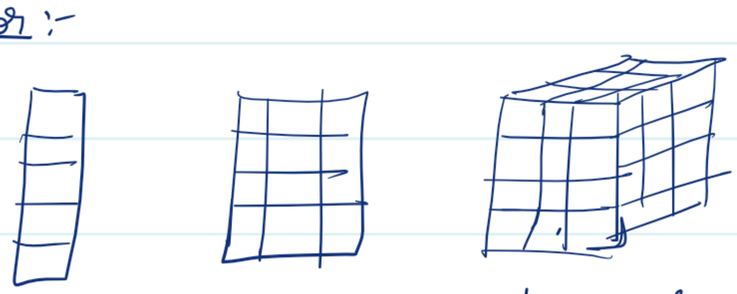
Tensorflow and Keras : Overview (3.1) :-

→ Best deep learning library as of now. Released in 2015. Before that people used Theano.

→ Can be used for research, development & deployment as well.

→ Core implementation is done in C/C++. Interfaces built in Python, Java and Javascript.

Tensor :-



tensor = n dim array.

Deep learning is all about tensor operations.

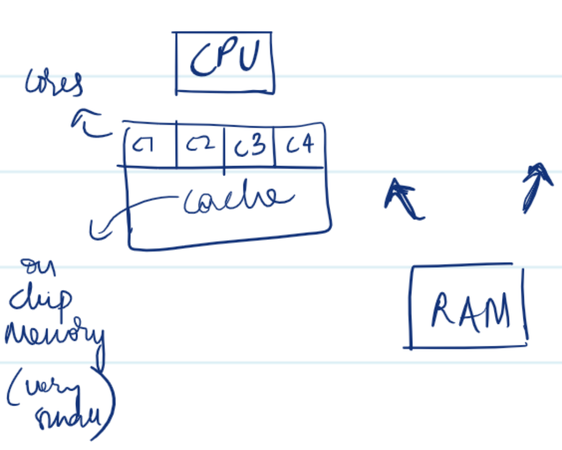
→ there is a lot of low level control in TF. Steep learning curve.

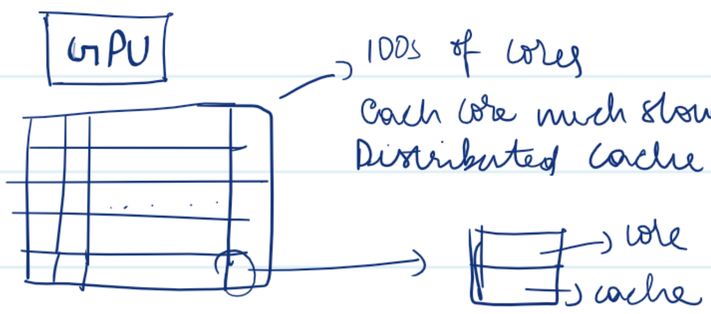
→ Keras has a lot of high level control & is easier to use. Keras internally uses tensorflow as well.

There are 2 backends for Keras : tensorflow and Theano.

→ keras 2 is more deeply integrated with the tensorflow and has some low level control as well.

CPU v/s GPU (3.2) :-





Each core much slower than CPU core (1GHz compared to CPU cores)
Distributed cache.

Sum of all cache = VRAM.

CUDA = programming env for Nvidia GPUs.

GPU can do lots of parallel computations.

GPUs are better if we have parallelizable tasks. Deep learning is highly parallelizable.

Softmax Classifier on MNIST (3.6) :-

→ Tensorflow has constants, variables and placeholders.
(as seen in python)

ex:- $W^T X + b$ (W, b : variables)

placeholder :- typically used to store data like mini batches etc; Memory allocated for the task.
Think of it as a modified form of variables.

Rule of thumb for number of neurons, $N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$

N_s = Number of samples in training dataset
N_i = Number of input neurons
N_o = Number of output neurons
α = arbitrary scaling factor 2-10. Sometimes 5-10.

Model 3 : Batch Normalization (3.10) :-

→ In batch normalization x's are not the ones that need to normalized. It's the $(W^T x + b)$ that should be normalized.