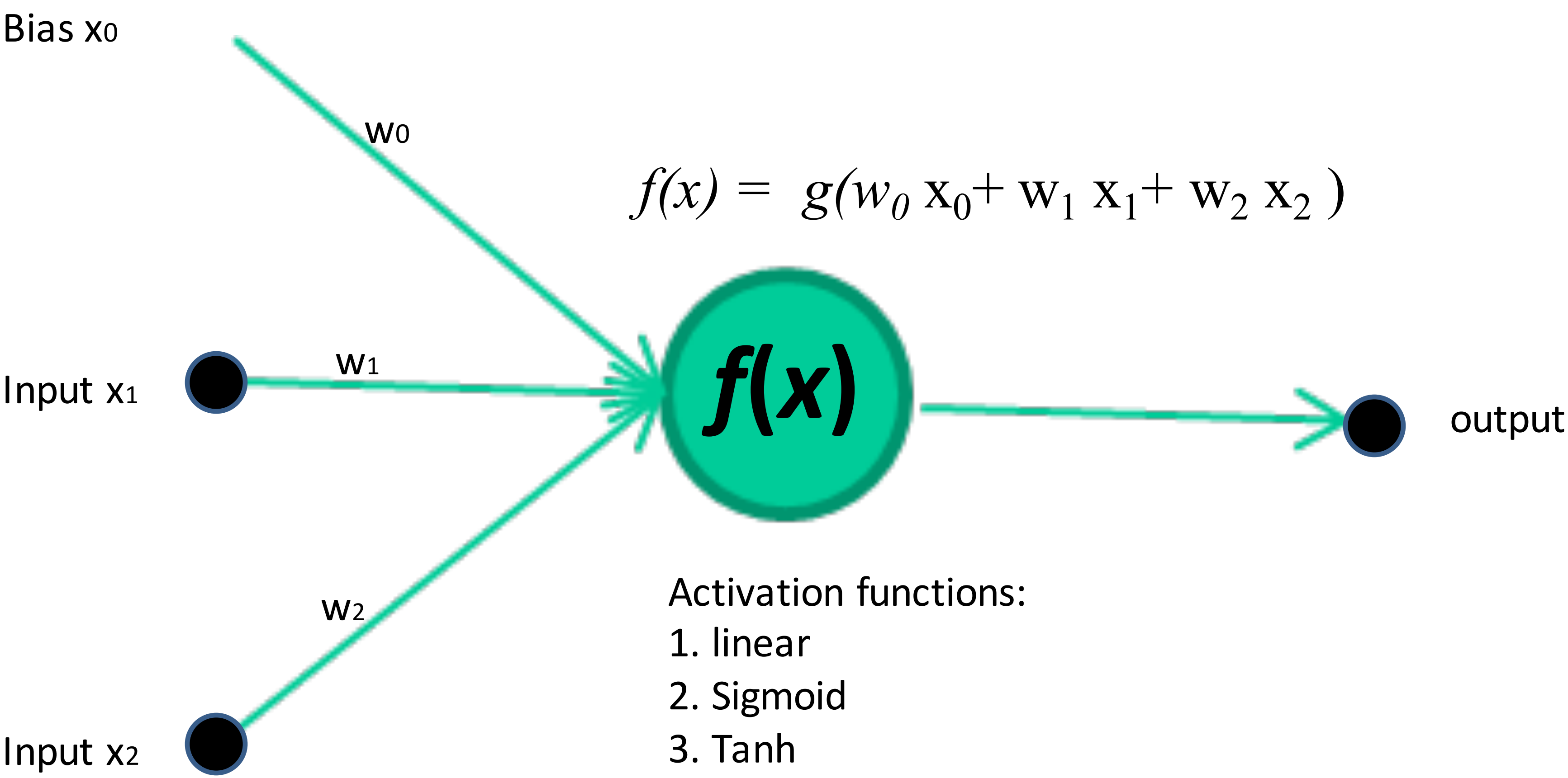
but these algorithms are not good at learning the weights for networks with more hidden layers

what's new is: <u>algorithms for training many-later networks</u>



https://www.macs.hw.ac.uk/~dwcorne/Teaching/**introdl.ppt**

Single Unit, Input, weights, activation function, output



$$f(x) = g(w_0\, x_0 + w_1\, x_1 + w_2\, x_2)$$

Activation functions:
1. linear
2. Sigmoid
3. Tanh
4. Relu
5. Softmax

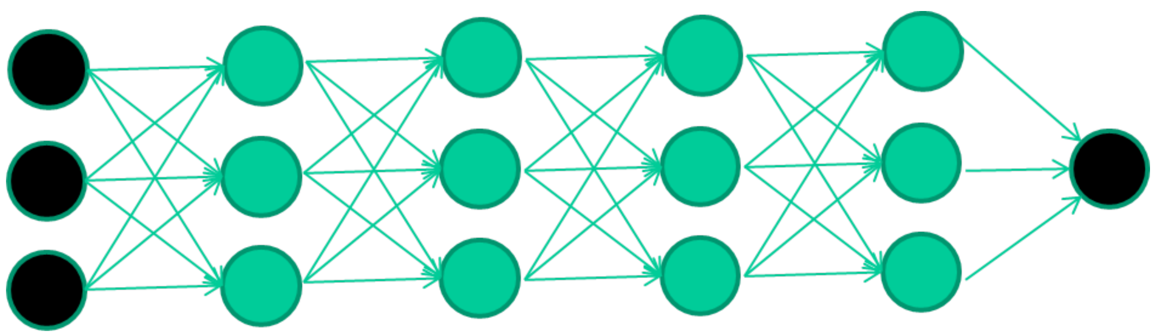
etc.

https://www.macs.hw.ac.uk/~dwcorne/Teaching/**intro**dl.**ppt**

A dataset

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Train the deep neural network

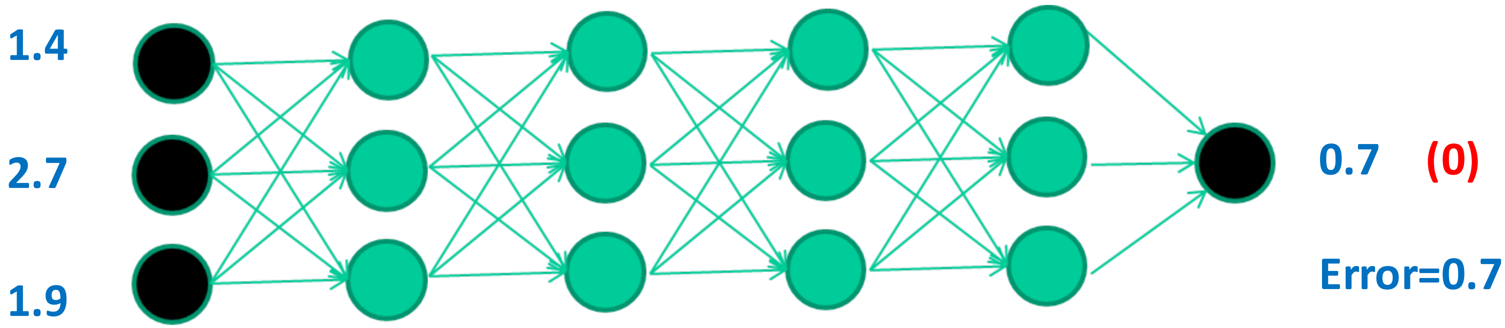


https://www.macs.hw.ac.uk/~dwcorne/Teaching/**intro**dl.**ppt**

A dataset

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

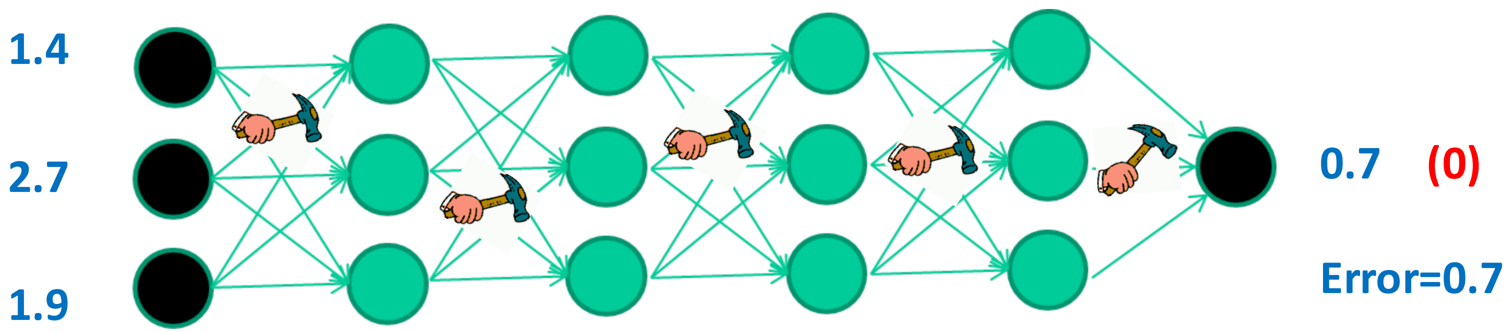
Initialize with random weights



Compare with the target output

https://www.macs.hw.ac.uk/~dwcorne/Teaching/**intro**dl.**ppt**

Adjust weights based on error

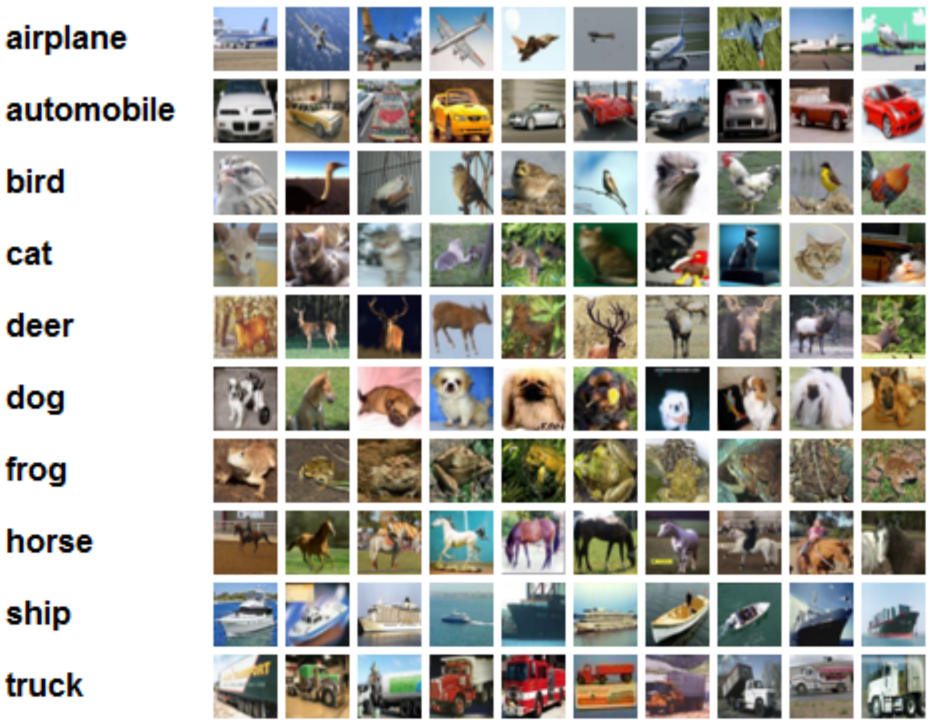


Repeat this thousands, maybe millions of times – each time taking a random training instance, and making slight weight adjustments

Algorithms for weight adjustment are designed to make changes that will reduce the error

https://www.macs.hw.ac.uk/~dwcorne/Teaching/**intro**dl.**ppt**

CIFAR 10 and Convolutional Neural Network



CIFAR 10 dataset:
50,000 training images
10,000 testing images
10 categories (classes)

Accuracies from different methods:

Human: ~94%

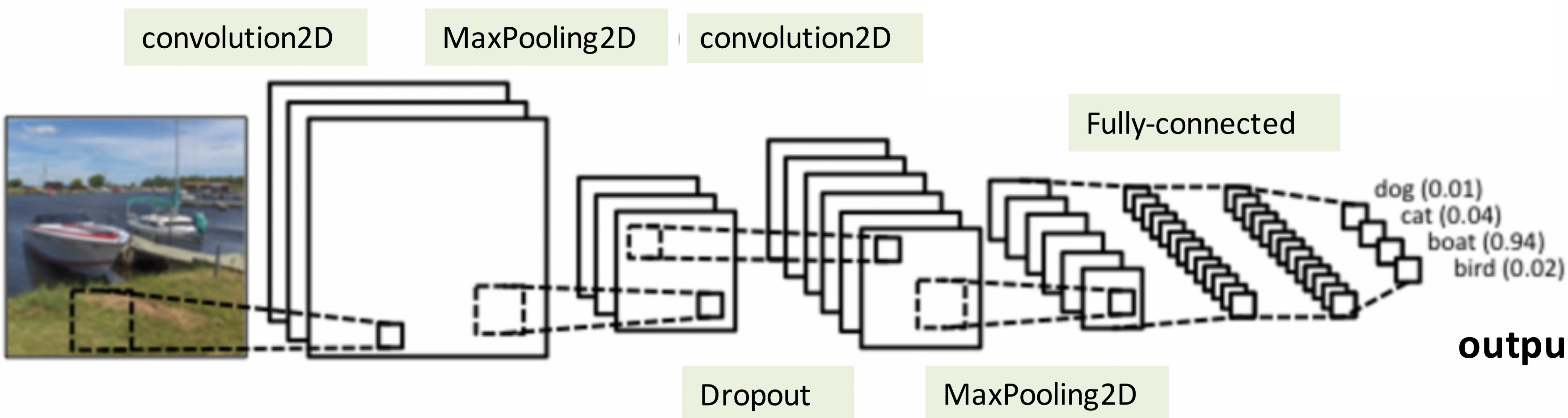
Whitening K-mean: 80%

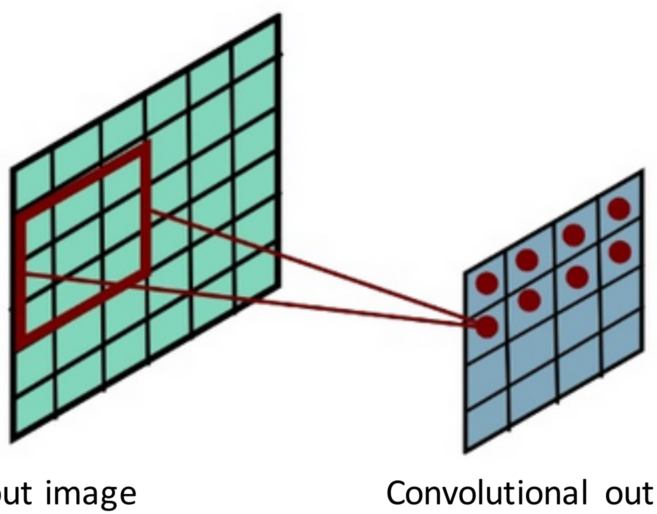
……

Deep CNN: 95.5%

http://karpathy.github.io/2011/04/27/manually-classifying-cifar10/
https://www.kaggle.com/c/cifar-10/leaderboard

Deep Convolutional Neural Networks on CIFAR10

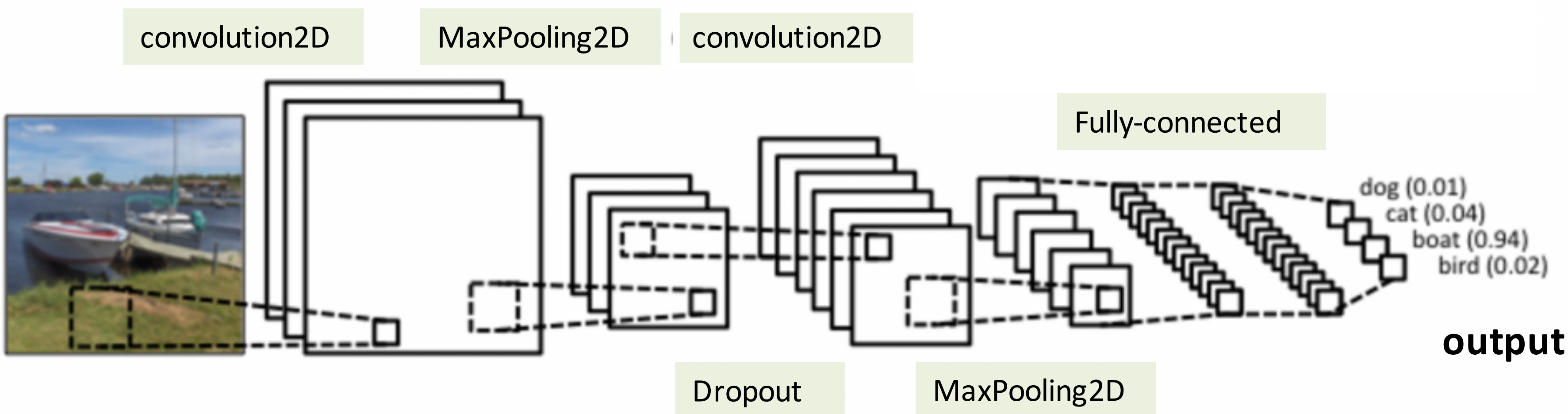


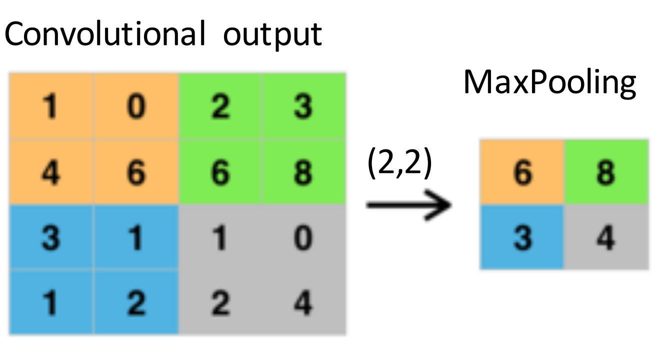


Convolutional Layer: filters work on every part of the image, therefore, they are searching for the same feature everywhere in the image.

http://www.slideshare.net/perone/deep-learning-convolutional-neural-networks

Deep Convolutional Neural Networks on CIFAR10

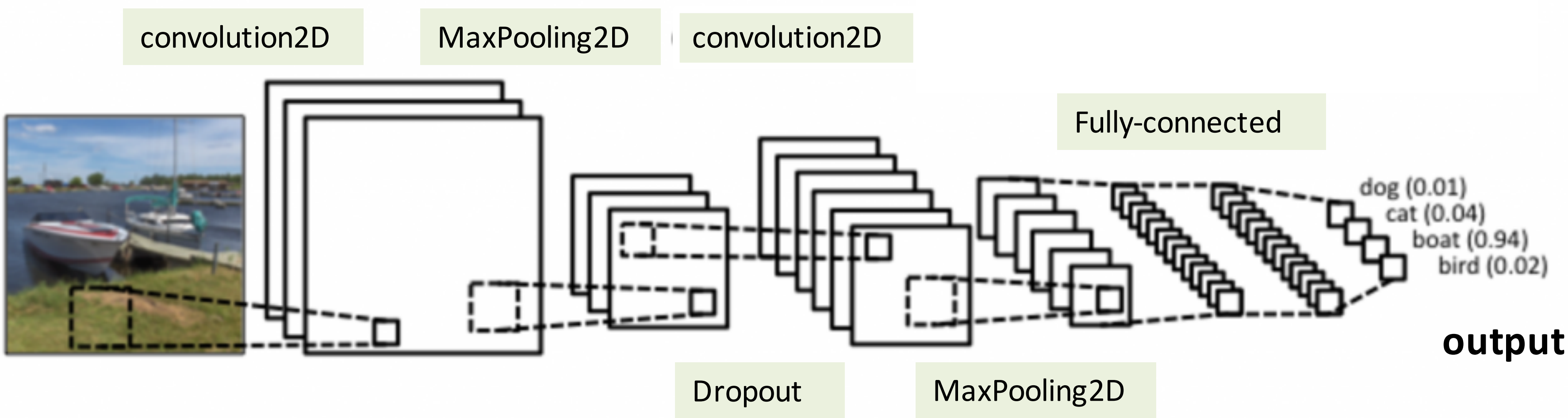


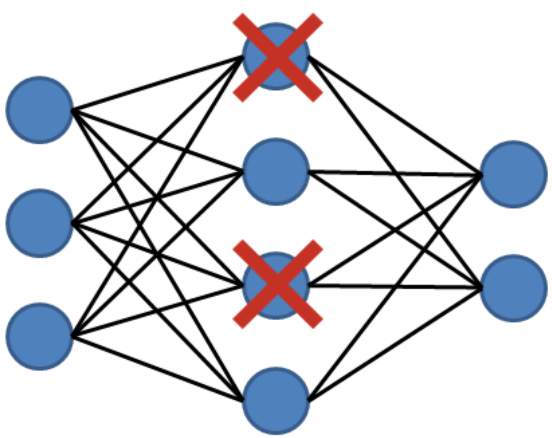


MaxPooling: usually present after the convolutional layer. It provides a down-sampling of the convolutional output

http://www.slideshare.net/perone/deep-learning-convolutional-neural-networks

Deep Convolutional Neural Networks on CIFAR10

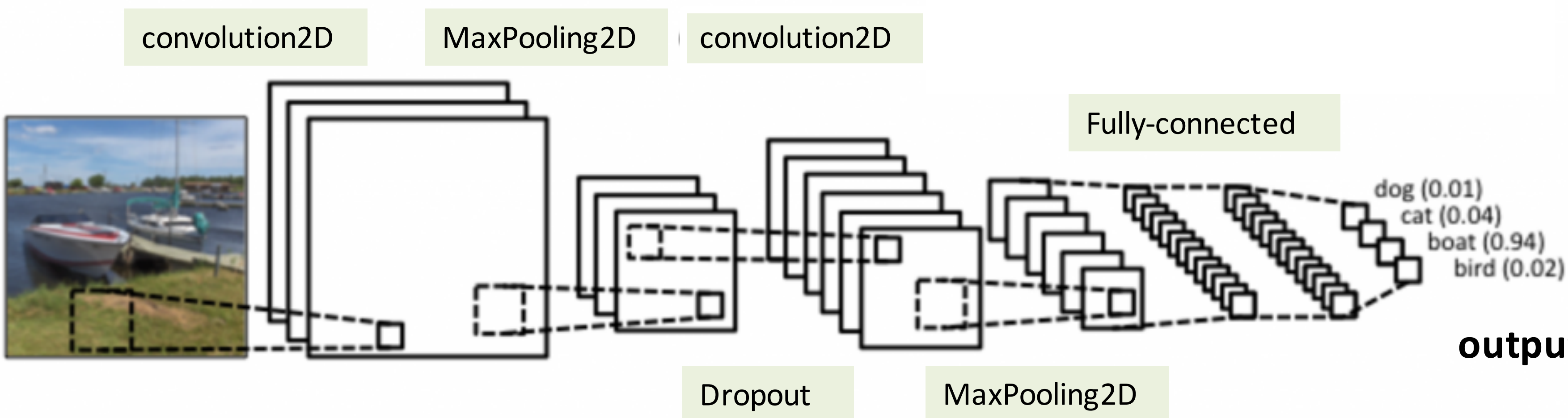


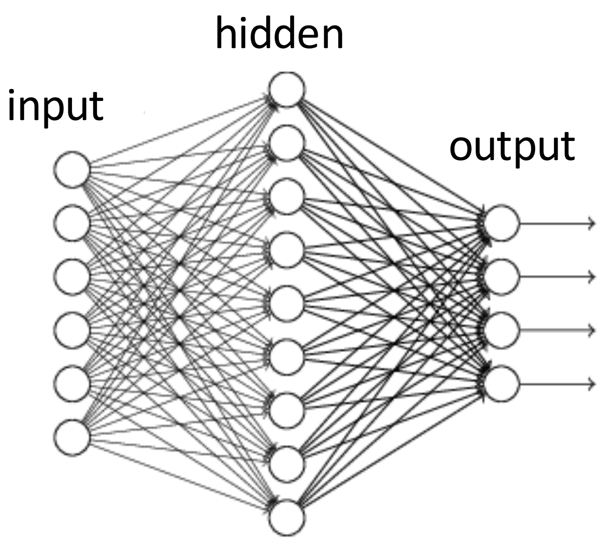


Dropout: randomly drop units along with their connections during training. It helps to learn more robust features by reducing complex co-adaptations of units and alleviate overfitting issue as well.

Srivastava et al. Dropout: A Simple way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15(2014):1929-1958

Deep Convolutional Neural Networks on CIFAR10

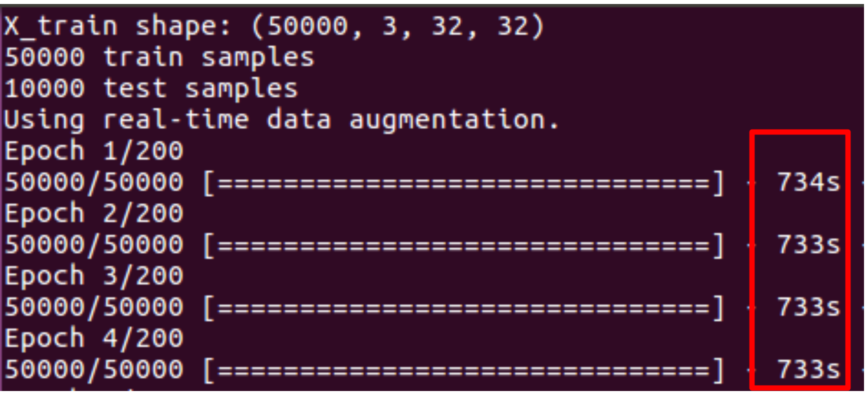




Fully-connected layer (dense): each node is fully connected to all input nodes, each node computes weighted sum of all input nodes. It has one-dimensional structure. It helps to classify input pattern with high-level features extracted by previous layers.

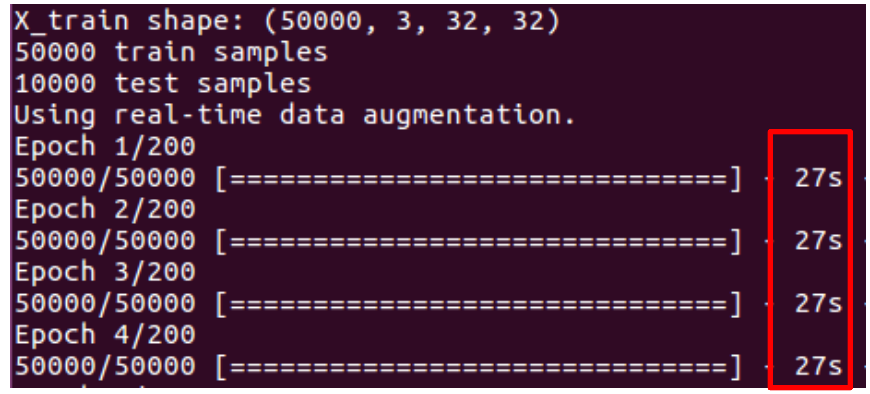
http://www.slideshare.net/datasciencekorea/5-20141107deeplearning

Why GPU Matters in Deep Learning?



Running time **without** GPU

VS



Running time **with** GPU

With GPU, the running time is 733/27=**27.1 times faster** then the running time without GPU!!!

Again, WHY GPUs?

1. Every set of weights can be stored as a matrix (m,n)
2. GPUs are made to do common parallel problems fast. All similar calculations are done at the same time. This extremely boosts the performance in parallel computations.

Blankevoort T., Neural Networks and Deep Learning, slides

Summary: 2010s Deep Learning

- Make it deep (many layers)

- Way more labeled data (1 million)

- A lot better computing power (GPU clusters)

THANK YOU VERY MUCH!

Louisiana State University

Deep Learning For Recommender Systems

Fei Li
M.S. Student in Computer Science
Supervisor: Dr. Mingxuan Sun
Louisiana State University
11/09/2016

Recommender System

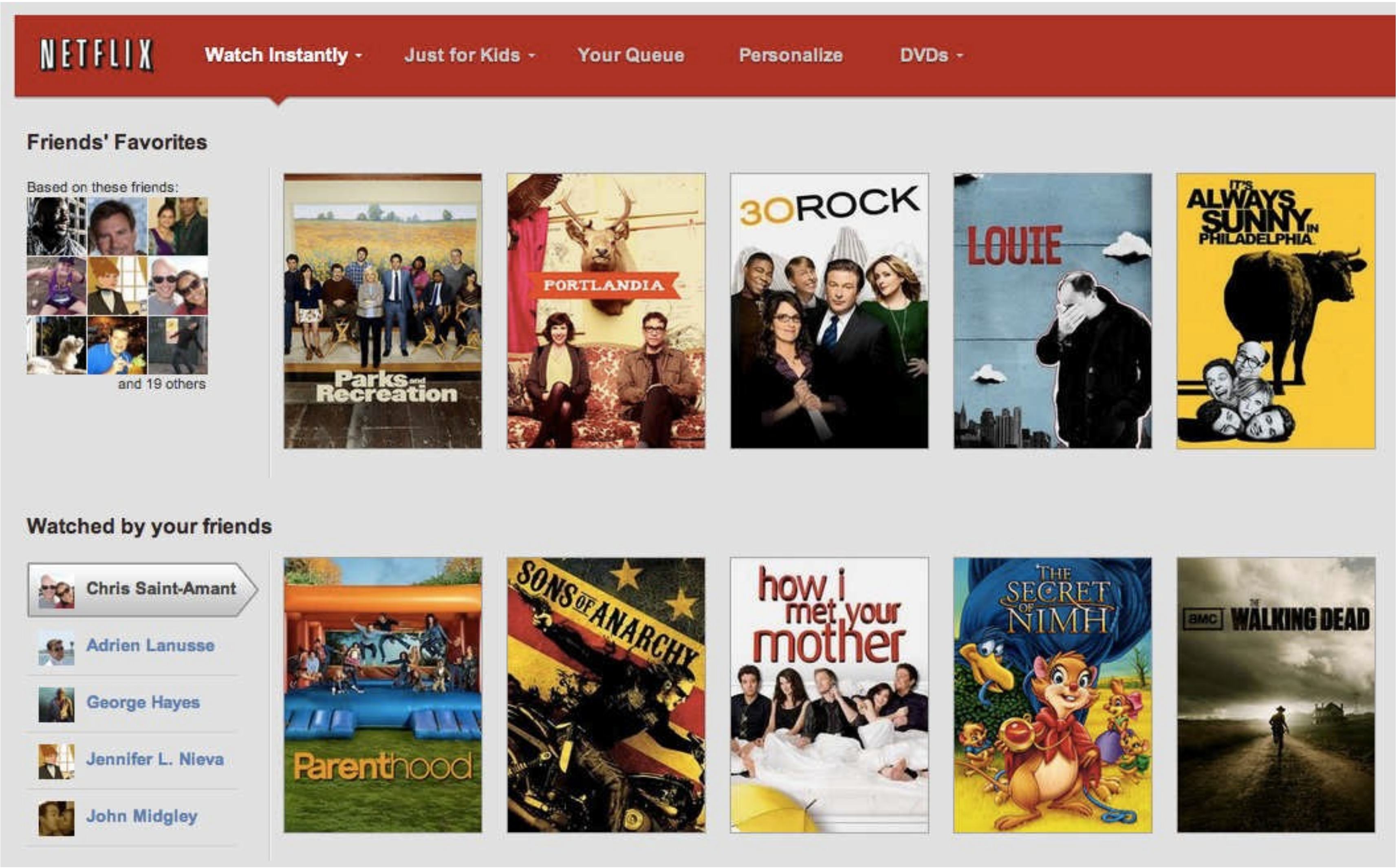
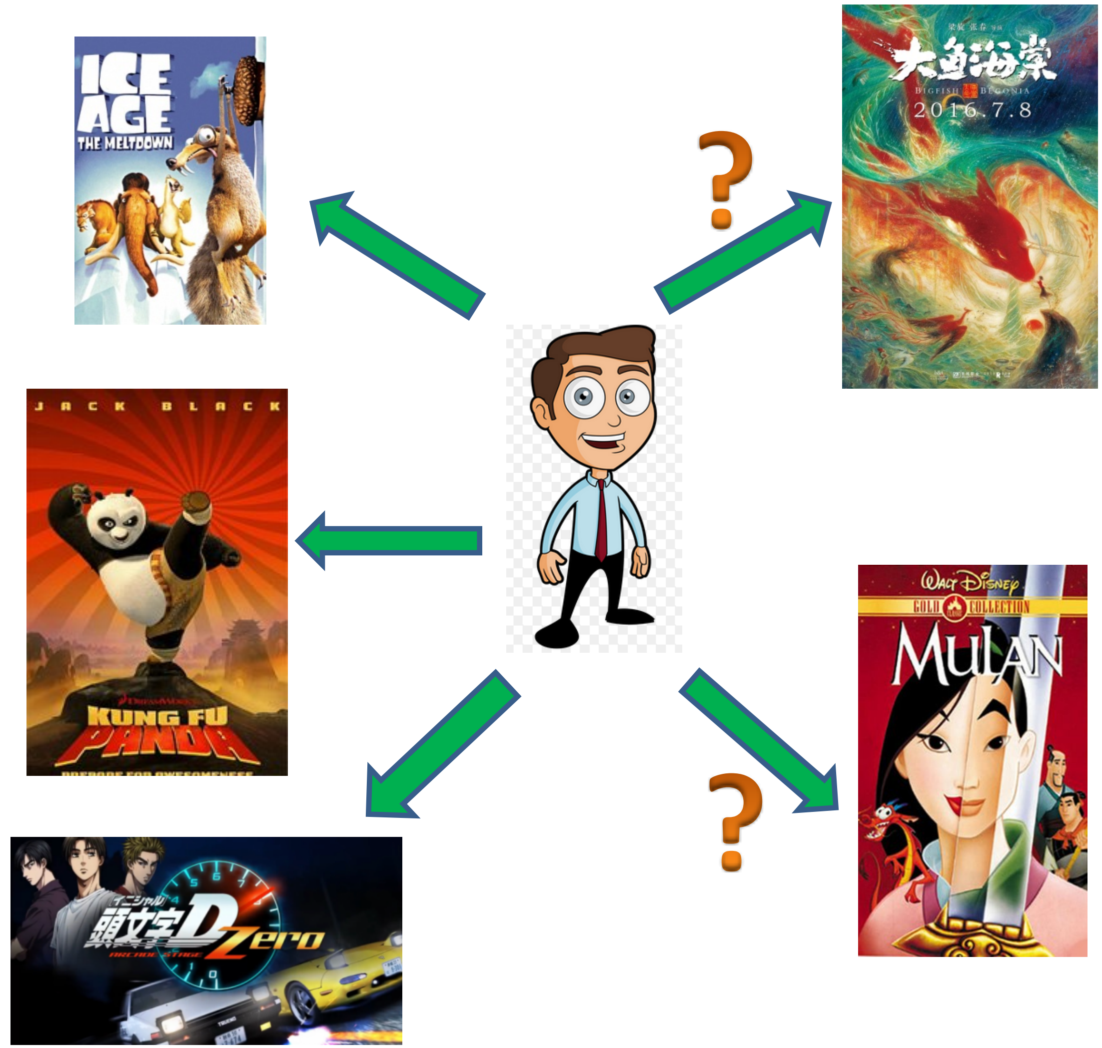


Image courtesy of Netflix

Recommender System

Image courtesy of Amazon

Recommender Systems: Software tools and techniques providing suggestions for items to be of use to a user.



Input Data:

1. A set of users $U=\{u_1,u_2,\ldots, u_m\}$
2. A set of items $V=\{v_1,v_2,\ldots,v_n\}$
3. The history preference ratings R_{ij}

Output Data:

Given user u and item v

Predict the rating or preference r_{uv}

Ricci et al. Introduction to Recommender Systems Handbook. 2011

Collaborative Filtering

		Airplane	Matrix	Room with a View	…	Shrek
		Comedy	Action	Romance	…	Cartoon
Joe	27, M	5	4	1		2
Carol	53, F	2		4		4
Tim	40, M					
Kumar	25, M	5	3			
Nancy	33, F	1		4		?
Stella	20, F					

Explicit or implicit feedbacks

Collaborative Filtering

		Airplane	Matrix	Room with a View	…	Shrek
		Comedy	Action	Romance	…	Cartoon
Joe	27, M	5	4	1		2
Carol	53, F	2		4		4
Tim	40, M					
Kumar	25, M	5	3			
Nancy	33, F	1		4		?
Stella	20, F					

Favorites of users like you

Matrix Factorization

m users and **n** items, each has **p** features.
user data U is a matrix with size m*p
Item data V is a matrix with size n*p

For user $u_i \in U$ and item $v_j \in V$
Predicting rating $r_{ij} = u_i * v_j^T$
Error= $u_i * v_j^T - R_{ij}$

$$\min \sum_i^m \sum_j^n I_{ij} [(u_i v_j - R_{ij})^2 + \lambda (||u_i||^2 + ||v_j||^2)]$$

I_{ij} = **1** if u_i has rating on v_j, otherwise **0**

Limitations of Collaborative Filtering Method:
Cold-start Problem: the user-item rating matrix could be extremely large and sparse. For new users and new items, this could be even worse.

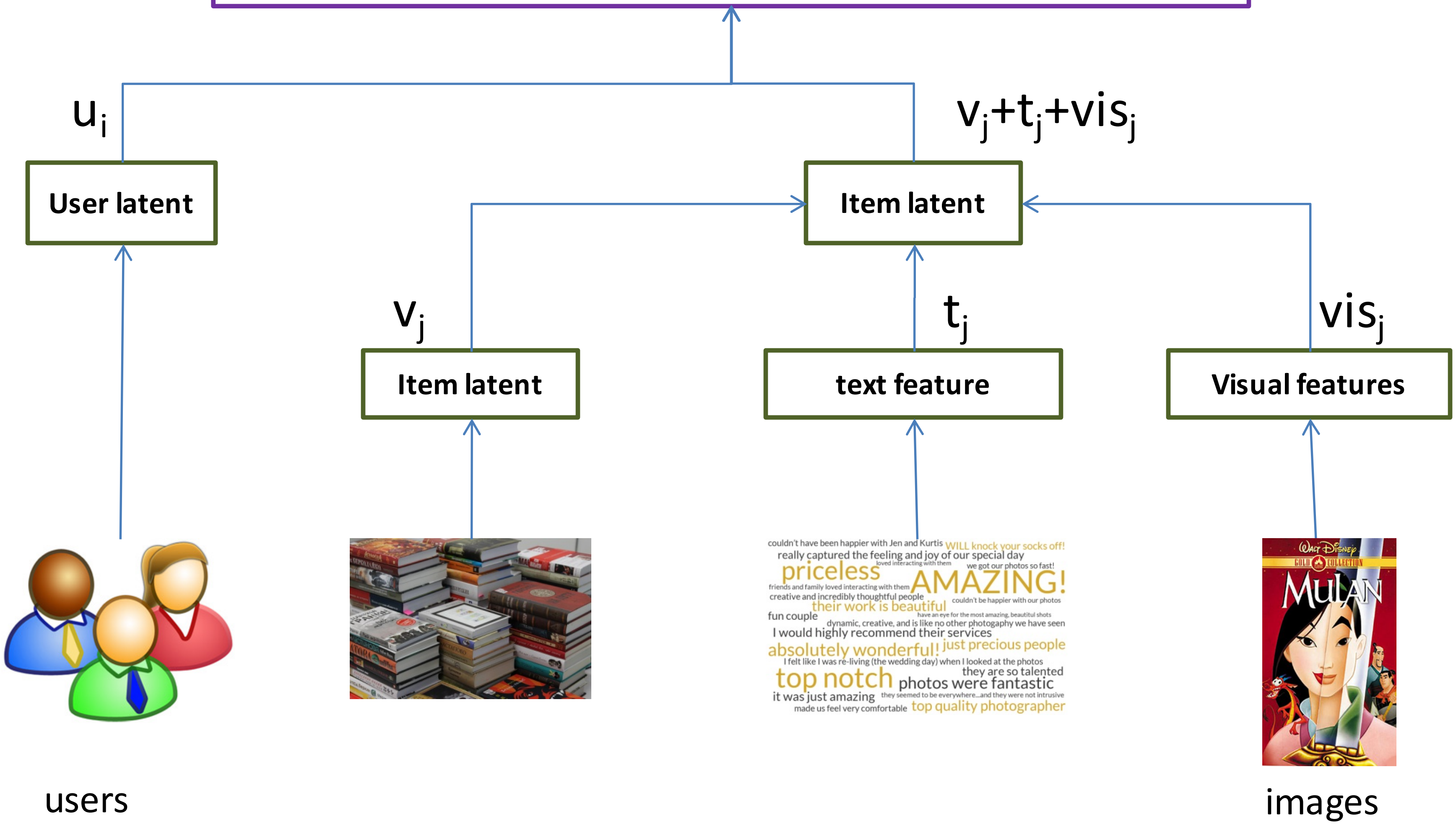
Netflix: rating number is 1.1% of all possible ratings

Solutions:
Adding extra features to item: review, image, etc.

Zhou et al. Large-scale Parallel Collaborative Filtering for the Netflix Prize.

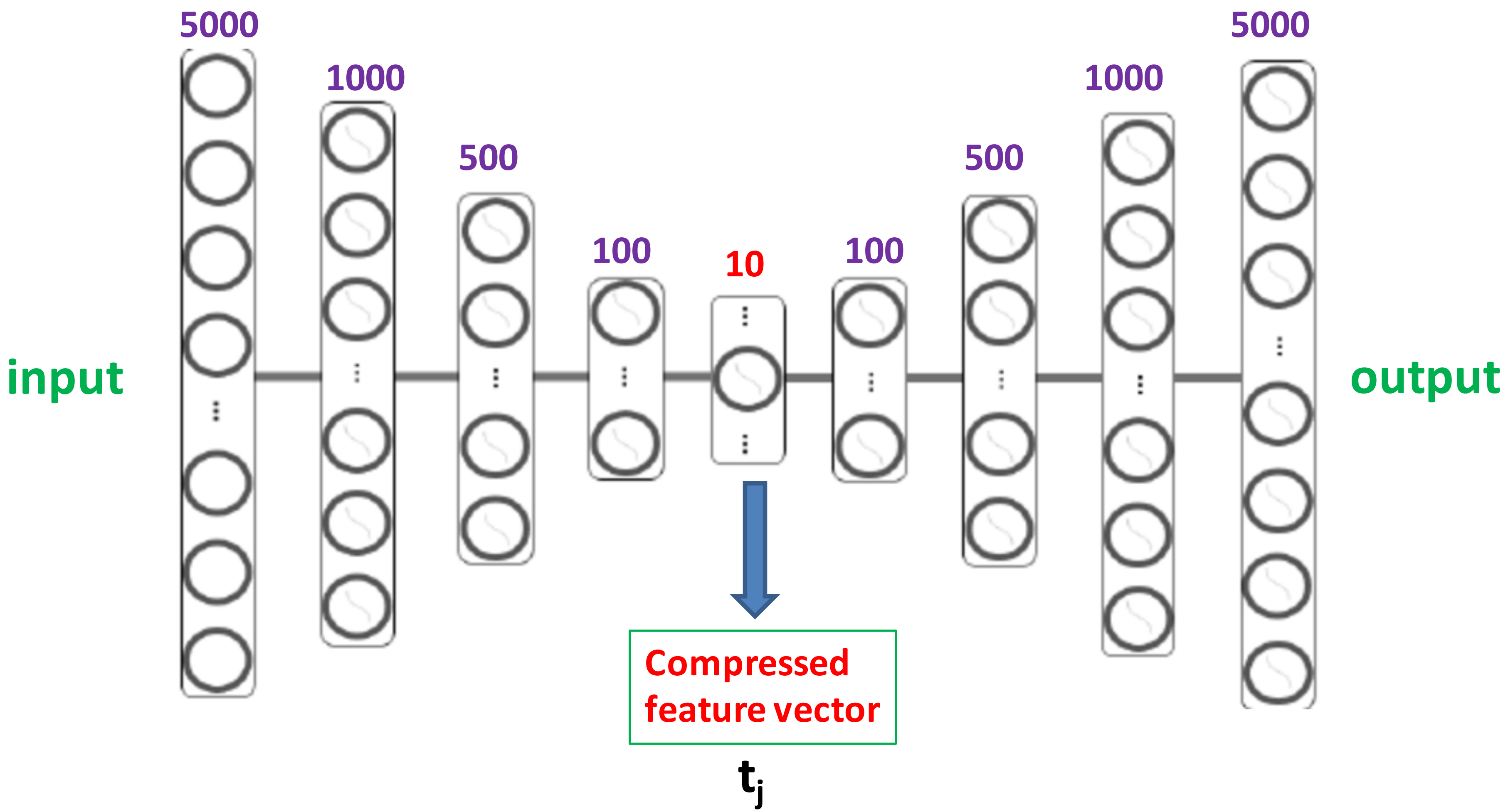
Collaborative Filtering with Deep Learning

$$\min \sum_i^m \sum_j^n I_{ij} \left[(u_i(v_j+t_j+vis_j)-R_{ij})^2 + \lambda \left(||u_i||^2 + ||v_j||^2 \right) \right]$$



Text Feature Learning

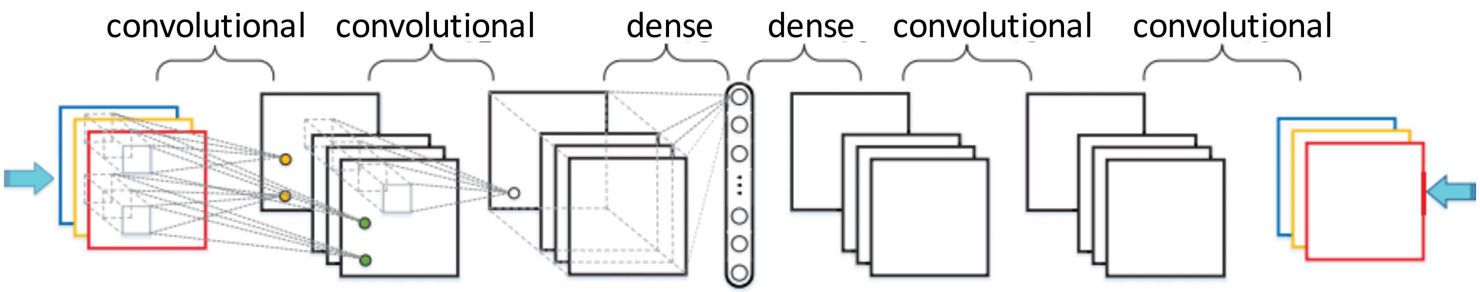
Stack Fully-Connected Autoencoder



Visual Feature Learning

Stack Convolutional Autoencoder

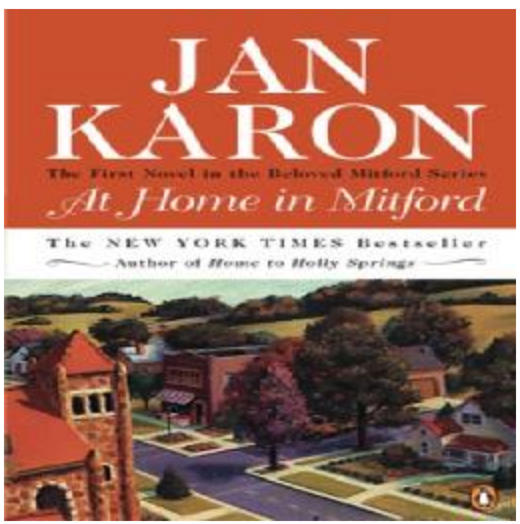
Input image

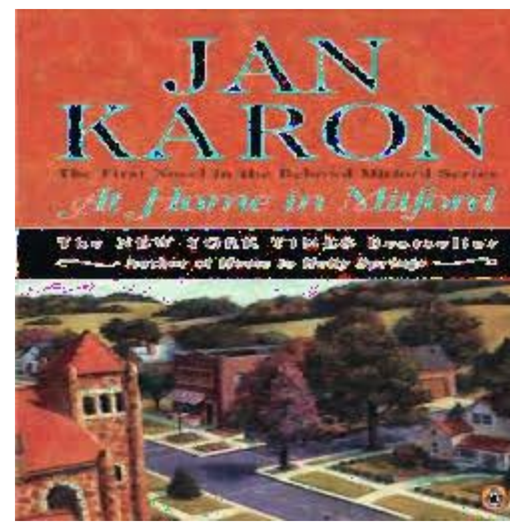


Output image

$\mathbf{vis_j}$

compressed feature vector

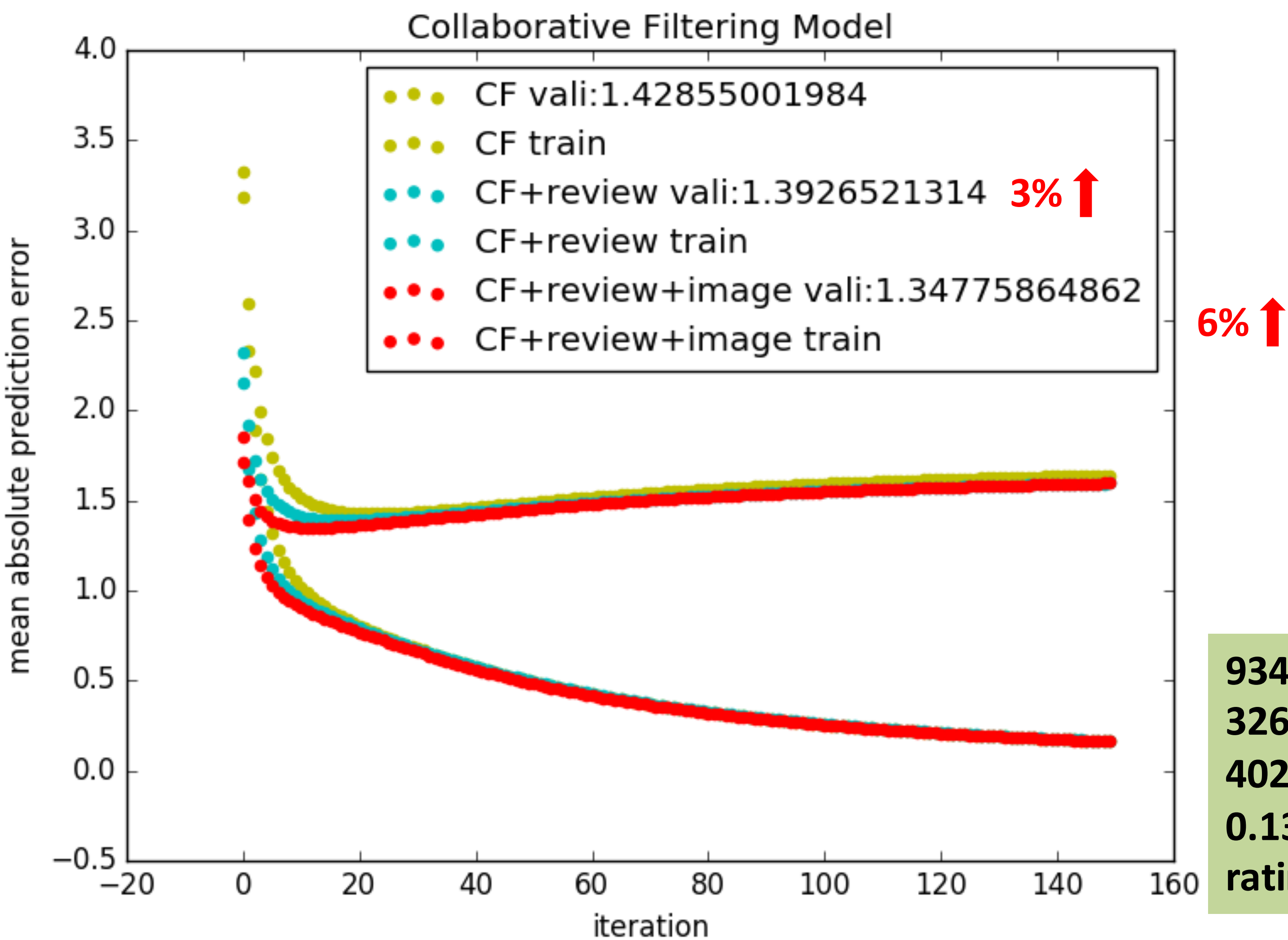




LSU
Louisiana State University

Results

Rating: 1 to 10

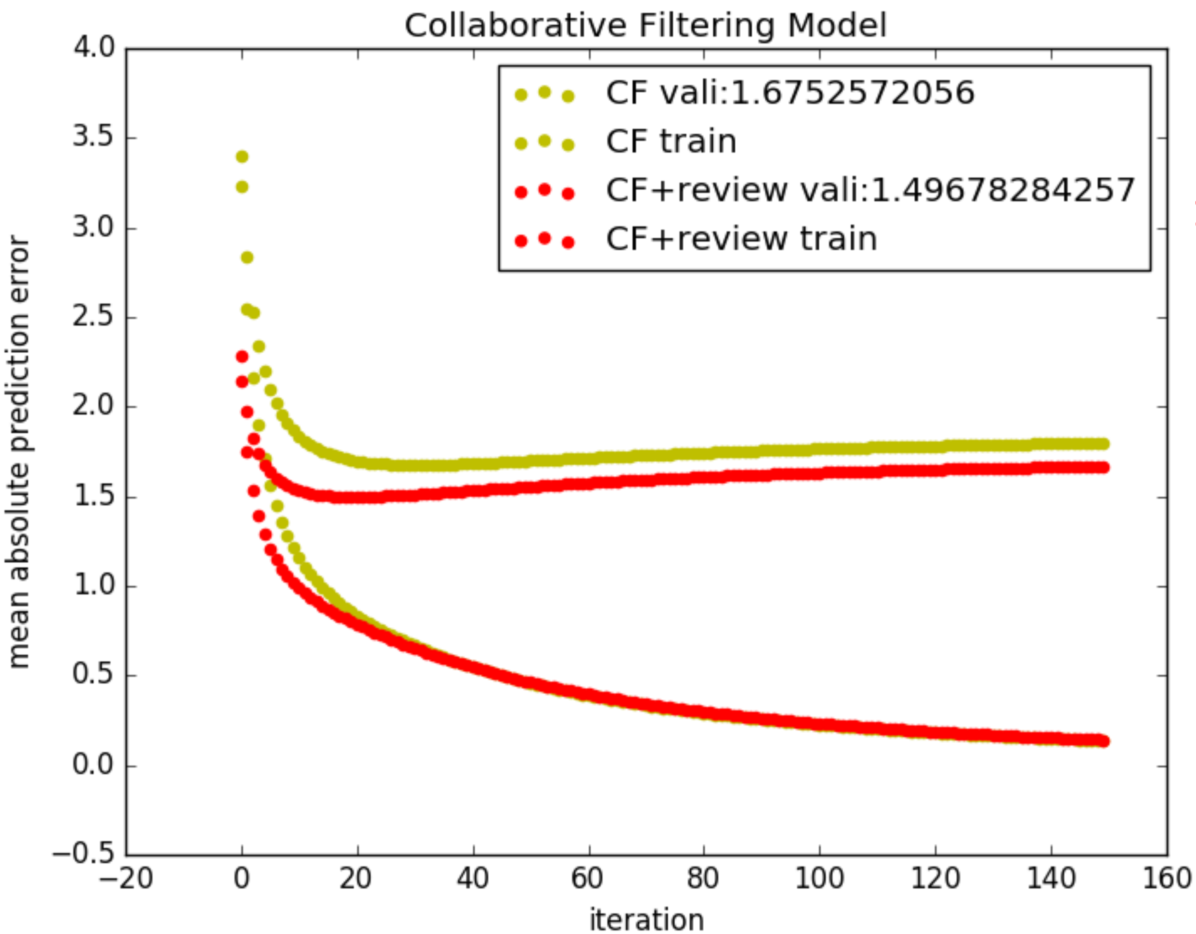


9342 users
3268 items
40247 ratings
0.13% of all possible ratings

Results

Louisiana State University

Rating: 1 to 10



10.7% ⬆

22072 users
14613 items
82811 ratings
0.03% of all possible ratings

Summary:

1. A hybrid recommender system which integrated collaborative filtering and deep learning has been implemented.

2. With extra features from texts and images, this system outperforms the traditional collaborative filtering method, especially when the rating matrix is extremely sparse.

Deep Learning Practice on LONI QB2

Feng Chen
HPC User Services
LSU HPC & LONI
sys-help@loni.org

Louisiana State University
Baton Rouge
November 9, 2016

Outline

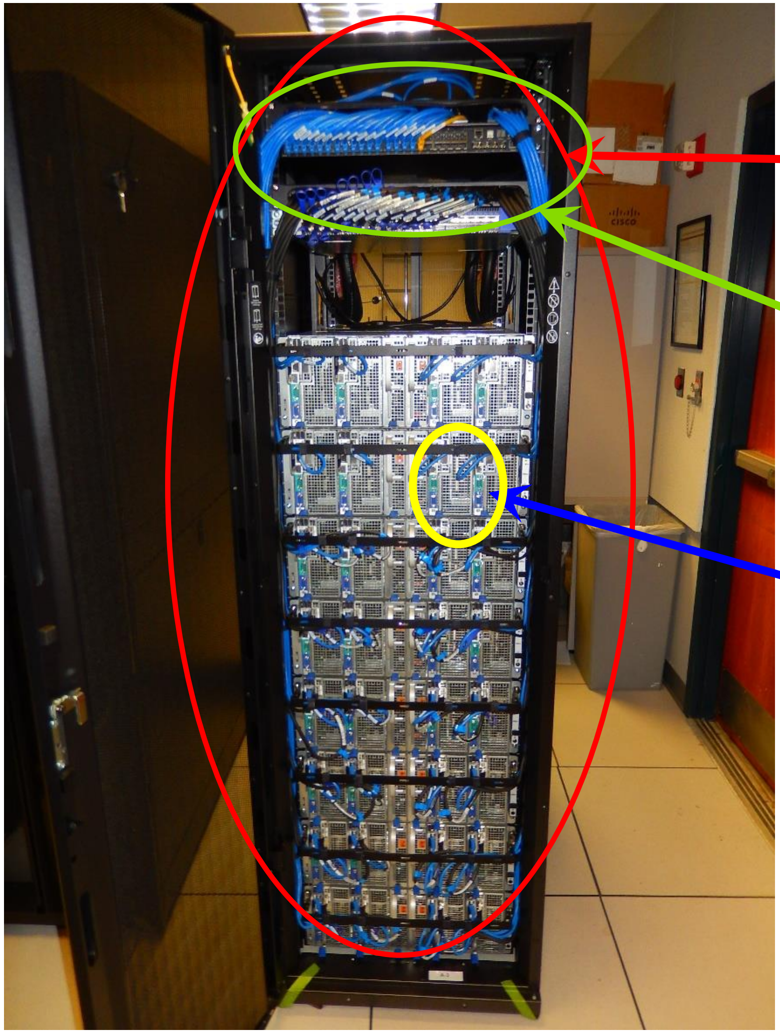
- **Overview of LONI QB2**
  - QB2 node specs
  - Cluster architecture
  - A few words about GPU
- **Access QB2 cluster**
  - Connect to QB2 clusters using ssh
  - Load python modules with Theano, Tensorflow and Keras installed
  - GPU Queues on QB2
- **Submitting jobs to QB2**
  - PBS script examples
    - Theano backend
    - Tensorflow backend
  - How to monitor your jobs

Overview of LONI QB2

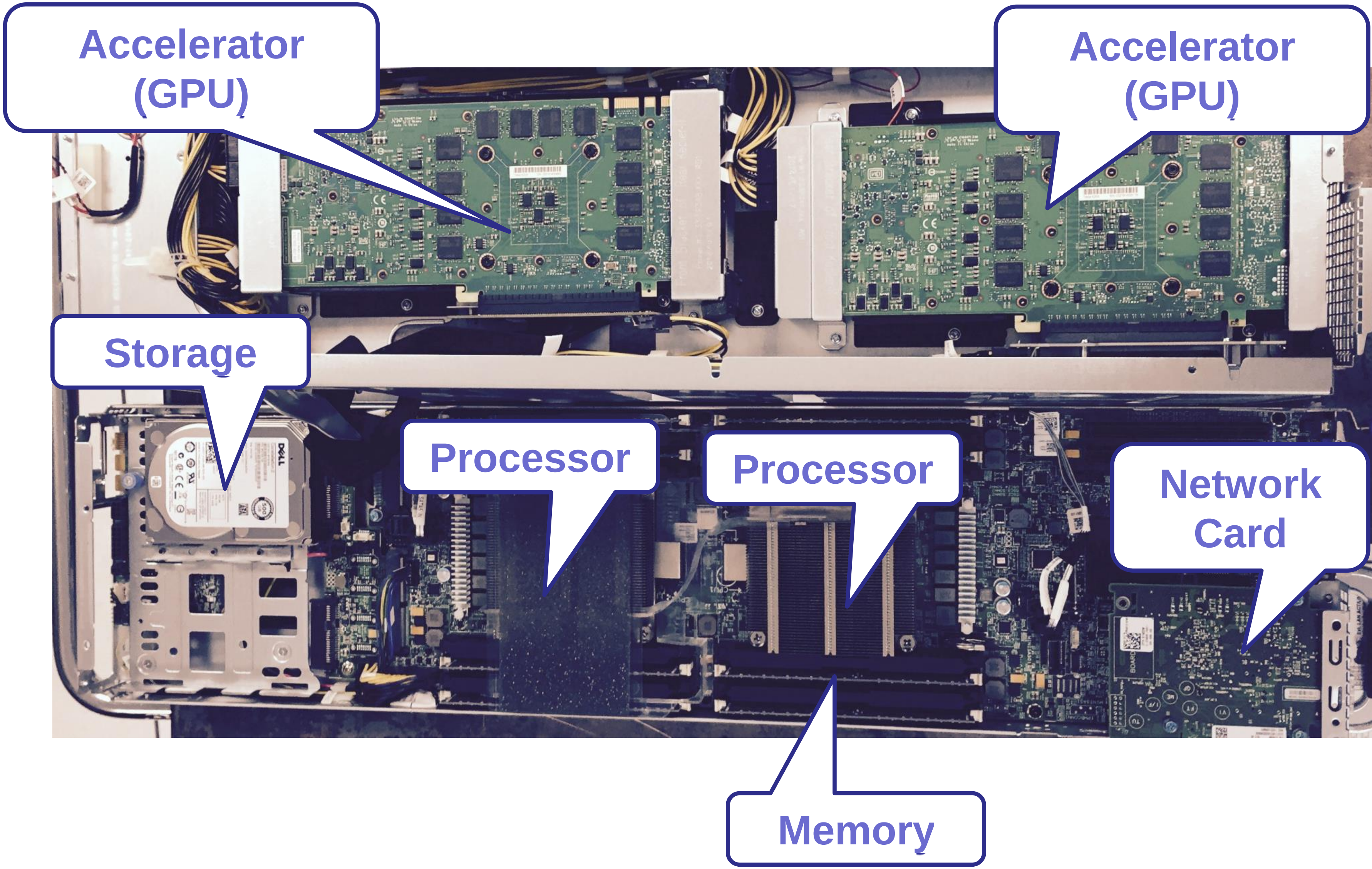
QB2 Hardware Specs

- **QB2 came on-line 5 Nov 2014.**
  - It is a 1.5 Petaflop peak performance cluster containing 504 compute nodes with
    - 960 NVIDIA Tesla K20x GPU's, and
    - Over 10,000 Intel Xeon processing cores. It achieved 1.052 PF during testing.
- **Ranked 46th on the November 2014 Top500 list.**
- **480 Compute Nodes, each with:**
  - Two 10-core 2.8 GHz E5-2680v2 Xeon processors.
  - 64 GB memory
  - 500 GB HDD
  - ***2 NVIDIA Tesla K20x GPU's***

Inside A QB Cluster Rack

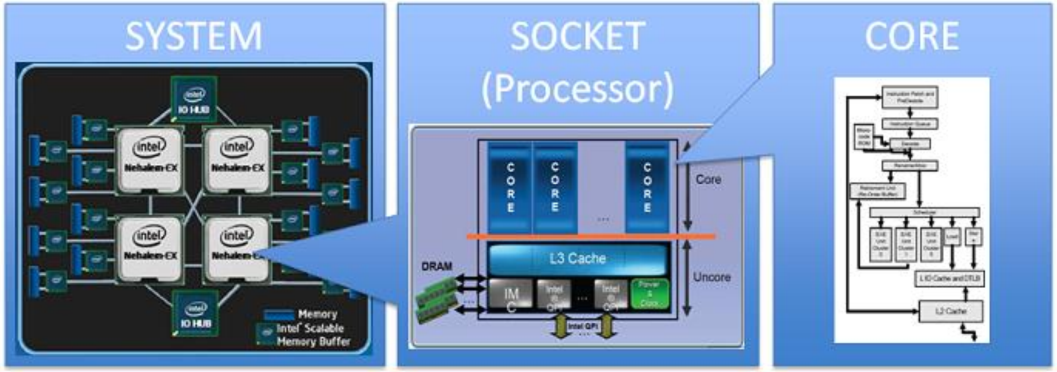


Inside A QB2 Dell C8000 Node



Cluster Nomenclature

Term	Definition
Cluster	The top-level organizational unit of an HPC cluster, comprising a set of nodes, a queue, and jobs.
Node	A single, named host machine in the cluster.
Core	The basic computation unit of the CPU. For example, a quad-core processor is considered 4 cores.
Job	A user's request to use a certain amount of resources for a certain amount of time on cluster for his work.
GPU	Graphics processing unit that works together with CPU to accelerate user applications



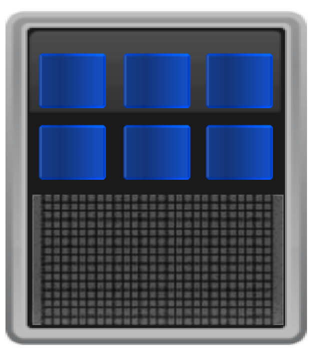
GPU Computing History

- **The first GPU (Graphics Processing Unit)s were designed as graphics accelerators, supporting only specific fixed-function pipelines.**
- **Starting in the late 1990s, the hardware became increasingly programmable, culminating in NVIDIA's first GPU in 1999.**
- **Researchers were tapping its excellent floating point performance. The General Purpose GPU (GPGPU) movement had dawned.**
- **NVIDIA unveiled CUDA in 2006, the world's first solution for general-computing on GPUs.**
- **CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model created by NVIDIA and implemented by the GPUs that they produce.**

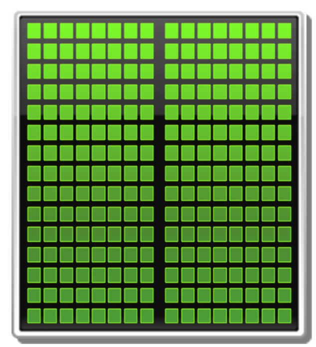
Add GPUs: Accelerate Science Applications

CPU

GPU



+



Why is GPU this different from a CPU?

- **Different goals produce different designs**
  - GPU assumes work load is highly parallel
  - CPU must be good at everything, parallel or not

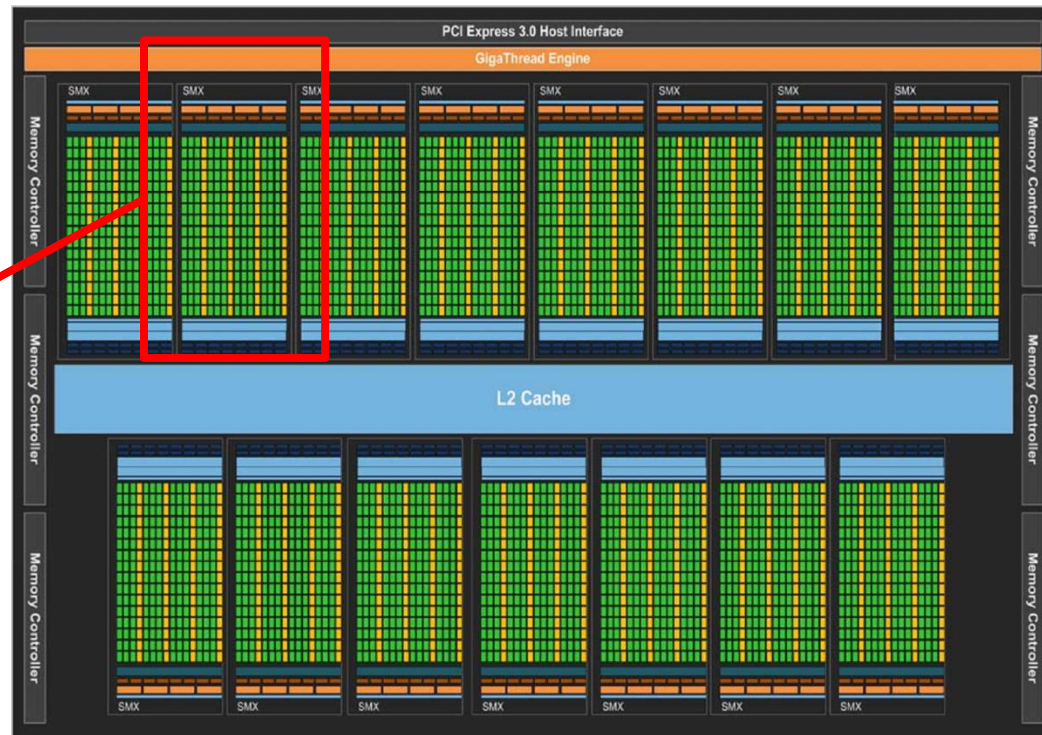
- **CPU: minimize latency experienced by 1 thread**
  - big on-chip caches
  - sophisticated control logic

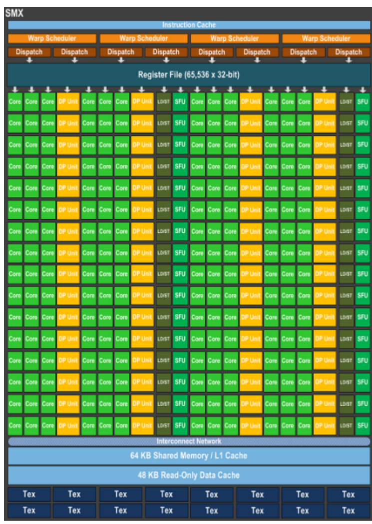
- **GPU: maximize throughput of all threads**
  - # threads in flight limited by resources => lots of resources (registers, bandwidth, etc.)
  - multithreading can hide latency => skip the big caches
  - share control logic across many threads

Overview of the GPU nodes

- **CPU: Two 2.6 GHz 8-Core Sandy Bridge Xeon 64-bit Processors (16)**
  - 64GB 1666MHz Ram
- **GPU: Two NVIDIA Tesla K20Xm**
  - 14 Streaming Multiprocessor (SMX)
  - 2688 SP Cores
  - 896 DP Cores
  - 6G global memory

K20Xm GPU Architecture

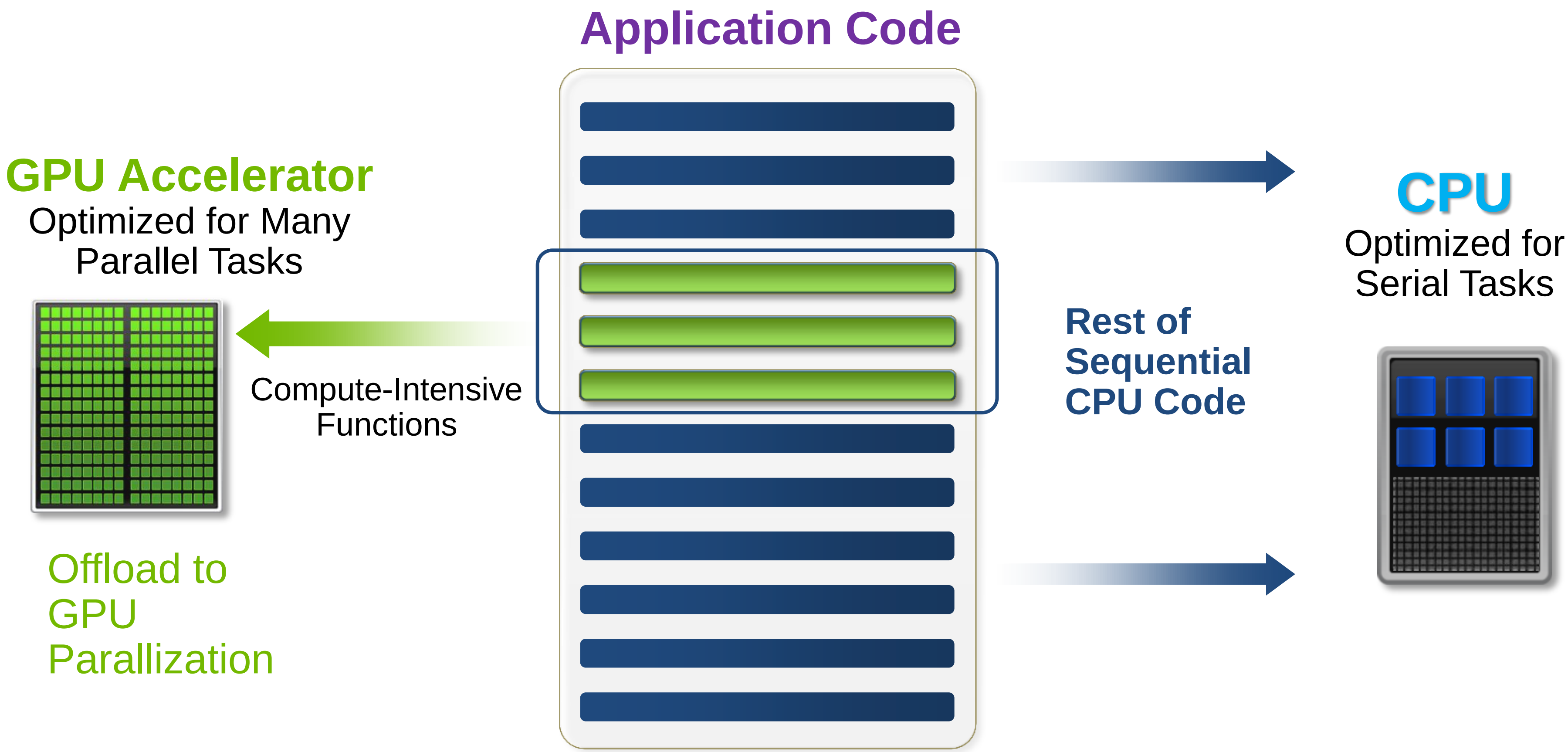




SMX (192 SP, 64 DP)

CUDA Execution Model

- Sequential code executes in a Host (CPU) thread
- Parallel code executes in many Device (GPU) threads across multiple processing elements



Heterogeneous Computing

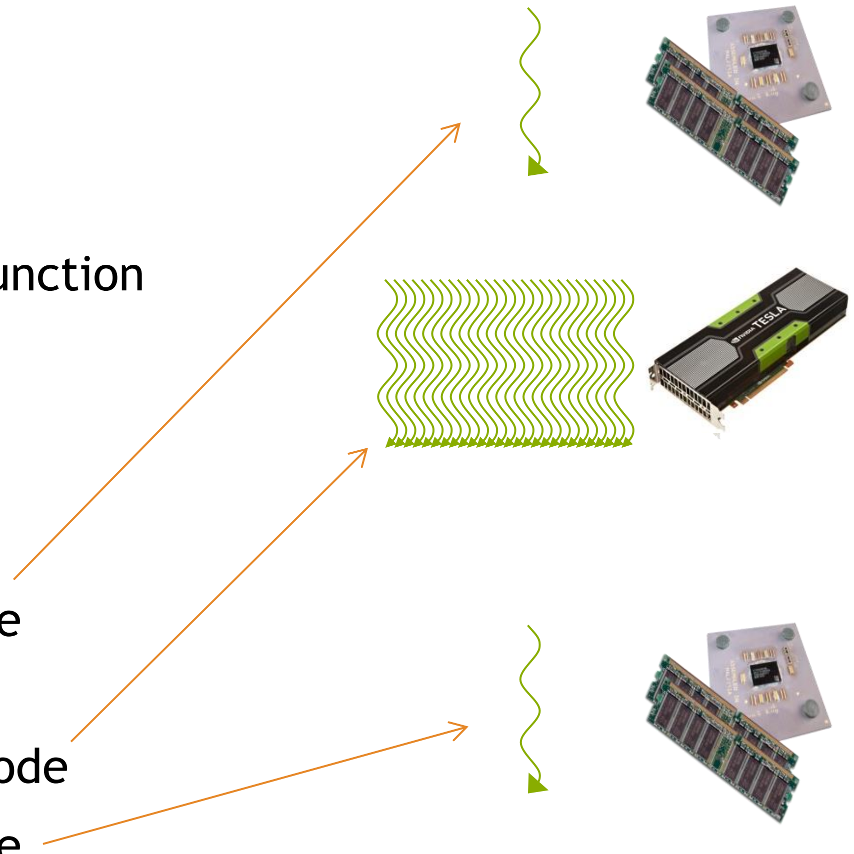
```
#include <iostream>
#include <algorithm>

using namespace std;

#define N        1024
#define RADIUS    3
#define BLOCK_SIZE 16

__global__ void stencil_1d(int *in, int *out) {
        __shared__ int temp[BLOCK_SIZE + 2 * RADIUS];
        int gindex = threadIdx.x + blockIdx.x * blockDim.x;
        int lindex = threadIdx.x + RADIUS;

        // Read input elements into shared memory
        temp[lindex] = in[gindex];
        if (threadIdx.x < RADIUS) {
                temp[lindex - RADIUS] = in[gindex - RADIUS];
                temp[lindex + BLOCK_SIZE] = in[gindex + BLOCK_SIZE];
        }

        // Synchronize (ensure all the data is available)
        __syncthreads();

        // Apply the stencil
        int result = 0;
        for (int offset = -RADIUS ; offset <= RADIUS ; offset++)
                result += temp[lindex + offset];

        // Store the result
        out[gindex] = result;
}

void fill_ints(int *x, int n) {
        fill_n(x, n, 1);
}

int main(void) {
        int *in, *out;           // host copies of a, b, c
        int *d_in, *d_out;       // device copies of a, b, c
        int size = (N + 2*RADIUS) * sizeof(int);

        // Alloc space for host copies and setup values
        in  = (int *)malloc(size); fill_ints(in,  N + 2*RADIUS);
        out = (int *)malloc(size); fill_ints(out, N + 2*RADIUS);

        // Alloc space for device copies
        cudaMalloc((void **)&d_in,  size);
        cudaMalloc((void **)&d_out, size);

        // Copy to device
        cudaMemcpy(d_in,  in,  size, cudaMemcpyHostToDevice);
        cudaMemcpy(d_out, out, size, cudaMemcpyHostToDevice);

        // Launch stencil_1d() kernel on GPU
        stencil_1d<<<N/BLOCK_SIZE,BLOCK_SIZE>>>(d_in + RADIUS,
d_out + RADIUS);

        // Copy result back to host
        cudaMemcpy(out, d_out, size, cudaMemcpyDeviceToHost);

        // Cleanup
        free(in); free(out);
        cudaFree(d_in); cudaFree(d_out);
        return 0;
}
```

parallel function

serial code

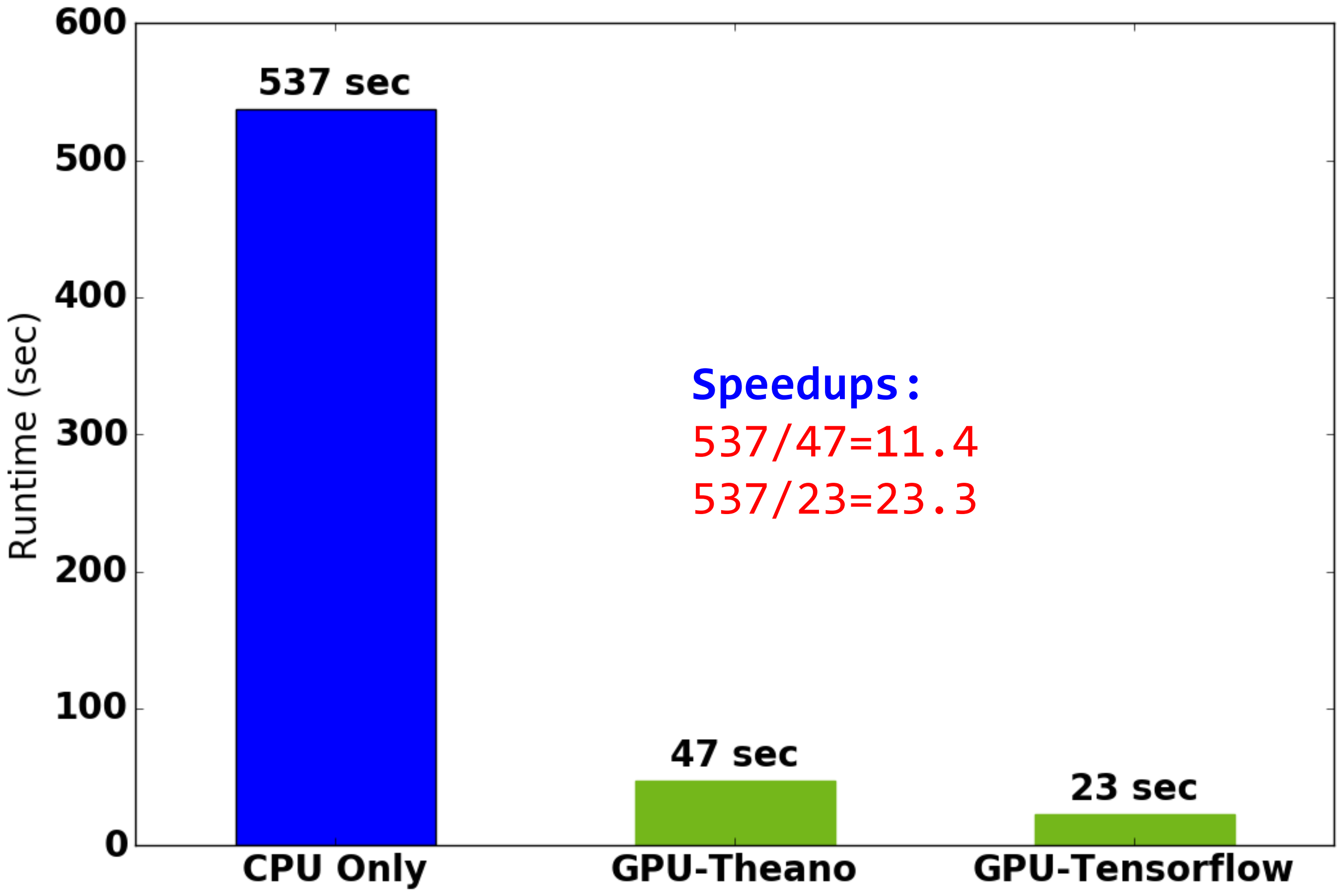
parallel code

serial code



Performance Comparison

- **Comparison of runtime for our deep learning example**
  - CIFAR10, 1 Epoch



Deep Learning Examples on LONI QB2

Access LONI QB2

LONI Cluster Architectures

- **Major architecture**
  - Intel x86_64 clusters
    - Vendor: Dell
    - Operating System: Linux (RHEL 4/5/6)
    - Processor: Intel

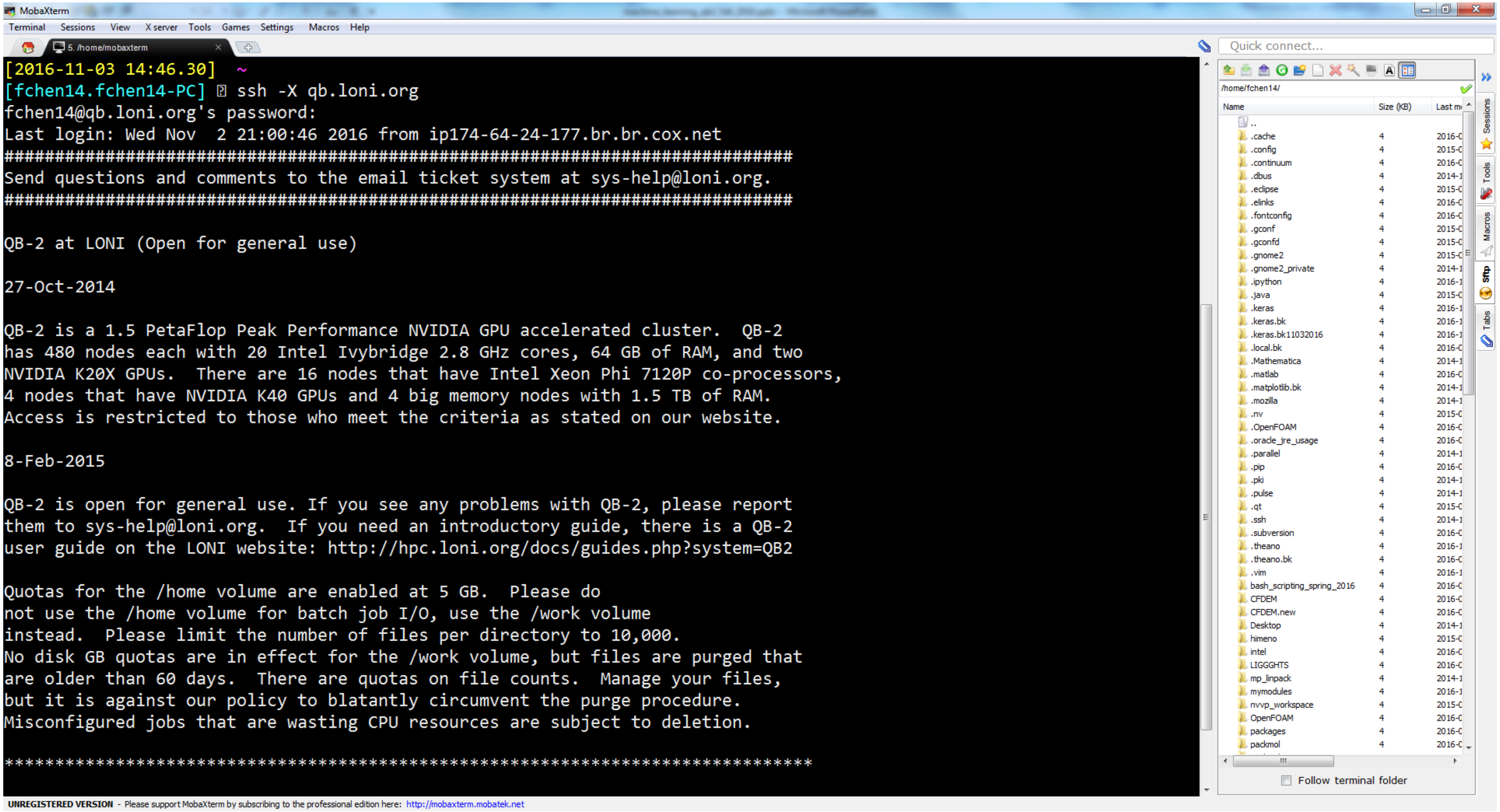
Accessing cluster using ssh (Secure Shell)

- **On Unix and Mac**
  - use ssh on a terminal to connect
- **Windows box (ssh client):**
  - Putty, Cygwin (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)
  - MobaXterm (http://mobaxterm.mobatek.net/)

- **Use this command to ssh to QB2:**

```
ssh username@qb.loni.org
```

Connect to QB2 using ssh (windows)



Connect to QB2 using ssh (Linux/Mac)

File Edit View Search Terminal Tabs Help

fchen14@feng-thinkm83: /home/fchen14/D... | fchen14@feng-thinkm83: /home/fchen14/D... | fchen14@feng-thinkm83: /home/fchen14/D... | fchen14@feng-thinkm83: /home/fchen14

```
[fchen14@feng-thinkm83 ~]$ssh -X qb.loni.org
fchen14@qb.loni.org's password:
Last login: Thu Nov  3 14:30:04 2016 from fchen14-2.lsu.edu
##############################################################################
Send questions and comments to the email ticket system at sys-help@loni.org.
##############################################################################

QB-2 at LONI (Open for general use)

27-Oct-2014

QB-2 is a 1.5 PetaFlop Peak Performance NVIDIA GPU accelerated cluster.  QB-2
has 480 nodes each with 20 Intel Ivybridge 2.8 GHz cores, 64 GB of RAM, and two
NVIDIA K20X GPUs.  There are 16 nodes that have Intel Xeon Phi 7120P co-processors,
4 nodes that have NVIDIA K40 GPUs and 4 big memory nodes with 1.5 TB of RAM.
Access is restricted to those who meet the criteria as stated on our website.

8-Feb-2015

QB-2 is open for general use. If you see any problems with QB-2, please report
them to sys-help@loni.org.  If you need an introductory guide, there is a QB-2
user guide on the LONI website: http://hpc.loni.org/docs/guides.php?system=QB2

Quotas for the /home volume are enabled at 5 GB.  Please do
not use the /home volume for batch job I/O, use the /work volume
instead.  Please limit the number of files per directory to 10,000.
No disk GB quotas are in effect for the /work volume, but files are purged that
are older than 60 days.  There are quotas on file counts.  Manage your files,
but it is against our policy to blatantly circumvent the purge procedure.
Misconfigured jobs that are wasting CPU resources are subject to deletion.

********************************************************************************
Allocations are required.  Using the -A qsub option to specify your allocation.
********************************************************************************
[fchen14@qb2 ~]$
```

Using Environment Modules on QB2

- **Environment Modules on QB2 is the framework to manage what software is loaded into a user's environment. Its functionality includes**
  - List all software packages currently available in the Environment Modules system,
  - List all software packages loaded into a user's environment,
  - Load/Unload/Switch software packages into a user's environment
  - Unload a software package from a user's environment.
- **Recall the following commands:**
  - `module avail {name}`
  - `module load <module_key>`
  - `module unload <module_key>`
  - `module disp <module_key>`
  - `module swap <module_key1> <module_key2>`

Use The Correct Python Module

- Use the following commands to load the correct python module to your environment:

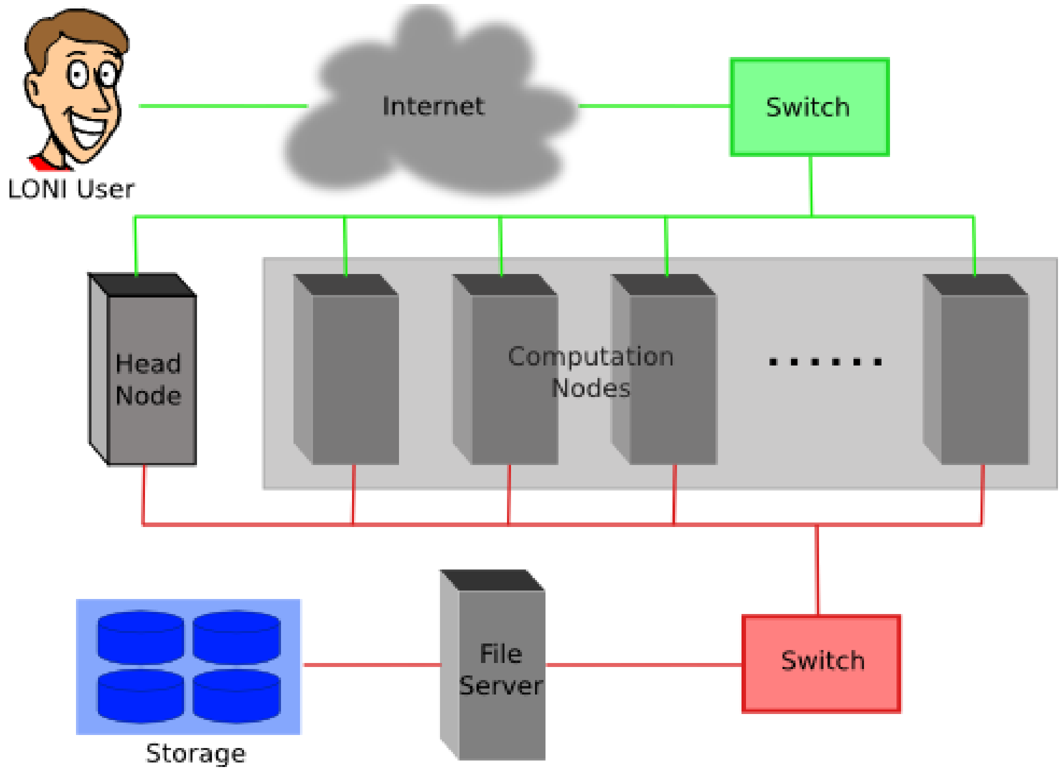
Must use this python key to import theano, tensorflow and keras!

```
[fchen14@qb001 ml_tut]$ module av python
----------- /usr/local/packages/Modules/modulefiles/apps ----------
python/2.7.10-anaconda python/2.7.12-anaconda python/2.7.7-anaconda
[fchen14@qb001 ml_tut]$ module load python/2.7.12-anaconda
[fchen14@qb001 ml_tut]$ which python
/usr/local/packages/python/2.7.12-anaconda/bin/python
[fchen14@qb001 ml_tut]$ python
Python 2.7.12 |Anaconda 4.1.1 (64-bit)| (default, Jul  2 2016, 17:42:40)
...
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> import keras, theano, tensorflow
Using Theano backend.
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcurand.so.7.5 locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcufft.so.7.5 locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcudnn.so.5.1 locally
I tensorflow/stream_executor/dso_loader.cc:111] successfully opened CUDA library libcublas.so.7.5 locally
>>>
```

Job Queues on QB2

Cluster Environment

- Multiple compute nodes
- Multiple users
- Each user may have multiple jobs running simultaneously
- Multiple users may share the same node



Job submission basics

- Find appropriate queue
- Understand the queuing system and your requirements and proceed to submit jobs
- Monitor jobs during execution

Job Queues

- **Nodes are organized into queues. Nodes can be shared.**
- **Each job queue differs in**
  - Number of available nodes
  - Max run time
  - Max running jobs per user
  - Nodes may have special characteristics: GPU/Xeon Phi’s, Large memory, etc.
- **Jobs need to specify resource requirements**
  - Nodes, time, queue

Available Queues on QB2

- **"qstat -q" to check available queues on QB2**

```
[fchen14@qb1 ~]$ qstat -q
server: qb3
Queue            Memory CPU Time Walltime Node  Run Que Lm  State
---------------- ------ -------- -------- ----  --- --- --  -----
single             --      --    168:00:0    1    2   0 --   E R
checkpt            --      --    72:00:00  256   68   0 --   E R
workq              --      --    72:00:00  128   61   0 --   E R
phi                --      --    72:00:00    4    0   0 --   E R
k40                --      --    72:00:00    4    3   0 --   E R
bigmem             --      --    72:00:00    1    4   2 --   E R
admin              --      --    24:00:00   --    0   0 --   E R
preempt            --      --    72:00:00   --    0   0 --   E R
priority           --      --    168:00:0  128    0   0 --   E R
                                               ----- -----
                                                 138     2
```

- **Each node on QB2:**
  - In workq queue has 2 k20xm NVidia GPUs
  - In k40 queue has 2 k40 NVidia GPUs
- **Use either workq or k40 queue to submit today's job script as we are using GPUs today**

Submit and Monitor Your Jobs

Two Job Types

- **Interactive job**
  - Set up an interactive environment on compute nodes for users
    - Advantage: can run programs interactively
    - Disadvantage: must be present when the job starts
  - Purpose: testing and debugging, compiling
    - **Do not run on the head node!!!**
    - Try not to run interactive jobs with large core count, which is a waste of resources)
- **Batch job**
  - Executed without user intervention using a job script
    - Advantage: the system takes care of everything
    - Disadvantage: can only execute one sequence of commands which cannot changed after submission
  - Purpose: production run

PBS Script (CIFAR10) Tensorflow Backend

```
#!/bin/bash
#PBS -l nodes=1:ppn=20
#PBS -l walltime=72:00:00
#PBS -q workq
#PBS -N cnn.tf.gpu
#PBS -o cnn.tf.gpu.out
#PBS -e cnn.tf.gpu.err
#PBS -A loni_loniadmin1

cd $PBS_O_WORKDIR
# use the tensorflow backend
export KERAS_BACKEND=tensorflow
# use this python module key to access tensorflow, theano and keras
module load python/2.7.12-anaconda
python cifar10_cnn.py
```

Tells the job scheduler how much resource you need.

How will you use the resources?

PBS Script (CIFAR10) Theano Backend

```
#!/bin/bash
#PBS -l nodes=1:ppn=20
#PBS -l walltime=72:00:00
#PBS -q workq
#PBS -N cnn.th.gpu
#PBS -o cnn.th.gpu.out
#PBS -e cnn.th.gpu.err
#PBS -A loni_loniadmin1
```

Tells the job scheduler how much resource you need.

How will you use the resources?

```
cd $PBS_O_WORKDIR
# use this python module key to access tensorflow, theano and keras
module load python/2.7.12-anaconda
# use the theano backend
export KERAS_BACKEND=theano
export THEANO_FLAGS="mode=FAST_RUN,device=gpu,floatX=float32,lib.cnmem=1"
python cifar10_cnn.py
```

Steps to Submit Jobs

```
[fchen14@qb1 ml_tut]$ cd /project/fchen14/machine_learning/ml_tut
[fchen14@qb1 ml_tut]$ qsub sbm_cifar10_cnn_tensorflow.pbs
305669.qb3
[fchen14@qb1 ml_tut]$ qstat -u fchen14

qb3:
                                                                        Req'd  Req'd   Elap
Job ID               Username    Queue    Jobname          SessID NDS   TSK    Memory Time  S Time
-------------------- ----------- -------- ---------------- ------ ----- ------ ------ ----- - -----
305667.qb3           fchen14     workq    cnn.tf.gpu        25633     1     20     --  72:00 R   --
305669.qb3           fchen14     k40      cnn.tf.gpu          --      1     20     --  72:00 R   --
[fchen14@qb1 ml_tut]$ qshow 305669.qb3
PBS job: 305669.qb3, nodes: 1
Hostname  Days Load CPU U# (User:Process:VirtualMemory:Memory:Hours)
qb002       24 0.32 205  4 fchen14:python:166G:1.6G:0.1 fchen14:305669:103M:1M
PBS_job=305669.qb3 user=fchen14 allocation=loni_loniadmin1 queue=k40 total_load=0.32 cpu_hours=0.11
wall_hours=0.05 unused_nodes=0 total_nodes=1 ppn=20 avg_load=0.32 avg_cpu=205% avg_mem=1647mb
avg_vmem=170438mb top_proc=fchen14:python:qb002:166G:1.6G:0.1hr:205%
toppm=msun:python:qb002:169456M:1190M node_processes=4
```

Job Monitoring - Linux Clusters

- **Check details on your job using `qstat`**

```
$ qstat -n -u $USER  : For quick look at nodes assigned to you
$ qstat -f jobid     : For details on your job
$ qdel jobid         : To delete job
```

- **Check approximate start time using `showstart`**

```
$ showstart jobid
```

- **Check details of your job using `checkjob`**

```
$ checkjob jobid
```

- **Check health of your job using `qshow`**

```
$ qshow jobid
```

- **Dynamically monitor node status using `top`**
  - See next slides
- **Monitor GPU usage using `nvidia-smi`**
  - See next slides

❖ **Please pay close attention to the load and the memory consumed by your job!**

Using the "top" command

- **The top program provides a dynamic real-time view of a running system.**

```
[fchen14@qb1 ml_tut]$ ssh qb002
Last login: Mon Oct 17 22:50:16 2016 from qb1.loni.org
[fchen14@qb002 ~]$ top
top - 15:57:04 up 24 days,  5:38,  1 user,  load average: 0.44, 0.48, 0.57
Tasks: 606 total,   1 running, 605 sleeping,   0 stopped,   0 zombie
Cpu(s):  9.0%us,  0.8%sy,  0.0%ni, 90.2%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  132064556k total,  9759836k used, 122304720k free,   177272k buffers
Swap: 134217720k total,        0k used, 134217720k free,  5023172k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
21270 fchen14   20   0  166g 1.6g 237m S 203.6  1.3  16:42.05 python
22143 fchen14   20   0 26328 1764 1020 R  0.7  0.0   0:00.76 top
   83 root      20   0     0    0    0 S  0.3  0.0  16:47.34 events/0
   97 root      20   0     0    0    0 S  0.3  0.0   0:25.80 events/14
  294 root      39  19     0    0    0 S  0.3  0.0  59:45.52 kipmi0
    1 root      20   0 21432 1572 1256 S  0.0  0.0   0:01.50 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.02 kthreadd
```

Monitor GPU Usage

- **Use nvidia-smi to monitor GPU usage:**

```
[fchen14@qb002 ~]$ nvidia-smi -l
Thu Nov  3 15:58:52 2016
+------------------------------------------------------+
| NVIDIA-SMI 352.93     Driver Version: 352.93         |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K40m          On   | 0000:03:00.0     Off |                    0 |
| N/A   34C    P0   104W / 235W |  11011MiB / 11519MiB |     77%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K40m          On   | 0000:83:00.0     Off |                    0 |
| N/A   32C    P0    61W / 235W |  10950MiB / 11519MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0     21270    C   python                                     10954MiB |
|    1     21270    C   python                                     10893MiB |
+-----------------------------------------------------------------------------+
```

Future Trainings

- **This is the last training for this semester**
  - Keep an eye on future HPC trainings at:
    - http://www.hpc.lsu.edu/training/tutorials.php#upcoming

- **Programming/Parallel Programming workshops**
  - Usually in summer

- **Visit our webpage: www.hpc.lsu.edu**