8.md

08. Deep learning software

- This section changes a lot every year in CS231n due to rabid changes in the deep learning softwares.

- CPU vs GPU

  - GPU The graphics card was developed to render graphics to play games or make 3D media,. etc.
    - NVIDIA vs AMD
      - Deep learning choose NVIDIA over AMD GPU because NVIDIA is pushing research forward deep learning also makes it architecture more suitable for deep learning.
  - CPU has fewer cores but each core is much faster and much more capable; great at sequential tasks. While GPUs has more cores but each core is much slower "dumber"; great for parallel tasks.
  - GPU cores needs to work together. and has its own memory.
  - Matrix multiplication is from the operations that are suited for GPUs. It has MxN independent operations that can be done on parallel.
  - Convolution operation also can be paralyzed because it has independent operations.
  - Programming GPUs frameworks:
    - **CUDA** (NVIDIA only)
      - Write c-like code that runs directly on the GPU.
      - Its hard to build a good optimized code that runs on GPU. Thats why they provided high level APIs.
      - Higher level APIs: cuBLAS, cuDNN, etc
      - **CuDNN** has implemented back prop. , convolution, recurrent and a lot more for you!
      - In practice you won't write a parallel code. You will use the code implemented and optimized by others!
    - **OpenCl**
      - Similar to CUDA, but runs on any GPU.
      - Usually Slower .
      - Haven't much support yet from all deep learning softwares.
  - There are a lot of courses for learning parallel programming.
  - If you aren't careful, training can bottleneck on reading dara and transferring to GPU. So the solutions are:
    - Read all the data into RAM. # If possible
    - Use SSD instead of HDD
    - Use multiple CPU threads to prefetch data!
      - While the GPU are computing, a CPU thread will fetch the data for you.
      - A lot of frameworks implemented that for you because its a little bit painful!

- **Deep learning Frameworks**

  - Its super fast moving!
  - Currently available frameworks:
    - Tensorflow (Google)
    - Caffe (UC Berkeley)
    - Caffe2 (Facebook)
    - Torch (NYU / Facebook)
    - PyTorch (Facebook)
    - Theano (U monteral)
    - Paddle (Baidu)

        - CNTK (Microsoft)
        - MXNet (Amazon)
    - The instructor thinks that you should focus on Tensorflow and PyTorch.
    - The point of deep learning frameworks:
        - Easily build big computational graphs.
        - Easily compute gradients in computational graphs.
        - Run it efficiently on GPU (cuDNN - cuBLAS)
    - Numpy doesn't run on GPU.
    - Most of the frameworks tries to be like NUMPY in the forward pass and then they compute the gradients for you.

- **Tensorflow (Google)**

    - Code are two parts:
        a. Define computational graph.
        b. Run the graph and reuse it many times.
    - Tensorflow uses a static graph architecture.
    - Tensorflow variables live in the graph. while the placeholders are feed each run.
    - Global initializer function initializes the variables that lives in the graph.
    - Use predefined optimizers and losses.
    - You can make a full layers with layers.dense function.
    - **Keras** (High level wrapper):
        - Keras is a layer on top pf Tensorflow, makes common things easy to do.
        - So popular!
        - Trains a full deep NN in a few lines of codes.
    - There are a lot high level wrappers:
        - Keras
        - TFLearn
        - TensorLayer
        - tf.layers `#Ships with tensorflow`
        - tf-Slim `#Ships with tensorflow`
        - tf.contrib.learn `#Ships with tensorflow`
        - Sonnet `# New from deep mind`
    - Tensorflow has pretrained models that you can use while you are using transfer learning.
    - Tensorboard adds logging to record loss, stats. Run server and get pretty graphs!
    - It has distributed code if you want to split your graph on some nodes.
    - Tensorflow is actually inspired from Theano. It has the same inspirations and structure.

- **PyTorch (Facebook)**

    - Has three layers of abstraction:
        - Tensor: `ndarray` but runs on GPU `#Like numpy arrays in tensorflow`
            - Variable: Node in a computational graphs; stores data and gradient `#Like Tensor, Variable, Placeholders`
        - Module: A NN layer; may store state or learnable weights `#Like tf.layers in tensorflow`
    - In PyTorch the graphs runs in the same loop you are executing which makes it easier for debugging. This is called a dynamic graph.
    - In PyTorch you can define your own autograd functions by writing forward and backward for tensors. Most of the times it will implemented for you.
    - Torch.nn is a high level api like keras in tensorflow. You can create the models and go on and on.
        - You can define your own nn module!
    - Also Pytorch contains optimizers like tensorflow.

    - It contains a data loader that wraps a Dataset and provides minbatches, shuffling and multithreading.
    - PyTorch contains the best and super easy to use pretrained models
    - PyTorch contains Visdom that are like tensorboard. but Tensorboard seems to be more powerful.
    - PyTorch is new and still evolving compared to Torch. Its still in beta state.
    - PyTorch is best for research.

- Tensorflow builds the graph once, then run them many times (Called static graph)

- In each PyTorch iteration we build a new graph (Called dynamic graph)

- **Static vs dynamic graphs**:

    - Optimization:

        - With static graphs, framework can optimize the graph for you before it runs.

    - Serialization

        - **Static**: Once graph is built, can serialize it and run it without the code that built the graph. Ex use the graph in c++

        - **Dynamic**: Always need to keep the code around.

    - Conditional

        - Is easier in dynamic graphs. And more complicated in static graphs.

    - Loops:

        - Is easier in dynamic graphs. And more complicated in static graphs.

- Tensorflow fold make dynamic graphs easier in Tensorflow through dynamic batching.

- Dynamic graph applications include: recurrent networks and recursive networks.

- Caffe2 uses static graphs and can train model in python also works on IOS and Android

- Tensorflow/Caffe2 are used a lot in production especially on mobile.