PARCIAL III

COMPUTACIÓN BLANDA

DANIEL DIAZ GIRALDO

DANIEL FELIPE MARIN

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PEREIRA, RISARALDA

Framework usado:

- Tensor flow http://www.tensorflow.org/

Dataset Usado:

- CIFAR10 https://www.cs.toronto.edu/~kriz/cifar.html
- MNIST http://yann.lecun.com/exdb/mnist/

Código base Aymeric Damien:

- https://github.com/aymericdamien/TensorFlow-Examples/

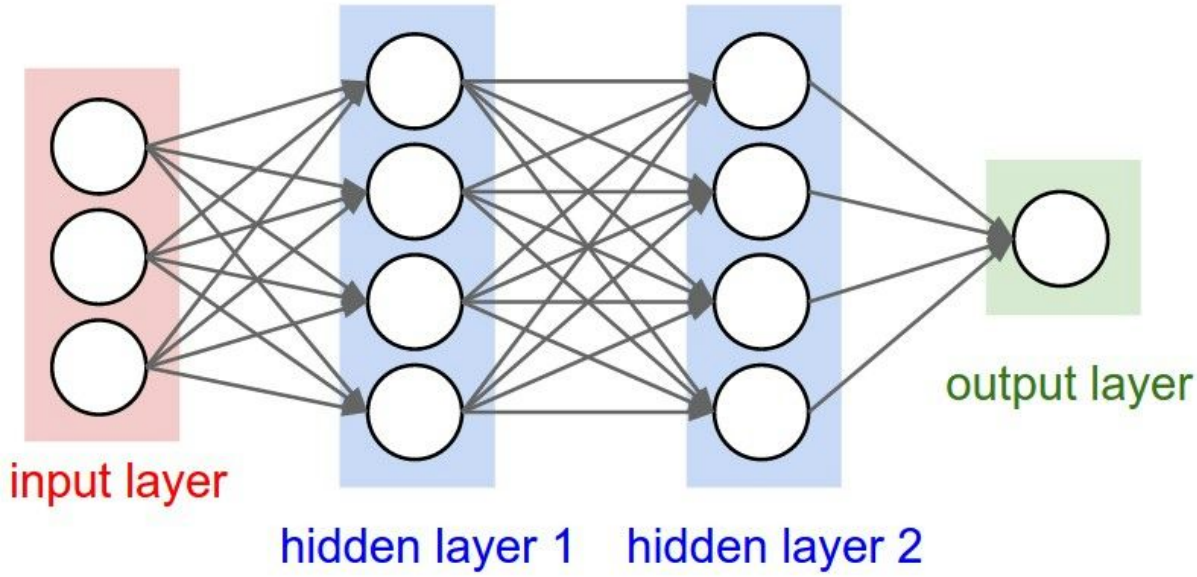
Metodología:

1. Carga de datos
    a. Carga de dataset en el entorno de tensor flow (python como front-end y parte del backend).
2. Elaboración del modelo funcional:
    a. Adecuación de una red neuronal para la clasificación de 10 clases.
    b. Técnicas usadas en el framework.
    c. Verificación con dataset de pruebas.
3. Toma de datos y evaluacion conceptual.
    a. Resultados.

Pasos realizados:

Algoritmo de multi-clasificación usando una red neuronal.

- Esquema utilizado para algunas pruebas:



Input layer: X[m x n] : Corresponde a nuestros datos de entrada, con la cual la red neuronal será entrenada para clasificar.
Hidden layer: Correspondiente a las funciones que harán la clasificación de la red, el tamaño de neuronas está limitado por las pruebas.
Output layer: Neuronas que contendrá el criterio de clasificación, gracias al proceso de las neuronas anteriores. Tamaño de 10 neuronas para ambos algoritmos.

- Para el MNIST: Sustracción correcta del dataset de imágenes, el cual comprende una colección de 60 mil imágenes comprendidas en 50 mil para entrenamiento del modelo y 10 mil para evaluarlo. Las categorías de clasificación están comprendidas entre los números desde el 0 hasta el 9. Cada imagen posee un tamaño de 28x28 en un solo canal de escala de grises.

- Para el CIFAR10 : Sustracción correcta del dataset de imágenes, el cual comprendía una colección de 60 mil imágenes comprendidas en 50 mil imágenes de entrenamiento y 10 mil imágenes de testing , todo este dataset comprendido entre 10 posibles categorías:
    - airplane
    - automobile

- bird
- car
- deer
- dog
- frog
- horse
- ship
- truck

Cada imagen tenía el tamaño de 32x32 conjunto a los 3 canales RGB, lo que generó una matriz del tamaño de 3072 para cada imagen. los labels están en formato correspondiente al orden de la lista (ejemplo: si es un caballo su valor correspondiente en el label es 8)

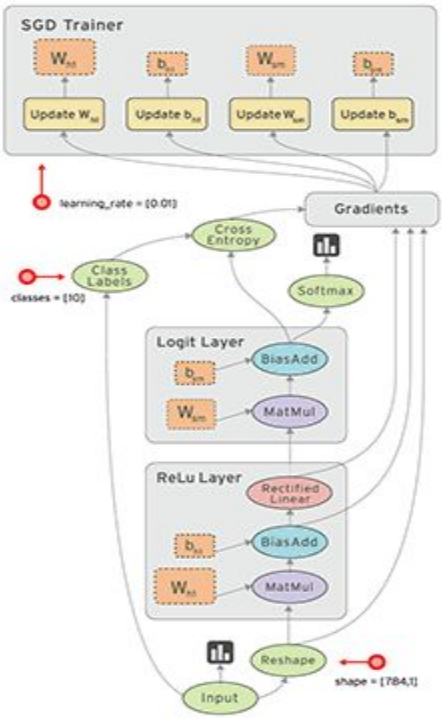
- Luego de tener los datos con los cuales trabajará la red, se procede a re-codificar los labels para tener un vector del tamaño igual a las neuronas de salida, esto es debido al proceso de clasificación binaria y al dominio de la función de activación (función sigmoidal).

Tensor Flow

TensorFlow™ es una librería de código abierto para la computación de cálculo numérico usando “data flow graphs”. los nodos en el grafo representan operaciones matemáticas, mientras que las aristas representan “multidimensional data arrays (tensors) ” comunicados entre ellos. La arquitectura flexible le permite implementar la computación a una o más CPU o GPU en un ordenador de escritorio, servidor o dispositivo móvil con una sola API.

¿Qué es un Data Flow Graph?

Los diagramas de flujo de datos (Data Flow Graph) describen el cálculo matemático con un grafo dirigido de nodos y bordes. Los nodos suelen aplicar operaciones matemáticas, pero también puede representar a los puntos finales para alimentarse en los datos, eliminar resultados, o leer / escribir variables persistentes. Los bordes describen las relaciones de entrada / salida entre los nodos. Estos bordes datos llevan conjuntos de datos multidimensionales, o tensores. El flujo de los tensores a través de la gráfica es donde TensorFlow recibe su nombre. Los nodos se asignan a los dispositivos de cómputo y ejecutar de forma asincrónica y en paralelo una vez que todos los tensores en sus bordes entrantes que se disponga.



tf.nn.relu : Rectificador lineal $f(x) = \max(0, x)$, biológicamente más viable que la regresión lineal y más práctica que la tangente hiperbólica. https://en.wikipedia.org/wiki/Rectifier_%28neural_networks%29#cite_note-glorot2011-1

tf.matmul : Multiplica la matriz por una matriz b, produciendo a * b.

tf.add: Devoluciones x + y elemento sabio.

tf.nn.softmax_cross_entropy_with_logits: Mide la probabilidad de error en las tareas de clasificación discreta en la que las clases son mutuamente excluyentes (cada entrada es exactamente una clase). Por ejemplo, cada imagen CIFAR-10 está marcado con una y sólo una etiqueta: una imagen puede ser un perro o un camión, pero no ambos.

tf.train.AdamOptimizer: Implementation is based on:http://arxiv.org/pdf/1412.6980v7.pdf

Características del ordenador, Cifar10:

```
hpc@arwen:/$ lscpu
Architecture:            x86_64
CPU op-mode(s):          32-bit, 64-bit
Byte Order:              Little Endian
CPU(s):                  8
On-line CPU(s) list:     0-7
Thread(s) per core:      2
Core(s) per socket:      4
Socket(s):               1
NUMA node(s):            1
Vendor ID:               GenuineIntel
CPU family:              6
Model:                   58
Stepping:                9
CPU MHz:                 1600.000
BogoMIPS:                7020.33
Virtualization:          VT-x
L1d cache:               32K
L1i cache:               32K
L2 cache:                256K
L3 cache:                8192K
NUMA node0 CPU(s):       0-7
```

```
hpc@arwen:/$ lsblk -d -o name,rota
NAME ROTA
sda     0
hpc@arwen:/$
```

ssd

Características del ordenador, Mnist:

```
Arquitectura:          x86_64
modo(s) de operación de las CPUs:32-bit, 64-bit
Orden de los bytes:    Little Endian
CPU(s):                4
Lista de la(s) CPU(s) en línea:0-3
Hilo(s) de procesamiento por núcleo:2
Núcleo(s) por «socket»:2
«Socket(s)»          1
Modo(s) NUMA:          1
ID de fabricante:      GenuineIntel
Familia de CPU:        6
Modelo:                42
Nombre del modelo:     Intel(R) Core(TM) i3-2120 CPU @ 3.30GHz
Revisión:              7
CPU MHz:               2080.031
CPU MHz máx.:          3300,0000
CPU MHz mín.:          1600,0000
BogoMIPS:              6587.28
Virtualización:        VT-x
Caché L1d:             32K
Caché L1i:             32K
Caché L2:              256K
Caché L3:              3072K
CPU(s) del nodo NUMA 0:0-3
```

Resultados

MNIST

Capas ocultas: 2

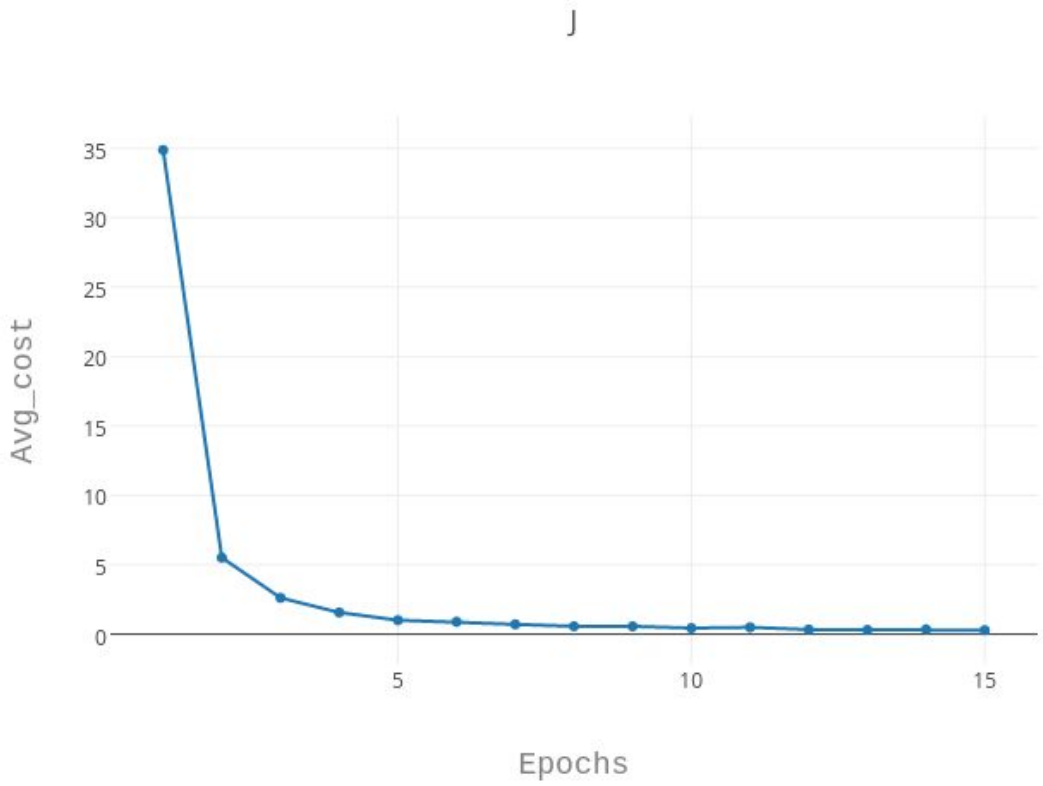
learning_rate	training_epochs	batch_size	n_hidden_1	n_hidden_2	accuracy	time(seg)
0.001	15	100	256	256	0.9483	101.11609602
0.1	15	100	256	256	0.2026	121.44608616
0.01	**15**	**100**	**256**	**256**	**0.965**	**106.66971206665039**
0.01	15	50	256	256	0.9422	168.1938169
0.001	8	50	256	256	0.9441	75.655388832
0.001	8	50	784	256	0.9541	194.08952498

Capas ocultas: 3

learning_rate	training_epochs	batch_size	n_hidden_1	n_hidden_2	n_hidden_3	accuracy	time
0.001	15	100	256	256	256	0.9403	116.313097954
0.001	15	100	300	200	300	0.9431	138.33596015
0.1	15	100	300	200	300	0.0985	169.41159415245056
0.01	15	100	300	200	300	0.959	144.07552409172058

J:

0.01	15	100	256	256	0.969	113.61912202

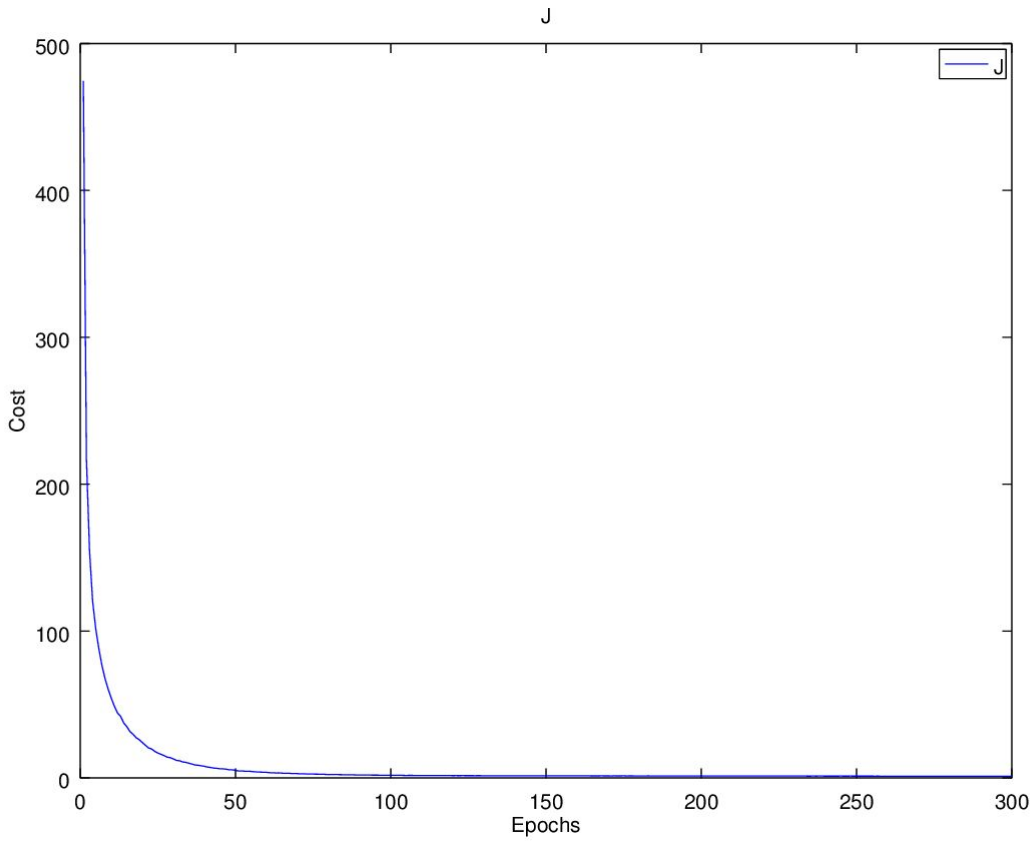


CIFAR-10

learning_rate	training_epochs	batch_size	n_hidden_1	n_hidden_2	accuracy	time(seg)
0.001	1000	100	256	256	0.3816	7449.4739561
0.001	500	80	256	256	0.3723	11345.222903
0.001	300	100	50	28	0.1	1626.5503900
0.001	300	100	150	28	0.1	1886.0486681461334
0.001	**300**	**150**	**250**	**150**	**0.4157**	**1850.2415947914124**

J:

0.001	300	150	250	150	0.4157	1850.2415947914124



Conclusiones :

1. Tensorflow utiliza todos los núcleos de la CPU. La precisión de 0.41 obtenida en tensorflow se logra aproximadamente en la mitad del tiempo que la red neuronal con octave.
2. El entrenamiento de Mnist, solo tarda en promedio dos minutos.
3. El uso de batches optimiza el modelo de la red neuronal, al permitir no tener que usar todo el dataset de entrenamiento en cada iteración.
4. Organizar el dataset para entregarlos a tensorflow es poco intuitivo.