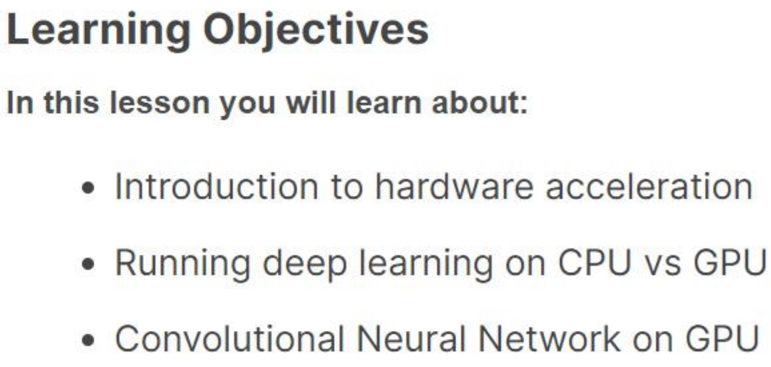
Module 2 - Hardware Accelerated Deep Learning

- 1.1 GPU-Accelerated Deep Learning Introduction
- 1.2 Parallelism in Deep Learning
- 1.3 Deep Learning with GPU vs CPU
- 1.4 Lab: Using GPU for Deep Learning
- 1.5 Lab: CNN on GPU
- 1.6 Single-GPU vs Multi-GPU Deep Learning
- 1.7 Graded Review Questions (2 Questions)



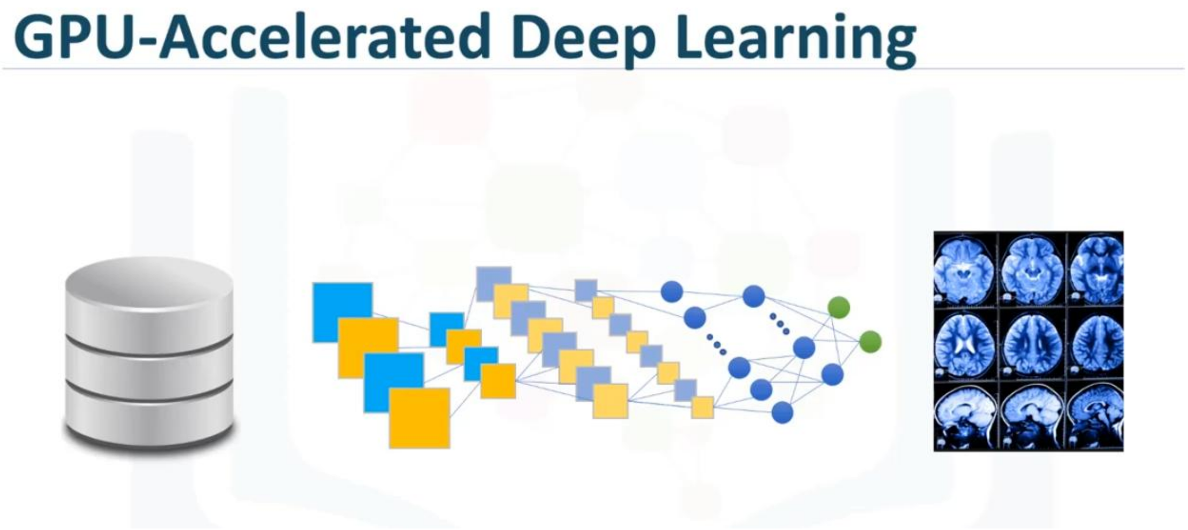
Learning Objectives

In this lesson you will learn about:

- Introduction to hardware acceleration
- Running deep learning on CPU vs GPU
- Convolutional Neural Network on GPU

总结：

2.1 GPU-Accelerated Deep Learning Introduction



GPU-Accelerated Deep Learning

Hello, and welcome! In the following videos, we'll be reviewing GPU-accelerated deep learning.

Have you ever tried to train a complex deep learning model with huge data? You should expect hours, days, or sometimes weeks, to train a complex model with a large dataset. So, what's the solution? Well, you should use accelerated hardware. For example, you can use Google's Tensor Processing Unit (or TPU) or Nvidia GPU to accelerate your deep neural network computation time.

In the following videos, I will talk about what kind of hardware would be better for doing deep learning, and how GPU can help to accelerate it. I assume that you have fundamental knowledge of deep learning concepts. If not, first go through the previous videos or courses about deep learning and then come back to this module. Thanks for watching this video.

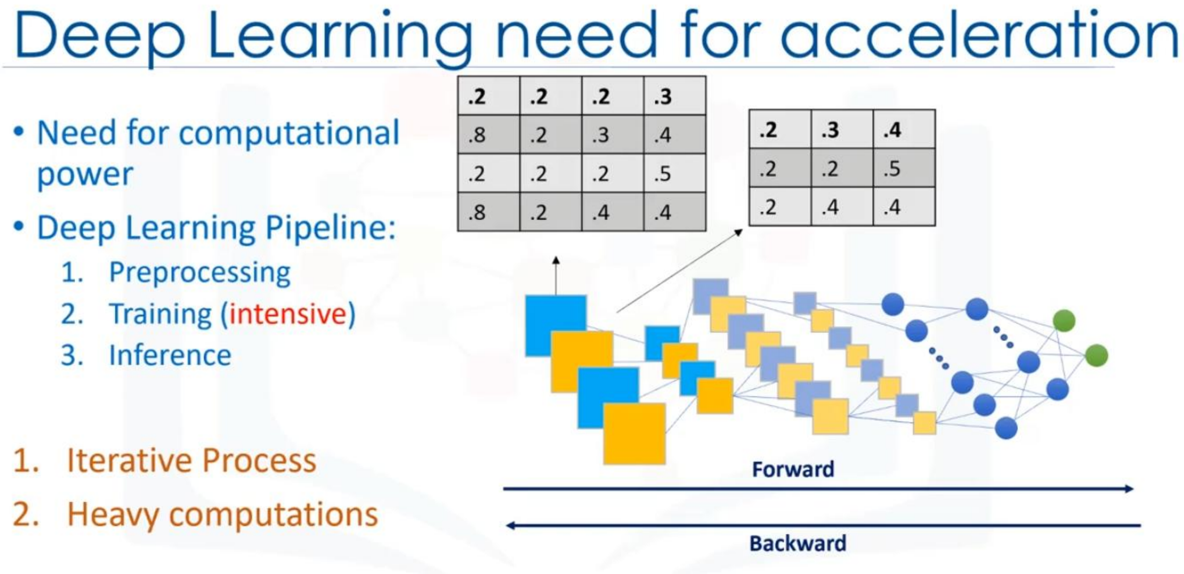
总结：

2.2 Parallelism in Deep Learning

Parallelism in Deep Learning

How to accelerate the training process?

Saeed Aghabozorgi



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

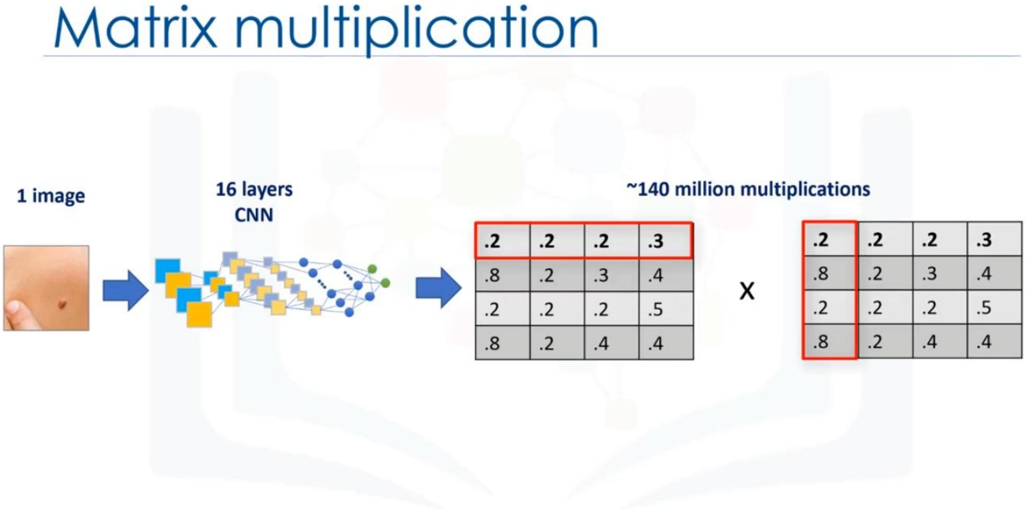
.2	.2	.2	.3
.8	.2	.3	.4
.2	.2	.2	.5
.8	.2	.4	.4

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

In this video, we dive into the training part of Deep Learning. We also see why the training part is slow and how we can accelerate the process through parallelism.

As mentioned before, some machine learning, and most of Deep Neural Networks, need a lot of computational power for building a model. We also mentioned that of the three phases in the deep learning pipeline, the training phase of the modeling is the most intensive task, and the most time consuming one. Why? It essentially comes down to two reasons:

- **First, deep learning is an iterative process**. When you train a deep learning model, two main operations are performed: **Forward Pass** and **Backward Pass**.
  - **In forward pass**, input is passed through the neural network and after processing the input, an output is generated.
  - Whereas **in the backward pass**, we update the weights of the neural network on the basis of the error(s) we get in the forward pass.
- **Second, deep learning involves heavy computations**. For example, in a Convolutional Neural Network, each pixel within a single image becomes a feature point after being multiplied by the color channel. We can consider the first array as the input to the neural network, and the second array can be considered as weights or a filter of the network. The size of these matrices are usually very big, that is, the data is high-dimensional here. So, considering that training is an iterative process, and Neural Networks have usually many weights, which should get updated with each iteration, it involves expensive computations that are **mostly matrix multiplication**. Therefore, Deep Learning requires much computing power.



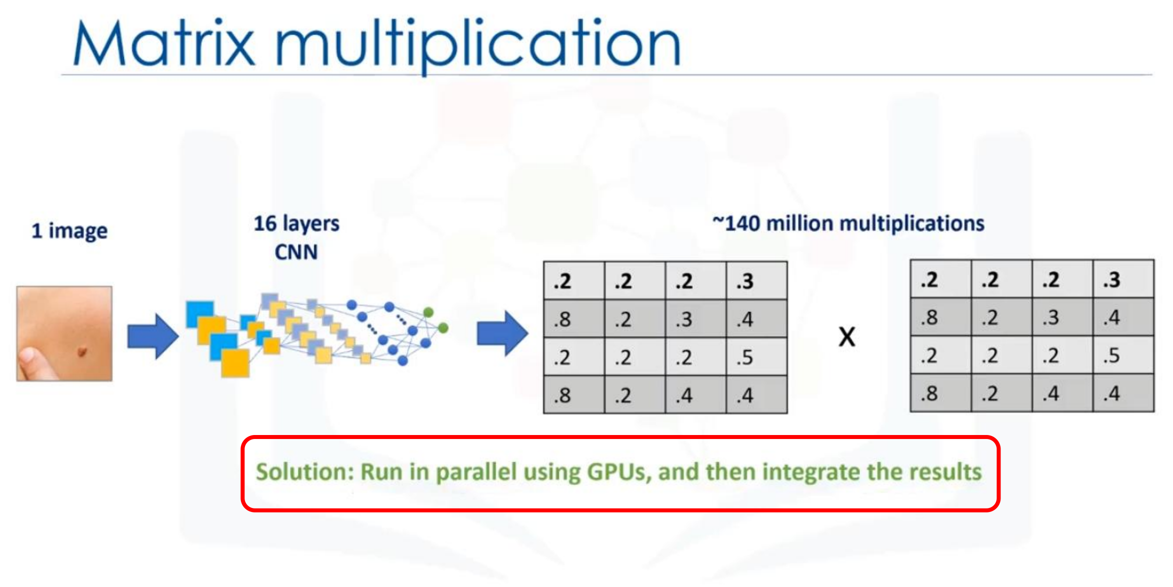
Matrix multiplication

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

X

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

总结：



Matrix multiplication

1 image

16 layers CNN

~140 million multiplications

$$\begin{bmatrix} .2 & .2 & .2 & .3 \\ .8 & .2 & .3 & .4 \\ .2 & .2 & .2 & .5 \\ .8 & .2 & .4 & .4 \end{bmatrix} \times \begin{bmatrix} .2 & .2 & .2 & .3 \\ .8 & .2 & .3 & .4 \\ .2 & .2 & .2 & .5 \\ .8 & .2 & .4 & .4 \end{bmatrix}$$

Solution: Run in parallel using GPUs, and then integrate the results

Let’s assume we just want to run a dot product of a 2 matrices; it includes many multiplications, right? The first row of the first matrix multiplied by the first column of the second matrix, and so on. The challenging part is that we have many of these multiplications in deep learning. For example, a convolutional neural network with 16 layers, has around 140 million weights and biases. Now imagine how many matrix multiplications you would have to do to a pass of just one image to this network! And think how many images you have in your training data set, and how many iterations you should do till your network learns properly. **It would take a long time to train this kind of model if we use a sequential computational approach**.

Now, think of doing all the operations at the same time instead of doing them one after the other. That is, the multiplications can be run in parallel, and we then integrate the results. It would most definitively speed up the process. This is called **parallelism**, which is very desirable in expensive computational tasks. **GPU will do the parallelism for us**.

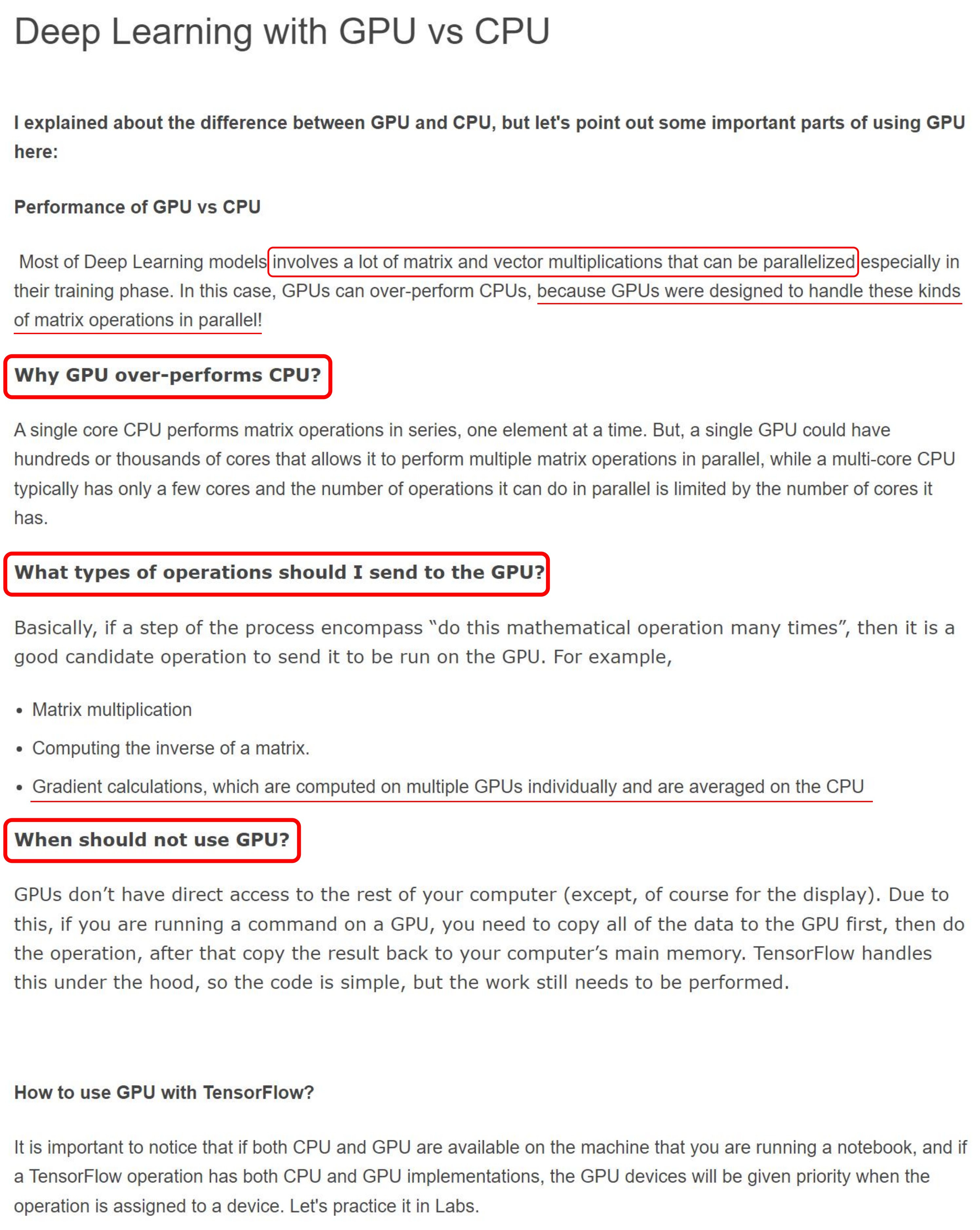
But before we explain how GPU can do it for us, let’s see why CPU cannot.

Thanks for watching this video!

总结：

2.3 Deep Learning with GPU vs CPU

详见下载下来的网页：
D:\KeepStudy\01_Edx\1 Using GPUs to Scale and Speed-up Deep Learning (IBM DL0122EN)**M2_3 Deep Learning with GPU vs CPU.html**



Deep Learning with GPU vs CPU

I explained about the difference between GPU and CPU, but let's point out some important parts of using GPU here:

Performance of GPU vs CPU

Most of Deep Learning models involves a lot of matrix and vector multiplications that can be parallelized especially in their training phase. In this case, GPUs can over-perform CPUs, because GPUs were designed to handle these kinds of matrix operations in parallel!

Why GPU over-performs 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.

What types of operations should I send to the GPU?

Basically, 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

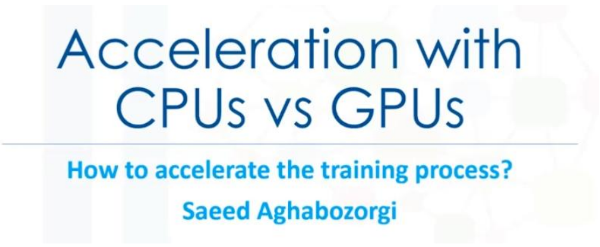
When should not use GPU?

GPUs don’t 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. TensorFlow handles this under the hood, so the code is simple, but the work still needs to be performed.

How to use GPU with TensorFlow?

It is important to notice that if both CPU and GPU are available on the machine that you are running a 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. Let's practice it in Labs.

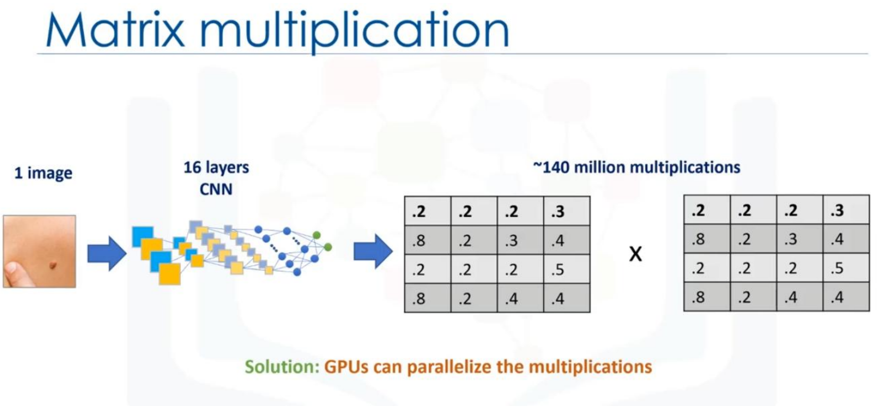
总结：



Acceleration with CPUs vs GPUs

How to accelerate the training process?

Saeed Aghabozorgi



Matrix multiplication

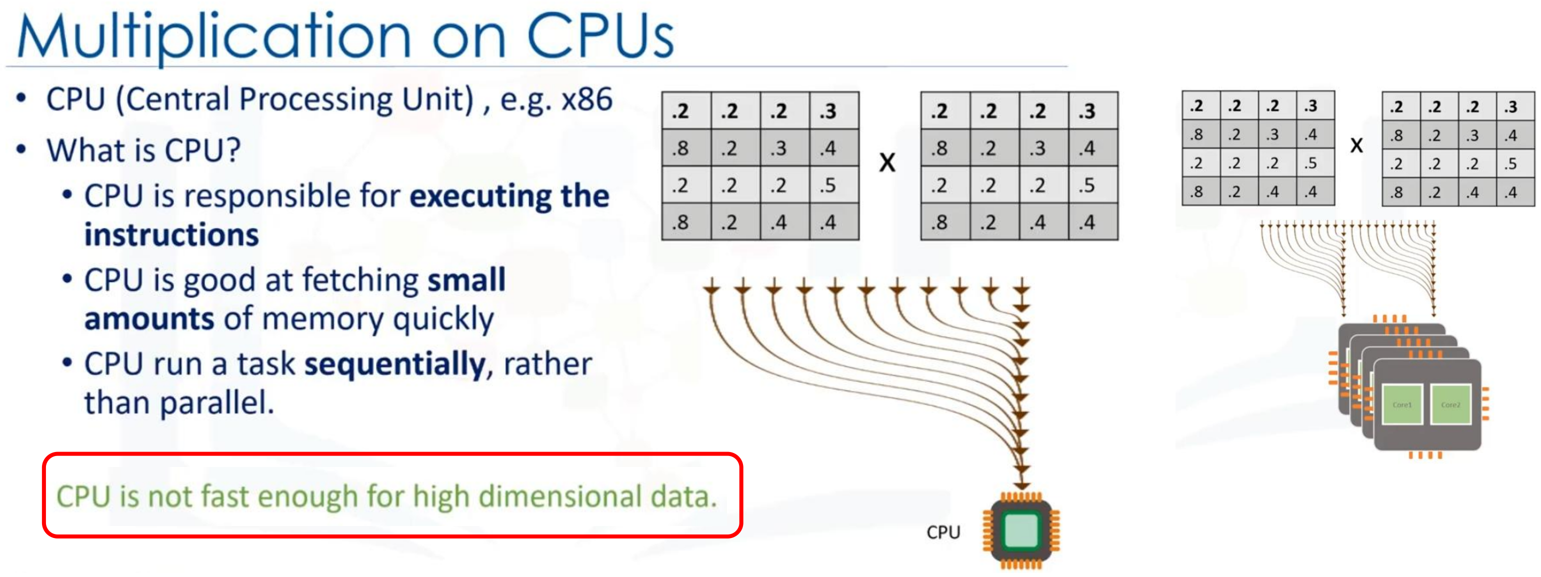
.2	.2	.2	.3
.8	.2	.3	.4
.2	.2	.2	.5
.8	.2	.4	.4

X

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

Solution: GPUs can parallelize the multiplications

In this video, we look at CPUs and GPUs, and see which one is better for accelerating the Deep Learning process. As mentioned before, the deep learning process, specifically its training phase, is an iterative process, and each iteration involves many matrix multiplications. We also noted that these multiplications can be run in parallel, to speed up the process; and that GPUs can certainly do this for us. But before I explain how GPUs do this, let's see why CPUs cannot?



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.

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

X

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

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

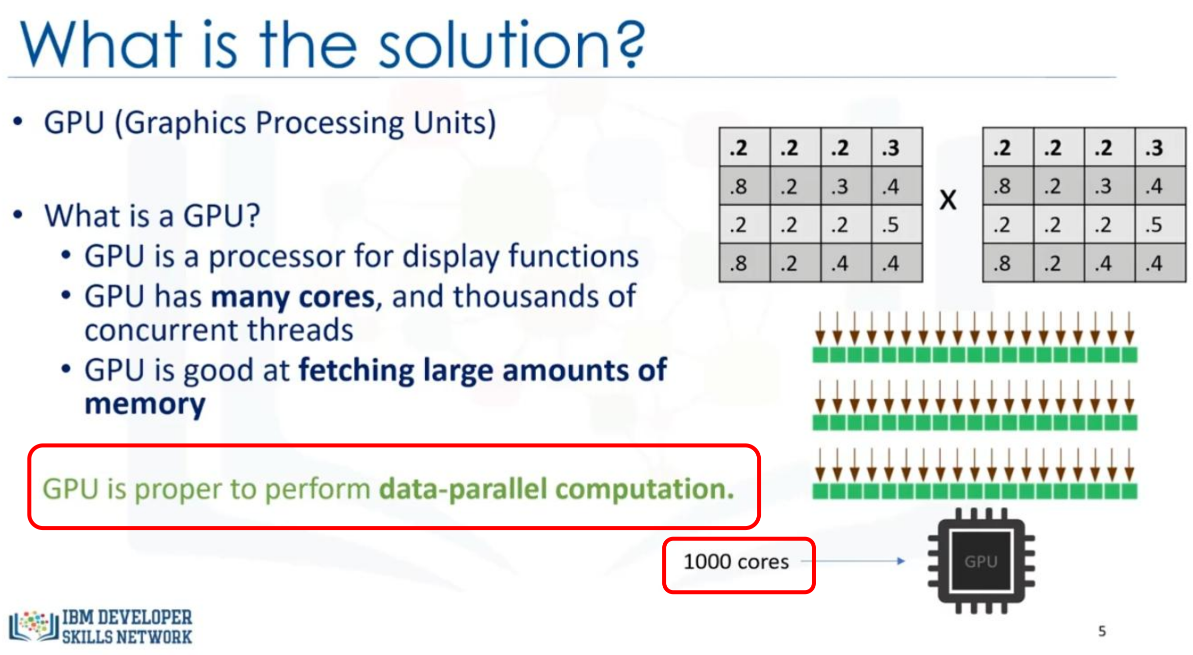
X

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

The **CPU** (or **Central Processing Unit**) is the part of a computer system that is known as the processor or microprocessor. You probably heard of the x86, which is one of the more prevalent CPUs by Intel.
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. Some companies used to build multiple clusters of CPUs to have a powerful system to do their processing in parallel, for example, built for training huge nets. These systems are usually very expensive and, as such, most businesses can't afford them.

So 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.

总结：



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.**

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

X

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

1000 cores

IBM DEVELOPER SKILLS NETWORK

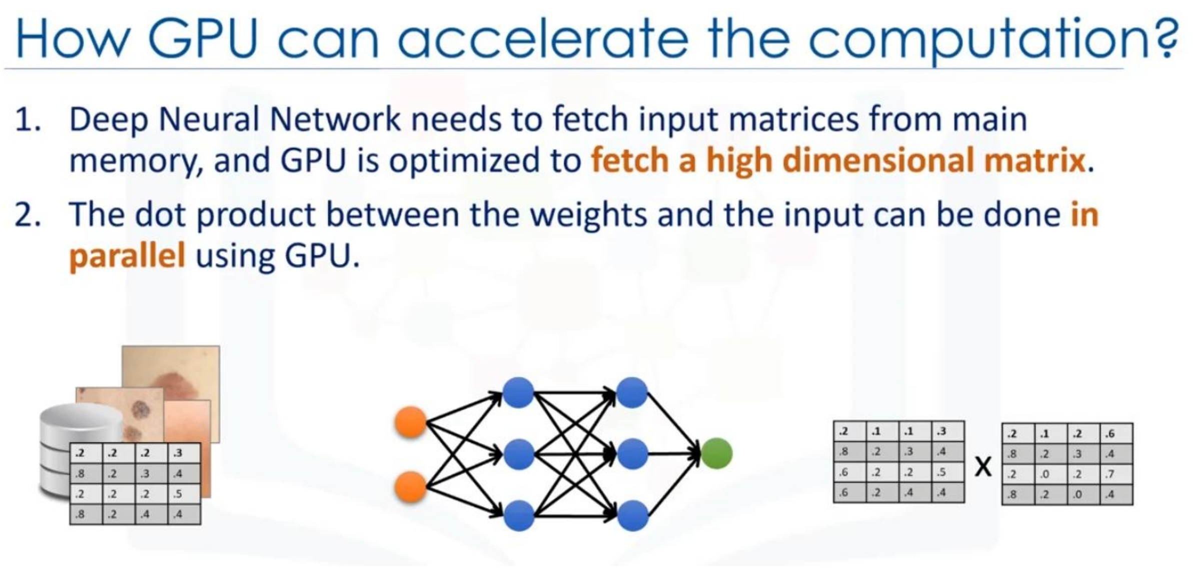
5

So, what is the solution? Well, the solution is **Graphics Processing Units**, or **GPUs** for short.

So, "What is a GPU and how can it accelerate the computation?"

- 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.
- Also, 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. We can say GPU computing is "A Paradigm Shift in Programming", designed to efficiently process massive data parallelism with huge datasets.



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.

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

.2	.1	.1	.3
.8	.2	.3	.4
.6	.2	.2	.5
.6	.2	.4	.4

X

.2	.1	.2	.6
.8	.2	.3	.4
.2	.0	.2	.7
.8	.2	.0	.4

Technically, there are many intricate reasons why a GPU is much better for handling matrices multiplication than a CPU. However, to thoroughly review all those intricacies is beyond the scope of this video. So conclude with the main reasons why GPUs are a good match for deep learning.

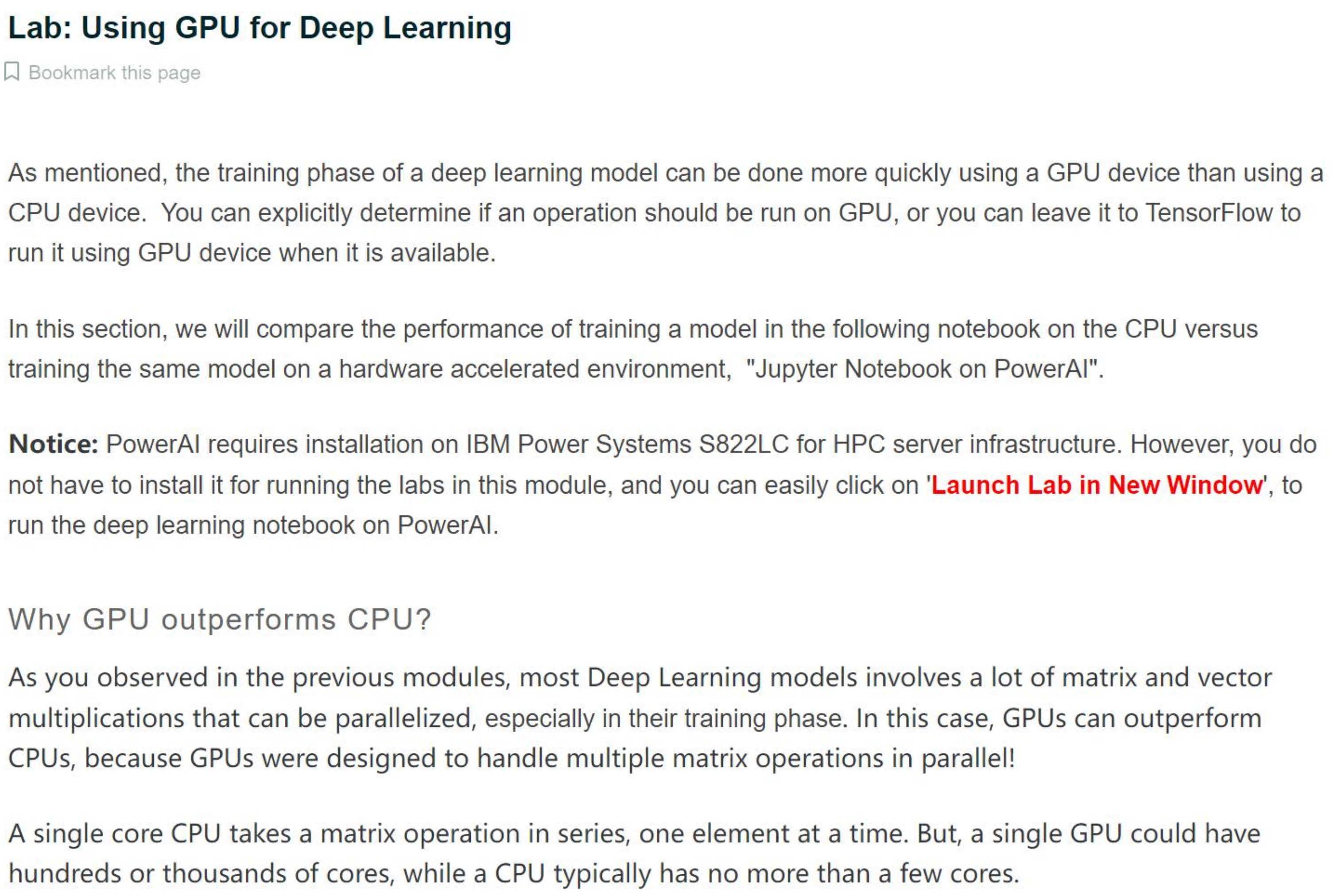
- First, recall that Deep Neural Networks need to fetch input images as matrices from main memory, and GPUs perform very well at fetching big chunks of memory. So, we can say that GPUs are optimized to properly fetch such high-dimensional matrices.
- Second, GPUs are the proper choice for parallelism operations on matrices. That is, they are very good where the same code runs on different sections of the same array.

Now, recall that a Deep Neural Network is a matrix of weights where the dot product between the weights and the input image needs to be done many times. GPUs can also use their ability for parallelism to do all the multiplications at the same time, instead of doing them one after the other. This type of concurrent processing is exactly what's required in deep learning. **So, the GPU parallelism feature reduces the computation times of the dot product of big matrices**. Thanks for watching this video!

总结：

2.4 Lab: Using GPU for Deep Learning

详见下载下来的网页：
D:\KeepStudy\01_Edx\1 Using GPUs to Scale and Speed-up Deep Learning (IBM DL0122EN)**M2_4 Lab - Using GPU for Deep Learning.html**
配套代码：
D:\KeepStudy\01_Edx\1 Using GPUs to Scale and Speed-up Deep Learning (IBM DL0122EN)**M2_4 Using GPU for Deep Learning - GPU vs CPU.ipynb**



Lab: Using GPU for Deep Learning

Bookmark this page

As mentioned, the training phase of a deep learning model can be done more quickly using a GPU device than using a CPU device. You can explicitly determine if an operation should be run on GPU, or you can leave it to TensorFlow to run it using GPU device when it is available.

In this section, we will compare the performance of training a model in the following notebook on the CPU versus training the same model on a hardware accelerated environment, "Jupyter Notebook on PowerAI".

Notice: PowerAI requires installation on IBM Power Systems S822LC for HPC server infrastructure. However, you do not have to install it for running the labs in this module, and you can easily click on '**Launch Lab in New Window**', to run the deep learning notebook on PowerAI.

Why GPU outperforms CPU?

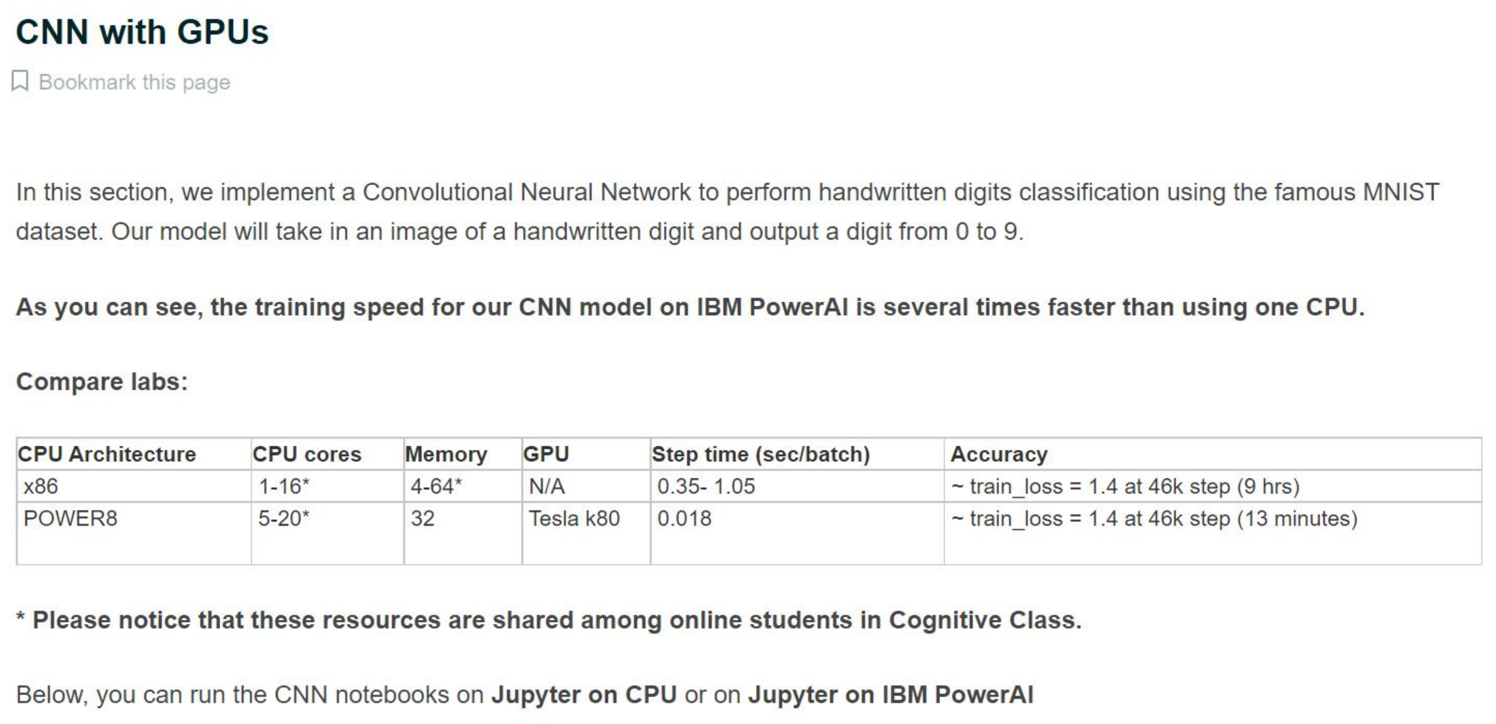
As you observed in the previous modules, most Deep Learning models involves a lot of matrix and vector multiplications that can be parallelized, especially in their training phase. In this case, GPUs can outperform CPUs, because GPUs were designed to handle multiple matrix operations in parallel!

A single core CPU takes a matrix operation in series, one element at a time. But, a single GPU could have hundreds or thousands of cores, while a CPU typically has no more than a few cores.

总结：

2.5 Lab: CNN on GPU

详见下载下来的网页：
D:\KeepStudy\01_Edx\1 Using GPUs to Scale and Speed-up Deep Learning (IBM DL0122EN)**M2_5 Lab - CNN on GPU.html**
配套代码：
D:\KeepStudy\01_Edx\1 Using GPUs to Scale and Speed-up Deep Learning (IBM DL0122EN)**M2_5 CNN - CPU.ipynb**
D:\KeepStudy\01_Edx\1 Using GPUs to Scale and Speed-up Deep Learning (IBM DL0122EN)**M2_5 CNN - GPUs with PowerAI.ipynb**



CNN with GPUs

Bookmark this page

In this section, we implement a Convolutional Neural Network to perform handwritten digits classification using the famous MNIST dataset. Our model will take in an image of a handwritten digit and output a digit from 0 to 9.

As you can see, the training speed for our CNN model on IBM PowerAI is several times faster than using one CPU.

Compare labs:

CPU Architecture	CPU cores	Memory	GPU	Step time (sec/batch)	Accuracy
x86	1-16*	4-64*	N/A	0.35- 1.05	~ train_loss = 1.4 at 46k step (9 hrs)
POWER8	5-20*	32	Tesla k80	0.018	~ train_loss = 1.4 at 46k step (13 minutes)

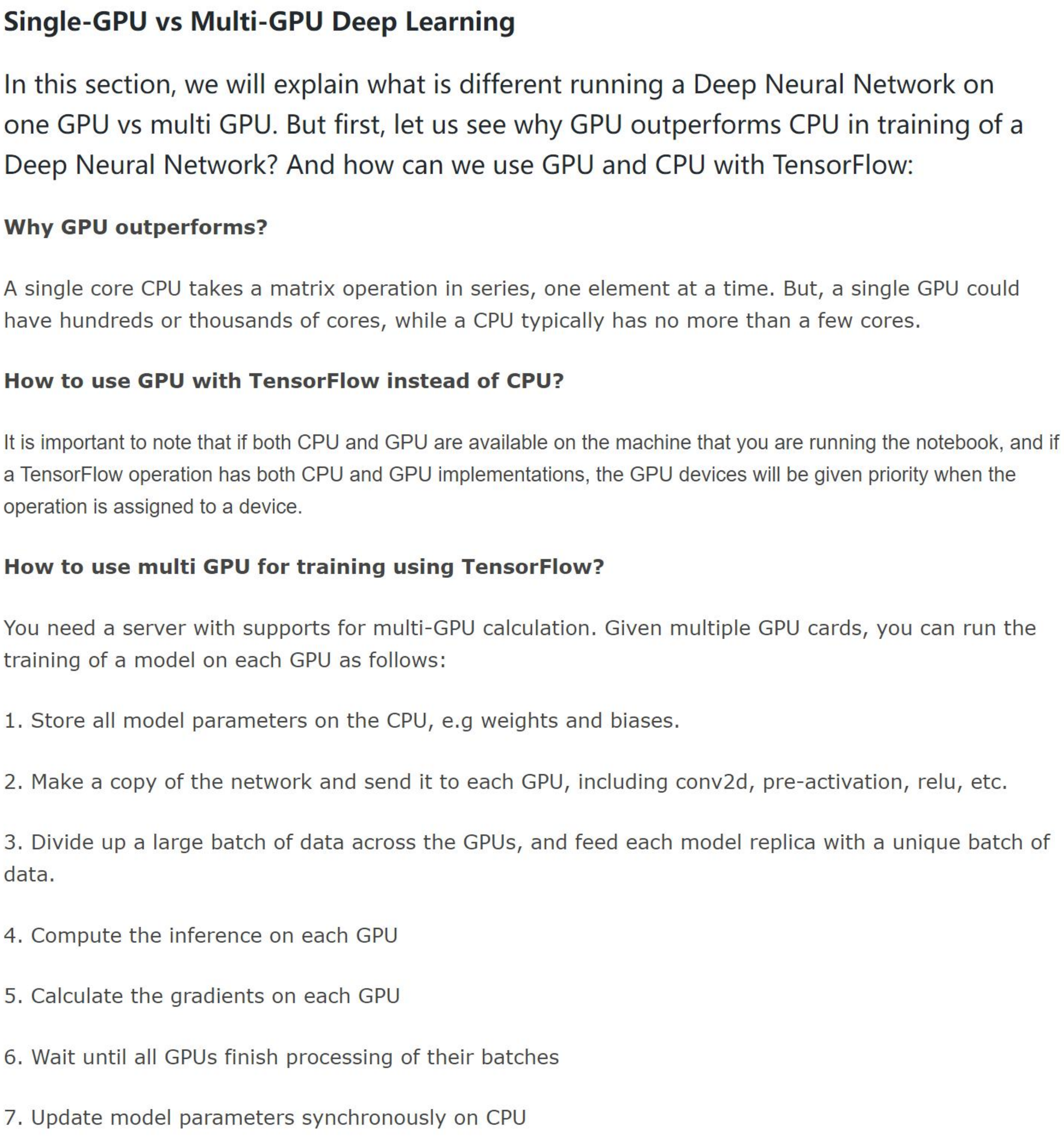
*** Please notice that these resources are shared among online students in Cognitive Class.**

Below, you can run the CNN notebooks on **Jupyter on CPU** or on **Jupyter on IBM PowerAI**

总结：

2.6 Single-GPU vs Multi-GPU Deep Learning

详见下载下来的网页：
D:\KeepStudy\01_Edx\1 Using GPUs to Scale and Speed-up Deep Learning (IBM DL0122EN)**M2_6 Single-GPU vs Multi-GPU Deep Learning.html**



Single-GPU vs Multi-GPU Deep Learning

In this section, we will explain what is different running a Deep Neural Network on one GPU vs multi GPU. But first, let us see why GPU outperforms CPU in training of a Deep Neural Network? And how can we use GPU and CPU with TensorFlow:

Why GPU outperforms?

A single core CPU takes a matrix operation in series, one element at a time. But, a single GPU could have hundreds or thousands of cores, while a CPU typically has no more than a few cores.

How to use GPU with TensorFlow instead of CPU?

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.

How 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

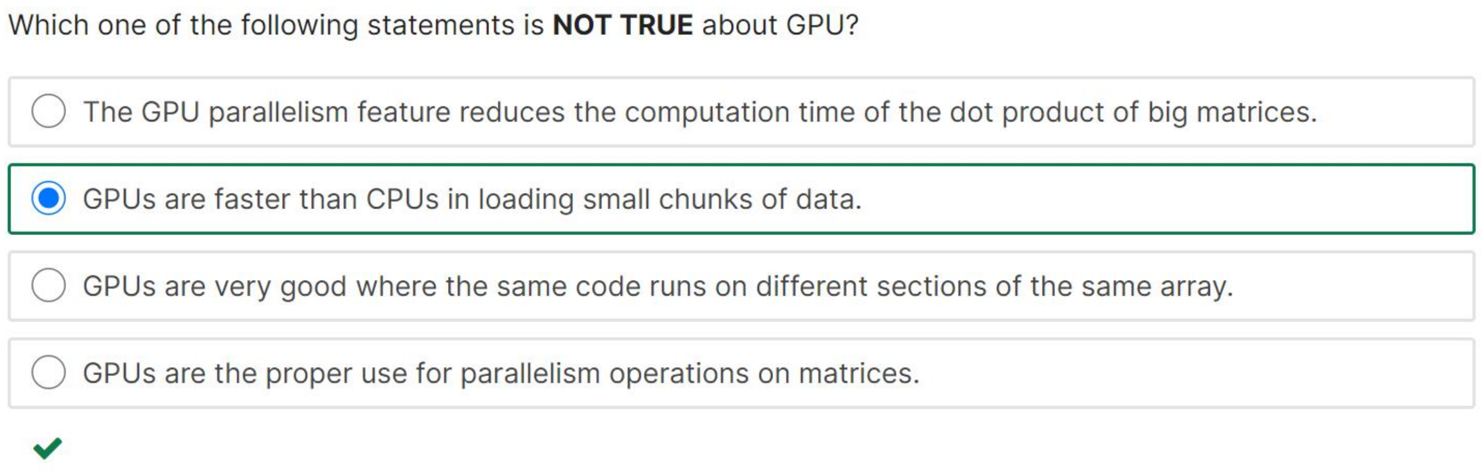
5. Calculate the gradients on each GPU

6. Wait until all GPUs finish processing of their batches

7. Update model parameters synchronously on CPU

总结：

2.7 Graded Review Questions (2 Questions)



Which one of the following statements is **NOT TRUE** about GPU?

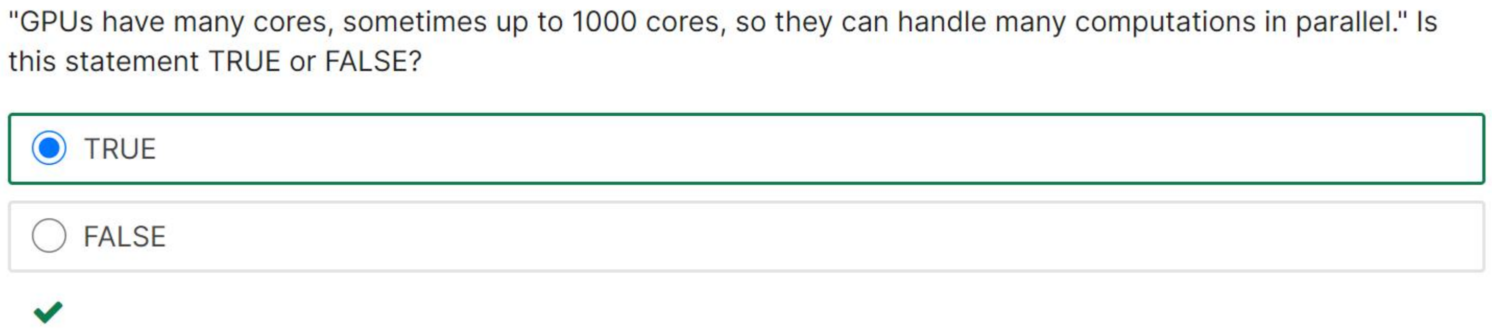
○ The GPU parallelism feature reduces the computation time of the dot product of big matrices.

● GPUs are faster than CPUs in loading small chunks of data.

○ GPUs are very good where the same code runs on different sections of the same array.

○ GPUs are the proper use for parallelism operations on matrices.

✓



"GPUs have many cores, sometimes up to 1000 cores, so they can handle many computations in parallel." Is this statement TRUE or FALSE?

● TRUE

○ FALSE

✓

总结：