HARDWARE ACCELERATED

Module 2

v-cardona

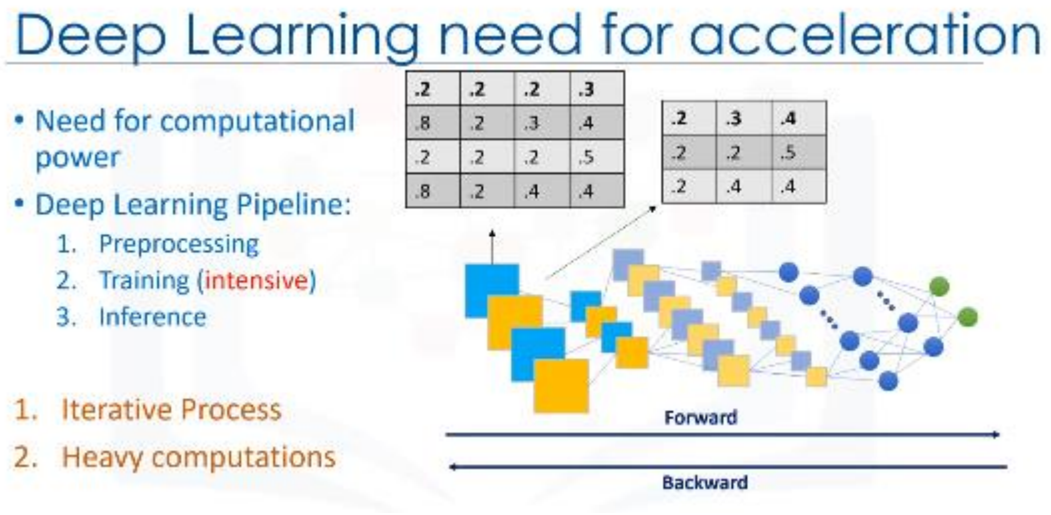
Using GPUs to Scale and Speed-up Deep Learning

Contenido

GPU-Accelerated deep learning introduction

You can use Google’s Tensor Processing Unit (or TPU) or Nvidia GPU to accelerate your deep neural network computation time. With that we can accelerated the computation of training and not waste days or weeks to train a model.

Parallelism in deep learning



Deep Learning need for acceleration

- Need for computational power
- Deep Learning Pipeline:
  1. Preprocessing
  2. Training (intensive)
  3. Inference

1. Iterative Process
2. Heavy computations

.2	.2	.2	.3
.8	.2	.3	.4
.2	.2	.2	.5
.8	.2	.4	.4

.2	.3	.4
.2	.2	.5
.2	.4	.4

Deep Learning with GPU vs CPU

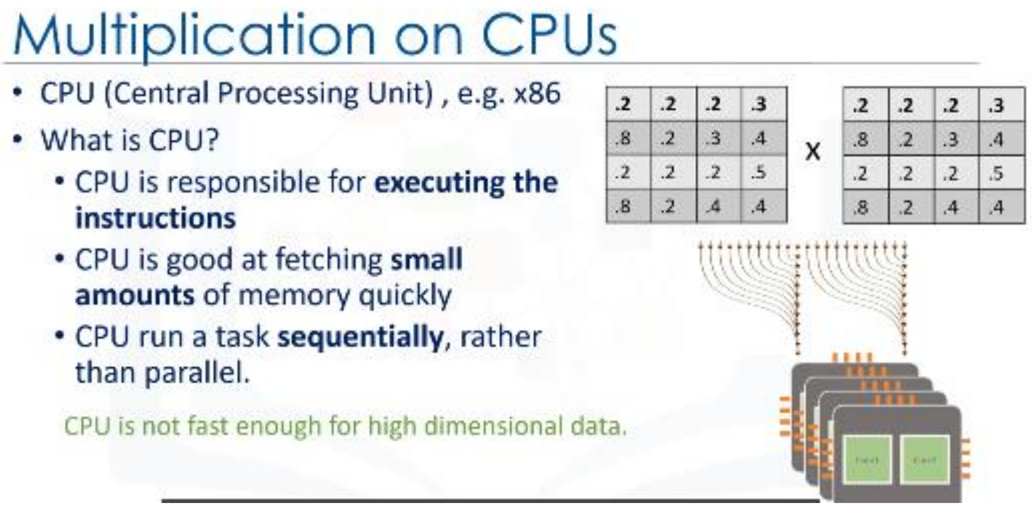
A single core CPU performs matrix operations in series, one element at a time. But, a single GPU could have hundreds or thousands of cores that allows it to perform multiple matrix operations in parallel, while a multi-core CPU typically has only a few cores and the number of operations it can do in parallel is limited by the number of cores it has. This is why GPU over performs CPU.

If a step of the process encompass “do this mathematical operation many times”, then it is a good candidate operation to send it to be run on the GPU. For example:

- Matrix multiplication
- Computing the inverse of a matrix.
- Gradient calculations, which are computed on multiple GPUs individually and are averaged on the CPU

GPUs do not have direct access to the rest of your computer (except, of course for the display). Due to this, if you are running a command on a GPU, you need to copy all of the data to the GPU first, then do the operation, after that copy the result back to your computer’s main memory, so for this types of operation we should not use GPU. Although, TensorFlow handles this under the hood, so the code is simple, but the work still needs to be performed.

The CPU (or Central Processing Unit) is the part of a computer system that is known as the processor or microprocessor. The CPU is responsible for executing a sequence of stored instructions, which in our case are multiplications. We need to fetch data and instructions from main memory to be run by the CPU. CPU is good at fetching small amounts of memory quickly, but not very good for big chunks of data, like big matrices, which are needed for deep learning. CPUs run tasks sequentially, rather than in parallel, even though they have 2 or 4 cores.



Multiplication on CPUs

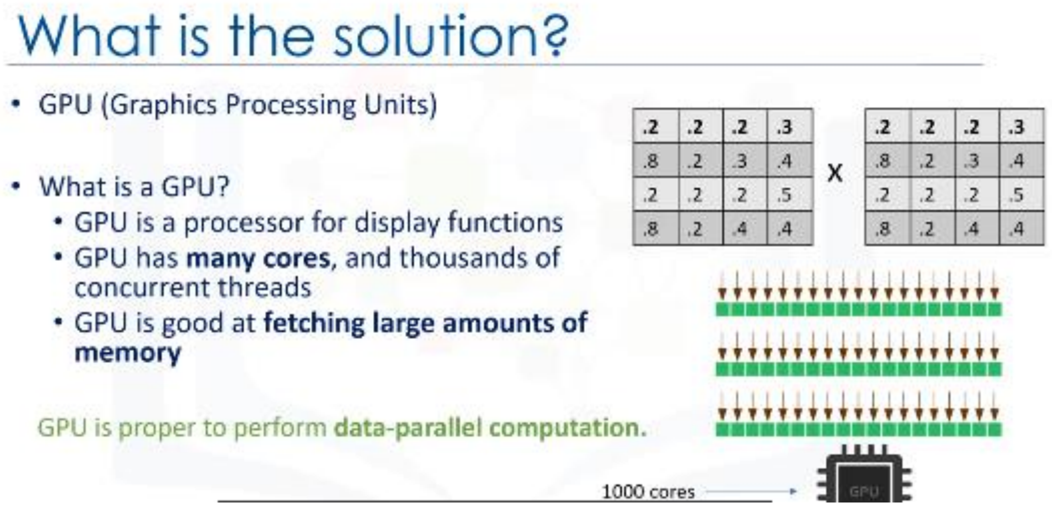
- CPU (Central Processing Unit) , e.g. x86
- What is CPU?
  - CPU is responsible for **executing the instructions**
  - CPU is good at fetching **small amounts** of memory quickly
  - CPU run a task **sequentially**, rather than parallel.

CPU is not fast enough for high dimensional data.

$$\begin{array}{|c|c|c|c|}\hline .2 & .2 & .2 & .3 \\ \hline .8 & .2 & .3 & .4 \\ \hline .2 & .2 & .2 & .5 \\ \hline .8 & .2 & .4 & .4 \\ \hline \end{array} \times \begin{array}{|c|c|c|c|}\hline .2 & .2 & .2 & .3 \\ \hline .8 & .2 & .3 & .4 \\ \hline .2 & .2 & .2 & .5 \\ \hline .8 & .2 & .4 & .4 \\ \hline \end{array}$$

We can conclude that CPUs are not fast enough for operations on big chunks of data and they are not the proper use for high parallelism, as they are very slow for these kinds of tasks.

A GPU is a chip (or processor) traditionally designed and specialized for rendering images, animations and video for the computer's screen. A GPU has many cores, sometimes up to 1000 cores, so it can handle many computations. In addition, a GPU is good at fetching large amounts of memory. So, although GPUs were invented and used for better and more general graphics processing, they were later found to fit scientific computing as well.

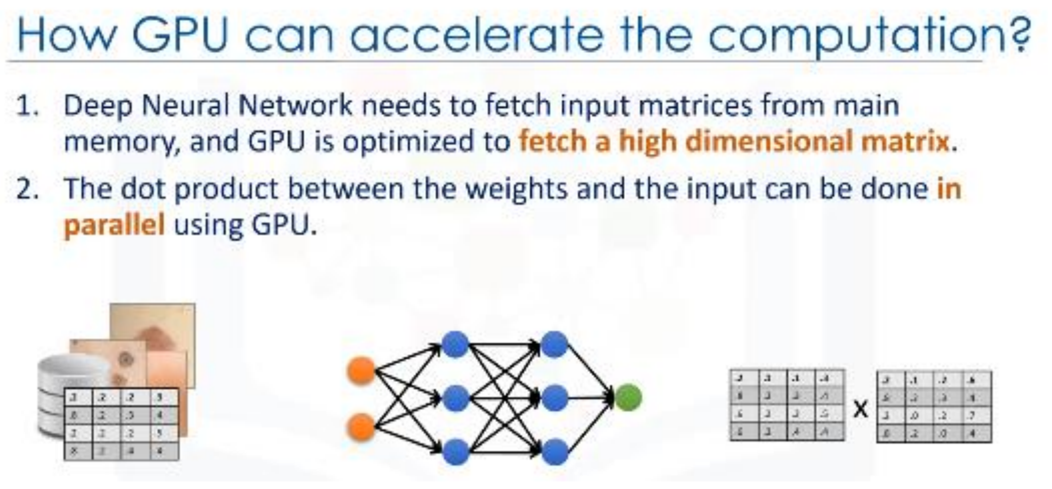


What is the solution?

- GPU (Graphics Processing Units)
- What is a GPU?
  - GPU is a processor for display functions
  - GPU has **many cores**, and thousands of concurrent threads
  - GPU is good at **fetching large amounts of memory**

GPU is proper to perform **data-parallel computation.**

$$\begin{array}{|c|c|c|c|}\hline .2 & .2 & .2 & .3 \\ \hline .8 & .2 & .3 & .4 \\ \hline .2 & .2 & .2 & .5 \\ \hline .8 & .2 & .4 & .4 \\ \hline \end{array} \times \begin{array}{|c|c|c|c|}\hline .2 & .2 & .2 & .3 \\ \hline .8 & .2 & .3 & .4 \\ \hline .2 & .2 & .2 & .5 \\ \hline .8 & .2 & .4 & .4 \\ \hline \end{array}$$



How GPU can accelerate the computation?

1. Deep Neural Network needs to fetch input matrices from main memory, and GPU is optimized to **fetch a high dimensional matrix**.
2. The dot product between the weights and the input can be done **in parallel** using GPU.

Single-GPU vs Multi-GPU Deep learning

It is important to note that if both CPU and GPU are available on the machine that you are running the notebook, and if a TensorFlow operation has both CPU and GPU implementations, the GPU devices will be given priority when the operation is assigned to a device.

To use multi GPU for training using TensorFlow You need a server with supports for multi-GPU calculation. Given multiple GPU cards, you can run the training of a model on each GPU as follows:

1. Store all model parameters on the CPU, e.g weights and biases.
2. Make a copy of the network and send it to each GPU, including conv2d, pre-activation, relu, etc.
3. Divide up a large batch of data across the GPUs, and feed each model replica with a unique batch of data.
4. Compute the inference on each GPU
5. Calculate the gradients on each GPU
6. Wait until all GPUs finish processing of their batches
7. Update model parameters synchronously on CPU