Intro to Tensors & Pytorch

SCC setup instructions at the end

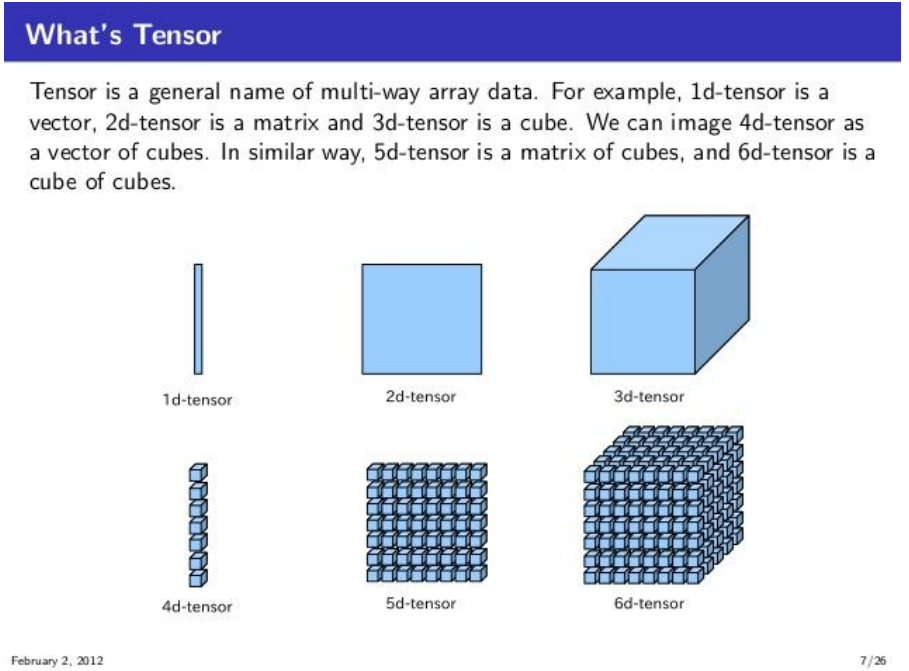
Outline

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

Starting from the basics…

What is a Tensor?

- Generic Answer: 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



source

More reading

- Tensor from the Math world: https://mathworld.wolfram.com/Tensor.html
- Tensor from the Physics world: https://www.physlink.com/education/askexperts/ae168.cfm
- Tensor from the ML world: A container to store your numerical values

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.

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:
(3, 28, 28, 3) → 3 images, 28 rows, 28 columns, 3 color channels (R, G, B)

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

Pytorch

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

Tensor Operations in Pytorch:

- Find a examples of tensor operations at:
  - https://github.com/DL4DS/sp2024_notebooks/blob/main/release/disc_nbs01/00_fundamentals.ipynb

SCC Setup

Connecting to SCC

- Launch **scc-ondemand.bu.edu** (check if you have '/projectnb /ds598 /' under files)
- Check if you can `ssh {your_username}@scc1.bu.edu` in your terminal.
- Recommended - Use VSCode - 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.
- 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/ds598/students/{Your_Folder_Name}/.conda
    - conda config --add envs_dirs /projectnb/ds598/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/ds598/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

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/sp2024/materials/

A few examples...

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:

```
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
```