Intro to Tensors & Pytorch

SCC setup instructions at the end

Outline

- What is a “Tensor”?
- Tensor Operations in Pytorch
- SCC + Environment setup

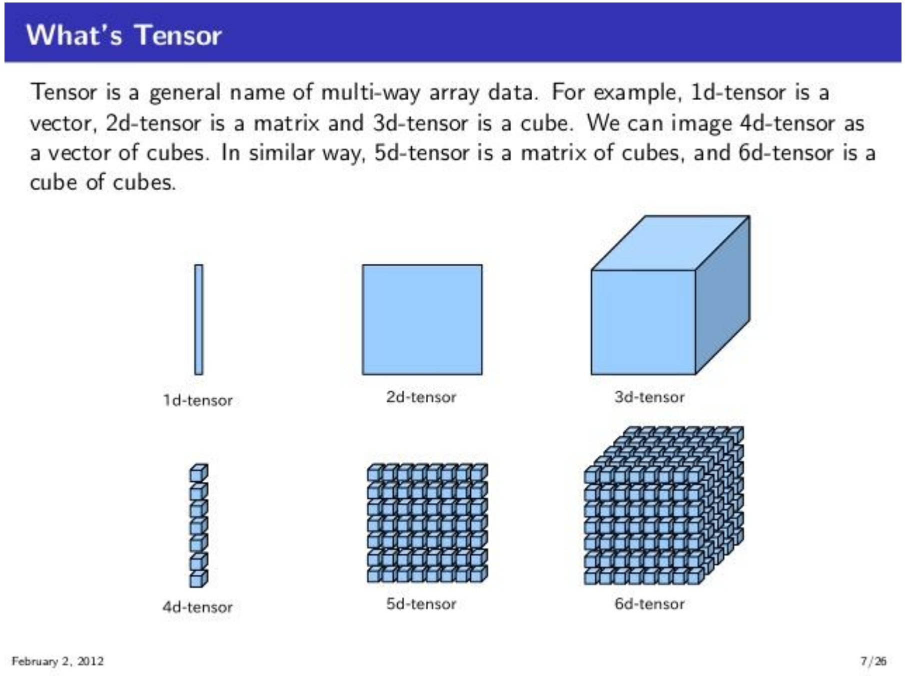
The bare basics…

A number is zero dimensional, called a scalar. Put a bunch together in an ordered list and you get a vector. Put numbers in a 2D grid and you get a matrix.

– A tensor is a generalization of these things to any number of dimensions.

What is a Tensor?

- A generic structure that can be used for storing, representing, and changing data.
- Tensors can be thought of as multidimensional generalizations of matrices.
- However, people from the math and physics community might have a different definitions of a tensor from those in the ML community.
  - Tensors in Physics and Math have a much more complex formulation. (More on the next slide)
  - Tensors in ML, are just multi-dimensional arrays with a fancy name. ("MultidimensionalArrayFlow" would not have made a great name)
- Each small cube → represents an entry



What's Tensor

Tensor is a general name of multi-way array data. For example, 1d-tensor is a vector, 2d-tensor is a matrix and 3d-tensor is a cube. We can image 4d-tensor as a vector of cubes. In similar way, 5d-tensor is a matrix of cubes, and 6d-tensor is a cube of cubes.

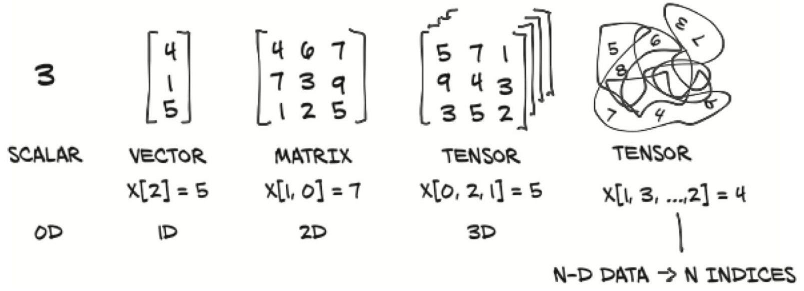
1d-tensor
2d-tensor
3d-tensor
4d-tensor
5d-tensor
6d-tensor

February 2, 2012 7/26

source

More reading

- Tensor from the Math world:
    - abstract objects that generalize scalars, vectors, and matrices to higher dimensions
    - defined based on their rank (number of indices) and follow strict transformation rules when moving between different coordinate systems
    - The focus in math is on the structure and transformation rules of these objects
    - https://mathworld.wolfram.com/Tensor.html
- Tensor from the Physics world:
    - A tensor in physics often represents a measurable quantity that depends on the system's position in space and time. Ex: the stress tensor describes how forces are distributed inside a material.
    - https://www.physlink.com/education/askexperts/ae168.cfm
- Tensor from the ML world: No fancy jargon – Just a container to store your numerical values



igure 3.2 Tensors are the building blocks for representing data in PyTorch.

Image source: Stevens et al.'s "Deep Learning with PyTorch"

Why do we need Tensors?

- Tensors along with improved hardware systems – allow for rapid processing of huge amounts of data.

- Automatic Differentiation - Modern deep learning frameworks leverage tensors for their automatic differentiation capabilities. When building and training neural networks, the gradients of the loss with respect to the parameters are required for optimization (e.g., using gradient descent). Tensors streamline the computation of these gradients.

- Why not lists or numpy arrays ? - Because of the level of abstraction - While lists and NumPy arrays may be capable of handling large amounts of data, they lack the level of abstraction required for efficient and flexible computation in deep learning systems. The tensor framework (PyTorch) hides the complexity of computing gradients, and other tasks. Tensors allow for:
  - Efficient GPU and TPU Support
  - Automatic Differentiation - efficient gradient computations, which are essential in backpropagation during neural network training
  - Dynamic Computation Graphs (PyTorch)
  - Optimized for Multi-Dimensional Data

Main attributes of tensors:

- Rank: the number of axes of a tensor
- Shape: an n-tuple (n is the rank), each value in the tuple is the size of that particular dimension.
- Data type: the type of value of the tensor (int, float, etc)

```
two_dim_tensor = [
[1, 2, 3],
[4, 5, 6]
]
```

- To find the entry 3, we need to index first into the 0-th list, then into the 2-nd element → two_dim_tensor[0][2] → Since we need 2 indices, the rank is 2.
- The shape of the above tensor is (2, 3)

Tensor representations

- In NLP:

sentence
Hi John
Hi James
Hi Brian

Word dictionary

Unique word	index	One hot encoding
Hi	0	[1, 0, 0, 0]
John	1	[0, 1, 0, 0]
James	2	[0, 0, 1, 0]
Brian	3	[0, 0, 0, 1]

Tensor representations

Sentence	Vector Representation
Hi John	[[1, 0, 0, 0], [0, 1, 0, 0]]
Hi James	[[1, 0, 0, 0], [0, 0, 1, 0]]
Hi Brian	[[1, 0, 0, 0], [0, 0, 0, 1]]

Mini-batch input will be:

```
[
[[1, 0, 0, 0], [0, 1, 0, 0]],         # Hi John
[[1, 0, 0, 0], [0, 0, 1, 0]],         # Hi James
[[1, 0, 0, 0], [0, 0, 0, 1]]          # Hi Brian
]
```

Shape → (3, 2, 4) → 3D tensor
3 → Number of examples/sentences
2 → Each sentence has 2 words
4 → Each world is represented by 4 indices

Tensor representations

Images:
(5, 3, 28, 28) → 5 images, 3 color channels (R, G, B), 28 rows (H), 28 columns (W)

Videos:
(5, 3, 24, 28, 28) →5 videos, 3 color channels (R, G, B), 24 Frames, 28 rows, 28 columns,

Pytorch

PyTorch mostly follows a numpy like naming convention. Most things are available as methods (t.tanh()) and functions (torch.tanh(t)).

More Here: Please do try out the Pytorch Notebook at: https://github.com/DL4DS/fa2024/blob/main/static_files/discussion_slides/00_fundamentals.ipynb - which introduces all the major torch operations you'd likely come across.

SCC Setup

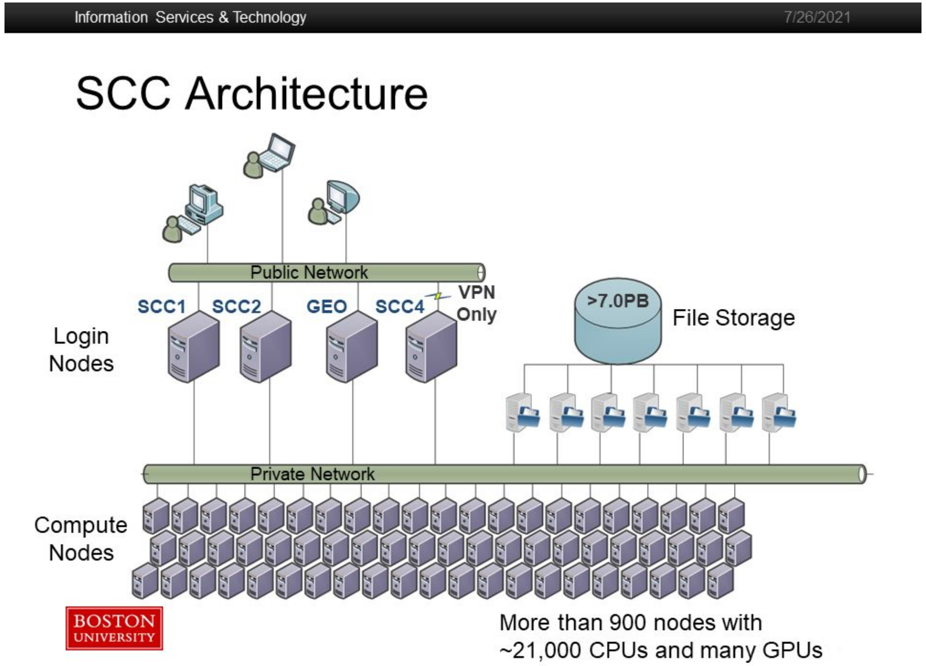
What is an SCC?

- SCC - Shared Computing Cluster - A Linux cluster that composes of both Shared and Buy-in components.

- The system currently includes over 12,000 shared CPU cores, over 16,000 Buy-in CPU cores, 300 GPU cores, and 12 petabytes of storage for research data (approximately 92% of this is Buy-in storage). (Source)

- Login Nodes - Mainly for Auth, file management, job submission and monitoring

- Compute Nodes - Running intensive computational job



Connecting to SCC

- Launch **scc-ondemand.bu.edu** (check if you have ‘/projectnb /ds542 /’ under files)
- Check if you can “`ssh {your_username}@scc1.bu.edu` ” in your terminal.
- Recommended - Use VSCode / Cursor - Follow these steps for remote development: https://www.bu.edu/tech/support/research/system-usage/scc-environment/editors-viewers-and-ides/vscode/

Things to Note

- Your **home directory** only has **10GB** of storage. Ideally, do not install anything there. Work in the /projectnb directory.
- Recommended steps to create a conda environment **in SCC**:
    - module load miniconda (if you get a "`WARNING: You do not have a .condarc file in your home directory` " message, run `setup_scc_condarc.sh` : https://www.bu.edu/tech/support/research/software-and-programming/common-languages/python/python-software/miniconda-modules/)
    - Check if "conda config --show pkgs_dirs" returns "/projectnb/…". Else do the below:
        - `conda config --add pkgs_dirs /projectnb/ds542/students/{Your_Folder_Name}/.conda`
        - `conda config --add envs_dirs /projectnb/ds542/students/{Your_Folder_Name}/.conda`
        - `conda config --show pkgs_dirs` - To confirm
    - By doing the "config –add" commands, the .conda path with "/projectnb/…" should be on top in the `~/.condarc` file. (You could just open this file and add the paths there, skipping the above step, use cat ~/.condarc to view the file)
    - `cd /projectnb/ds542/students/{Your_Folder_Name}`
    - Create an environment using, "`conda create -n dl4ds python=3.9`"
    - `conda activate dl4ds`
    - You can then install packages using conda install…, pip install…, etc. Ref: https://www.bu.edu/tech/support/research/software-and-programming/common-languages/python/python-installs/conda/
    - Try installing a package (example: pip install numpy), and try importing it:

      ```
      import numpy
      print(numpy.__version__)
      ```
    - Install torch and try running some of the tensor operations
- If you hit your home quota limit follow this: https://www.bu.edu/tech/support/research/system-usage/using-file-system/storage-quotas/

SCC Resources

You should be able to install the packages you need now, for submitting jobs, requesting interactive jobs, etc. Take a look at the SCC cheat sheet here:

https://dl4ds.github.io/fa2024/materials/

A few examples…

SCC Modules

- You can make use of the stored software modules on the SCC.
- To check if a certain package exists: “`module avail cuda`” – shows all version available
- To load a package: “`module load cuda`” or to load a specific version “`module load cuda/12.2`”

SCC - Requesting an Interactive Job

- Once you’ve logged in, and are on your terminal – you are now using a login node.
- To request for a compute node – “`qrsh -pe omp 4 -P ds598 -l gpus=1`” – to see all the args that can be passed and what they mean, see https://dl4ds.github.io/fa2024/materials/
- Once the request goes through, you’ll be taken to a compute node. Note that you’ll be taken to your home directory. So do a “`cd /projectnb/ds542/students/{your_username}`” or “File → Open Folder…” in VSCode and enter this path.
- You can create your directories under the ”students/{your_username}” directory – again make sure to create files, directories, etc under “/projectnb”
- Try “nvidia-smi” – to see the GPU assigned and to monitor GPU usage. – You may be assigned specific GPU ids, and notice that some of the GPU cores are already being used. There's no need for additional configuration—SCC automatically sets the CUDA_VISIBLE_DEVICES environment variable for you. You can proceed with your code as normal. – How? -

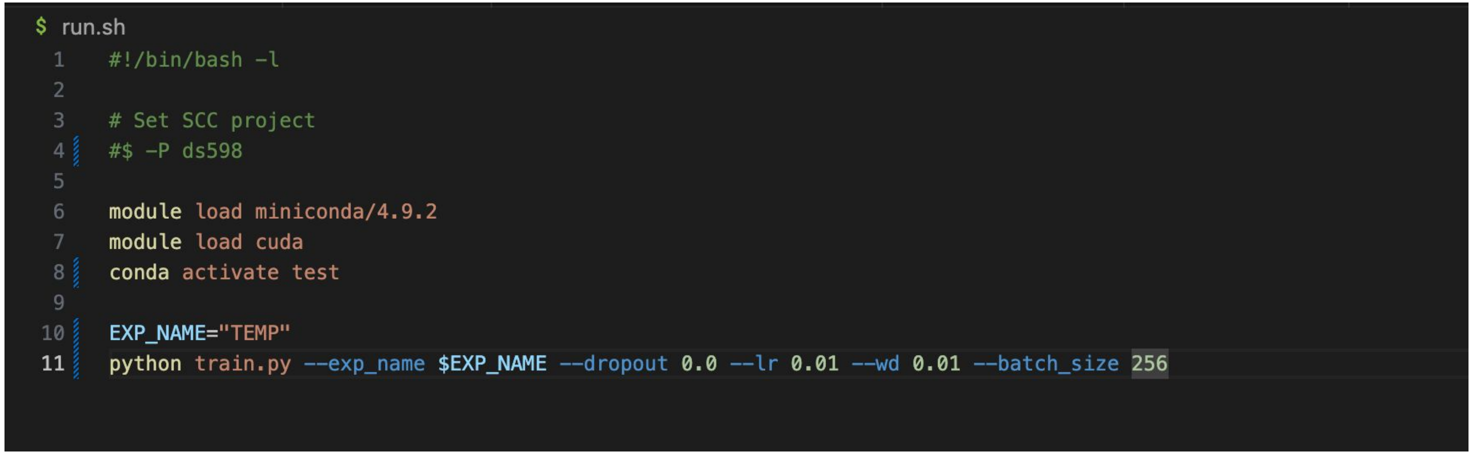
Requesting an interactive job with GPU

- Helpful for developing and debugging your DL models `qrsh -pe omp 4 -P ds598 -l gpus=1`
- You should be able to run your code with gpu after this. Try these:
- Cd to your /projectnb dir first, activate your conda env then:

```
print(torch.cuda.is_available())
t = torch.tensor([1,2,3])
t = t.cuda()
print(t)
```

Submitting a bash job script

- Once you have your model ready, run this for the entire training duration.
- Create a *.sh file, and then run
  qsub -l gpus=1 -l gpu_c=7.0 -pe omp 8 run.sh
- Check status with: qstat -u {user_name}
- The *.sh file contents looks something like this:



```
$ run.sh
#!/bin/bash -l

# Set SCC project
#$ -P ds598

module load miniconda/4.9.2
module load cuda
conda activate test

EXP_NAME="TEMP"
python train.py --exp_name $EXP_NAME --dropout 0.0 --lr 0.01 --wd 0.01 --batch_size 256
```

Monitoring a Submitted Job

```
scc % qstat -u userID
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
-
1000001 0.10000 network_gr  userID       r     05/19/2014 09:17:53 b@scc-bd7.scc.bu.edu               1
1000002 0.00000 network_gr  userID       qw    05/19/2014 09:14:53                                    16
```

- job-ID - Unique name for your job
- prior - Priority set by the Job scheduler
- name - You can set names to your job by passing the -N option to the qsub command
- state - The state of the job: (r) – running; (qw) – waiting to run; (hqw) – on hold, waiting to run; (Eqw) – job in error state; (s) – suspended; (t) – transfering.
- queue - The queue name and the node ID on which the job is running.
- slots - number of slots the job requested. (number of CPU cores)
- ja-task-ID - task id if submitted a job array (job array – submitting multiple jobs under a single job submission where each job has its own task ID, allowing the system to manage and execute each task individually).