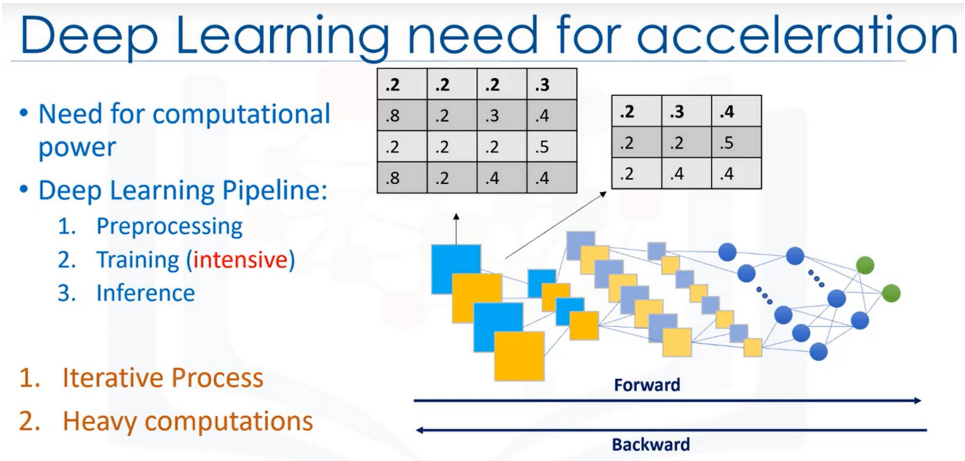
Mod 2: HW accelerated DL

Tuesday, April 21, 2020 10:30 AM

PARALLELISM IN DEEP LEARNING

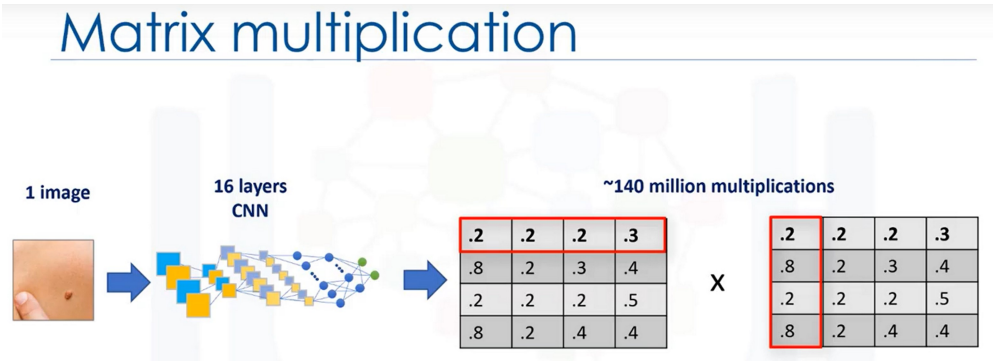


La etapa de entrenamiento es la mas intensiva y la que demora más por las siguientes razones (ej. CNN)

1. DL es un proceso iterativo porque hay 2 etapas que se repiten varias veces:
   - Forward pass: Esto es cuando las muestras/imágenes ingresan a la red neuronal. Una vez procesadas, se genera una salida (un error)
   - Backward pass: Los pesos de la red neuronal se actualizan según el error que se obtuvo en el forward pass

2. DL utiliza cálculos muy pesados
   En una red CNN (Convolutional Neural Network), cada pixel de cada imagen es multiplicado por el canal de color (RGB). Y este producto sera un atributo

La primera matriz (4x4) es la informacion/imágenes que ingresan hacia la red neuronal. La segunda matriz (3x3) son los pesos o un filtro de la red neuronal.
Ambas matrices son muy grandes (de varias dimensiones). Y su multiplicacion se repite varias veces, ya que los pesos se actualizan por cada iteración. Entonces el procesamiento de cálculo es muy elevado/pesado.

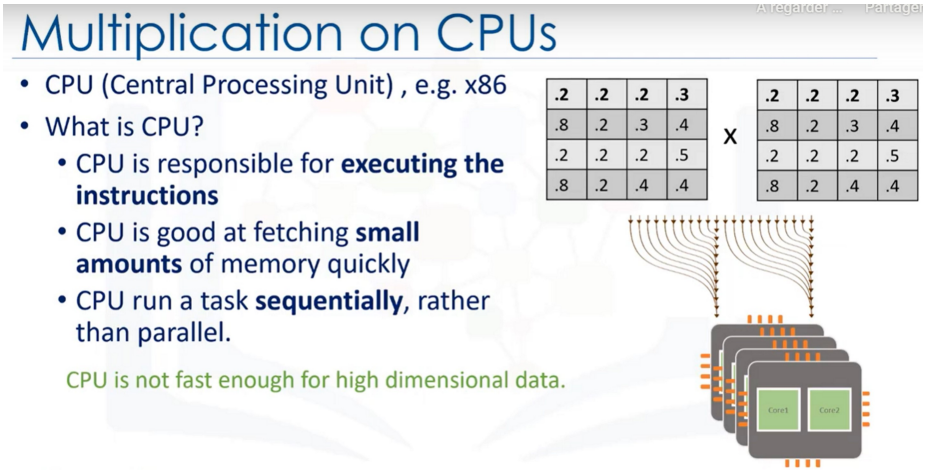


Una red CNN de 16 capas tendrá aproximadamente unos 140 millones de pesos y biases. Entonces al

pasar 1 imagen por la red neuronal, se realizar160n muchísimas multiplicaciones. Y muchísimas más al pasar el resto de imágenes de nuestra base de datos. Además de que para que la red neuronal aprenda adecuadamente tendrán que haber muchas iteraciones.

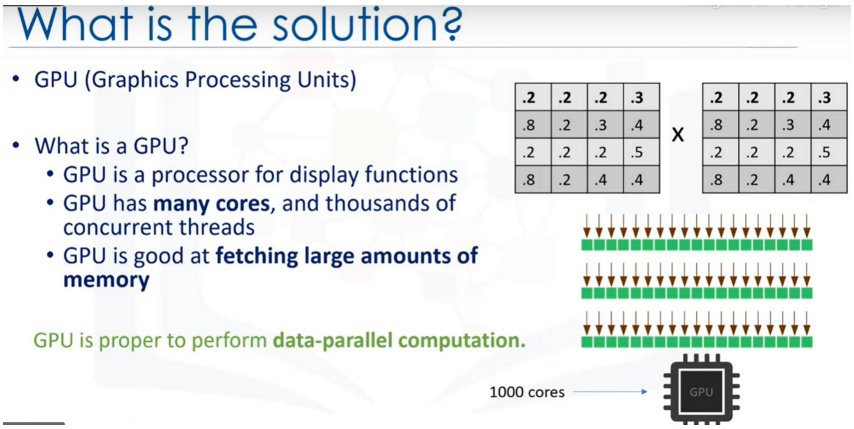
Para acelerar este proceso se usa el paralelismo. Este consiste en efectuar todas estas multiplicaciones en simultáneo para luego integrar/juntar los resultados. El GPU hace esto. Con un CPU no es posible porque éste es secuencial (ejecuta una operación después de la otra).

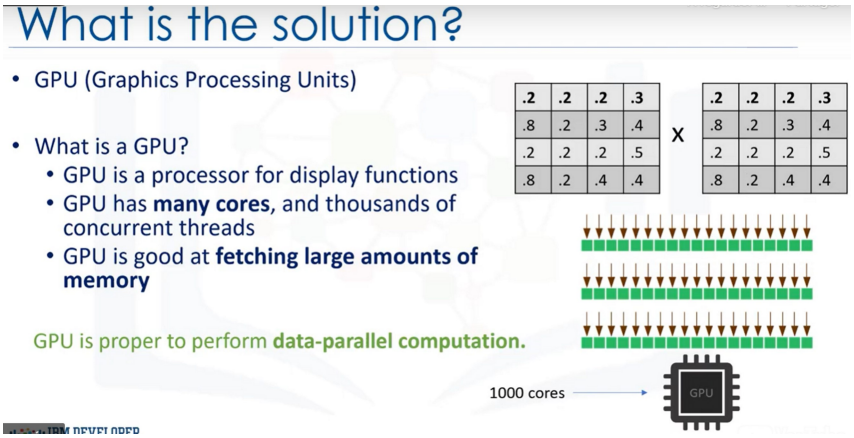
ACCELERATION WITH CPUs VS GPUs



CPU (Central Processing Unit)
Son procesadores secuenciales (Ex: x86 de Intel). Pueden efectuar operaciones en paralelo, pero su número es limitado porque el CPU es limitado en número de cores (Ej. 1 CPU = 7 cores = 7 operaciones en paralelo).

<u>Por qué no son buenos para el Deep Learning</u>?
En DL, las instrucciones que se tienen que ejecutar son multiplicaciones (grandes cantidades de matrices).
El CPU es bueno para recuperar pequenas cantidades de memoria rapidamente, pero NO para grandes cantidades de datos como las matrices en DL. Por lo tanto no es lo suficientemente rápido para trabajar con datos de varias dimensiones.





GPU (Graphics Processing Units)
Procesadores, originalmente, disenados para renderizar imágenes, animaciones y video para las PC. Pero también sirven para efectuar operaciones en paralelo de grandes catidades de datos (ej. multiplicaciones de matrices), ya que 1 solo puede tener varios cores (>1000).

Qué operaciones enviarle?
Cuando una fase de un proceso abarca "hacer una operación matemática varias veces". Ej:

- Multiplicación de matrices
- Calcular la inversa de una matriz
- Cálculo de gradientes: éstos son calculados en varios GPUs (individualmente) y promediados en el CPU

Cuando NO usarlo?
Los GPUs no tienen acceso al resto de periféricos de la PC (excepto al monitor). Debido a esto, para ejecutar un comando en el GPU es necesario:

- Primero, copiar toda la información relevante en el GPU
- Segundo, hacer la operacón
- Tercero, copiar el resultado de la operación a la memoria principal

Con TensorFlow es posible hacer todo esto de manera "transparente". Sin embargo, el GPU tiene prioridad sobre el CPU para el cálculo. Es decir, cuando hay CPU y GPU en la PC y hay operaciones que pueden efectuarse en cualquier tipo de procesador, el cálculo de estas operaciones será asignado al GPU.

Por qué son excelentes para el DL?
Una red neuronal necesita recuperar matrices de varias dimensiones (las imágenes de entrada) de la memoria principal.

- El GPU está optimizado para recuperar grandes cantidades de memoria.

Un red DL es una matriz de pesos en el que el "dot product" entre éstos y las imágenes de entrada tiene que hacerse varias veces.

- El GPU es muy bueno para hacer operaciones en paralelo: "donde el mismo código se ejecuta en diferentes secciones de la misma matriz." Entonces el tiempo de cálculo será poco.

Como usar un sistema multi-GPU para el entrenamiento con TensorFLow?

1. Guardar los parámetros del modelo en el CPU. Ex: pesos, bias
2. Hacer una copia del modelo y enviarla a cada GPU. Esto incluye: conv2d, pre-activación, relu, etc
3. Dividir los datos en grandes lotes por cada GPU y alimentar con un lote único a cada copia del modelo

4. Calcular la inferencia en cada GPU
5. Calcular los gradientes en cada GPU
6. Esperar a que todos los GPUs terminen de procesar sus lotes
7. Actualizar sincrónicamente los parámetros del modelo, en el CPU

Watson Studio setup --No es gratis!
https://github.com/IBM/skillsnetwork/wiki/Watson-Studio-Setup