

PARCIAL FINAL
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:

- MNIST <http://yann.lecun.com/exdb/mnist/>

Código base Aymeric Damien:

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

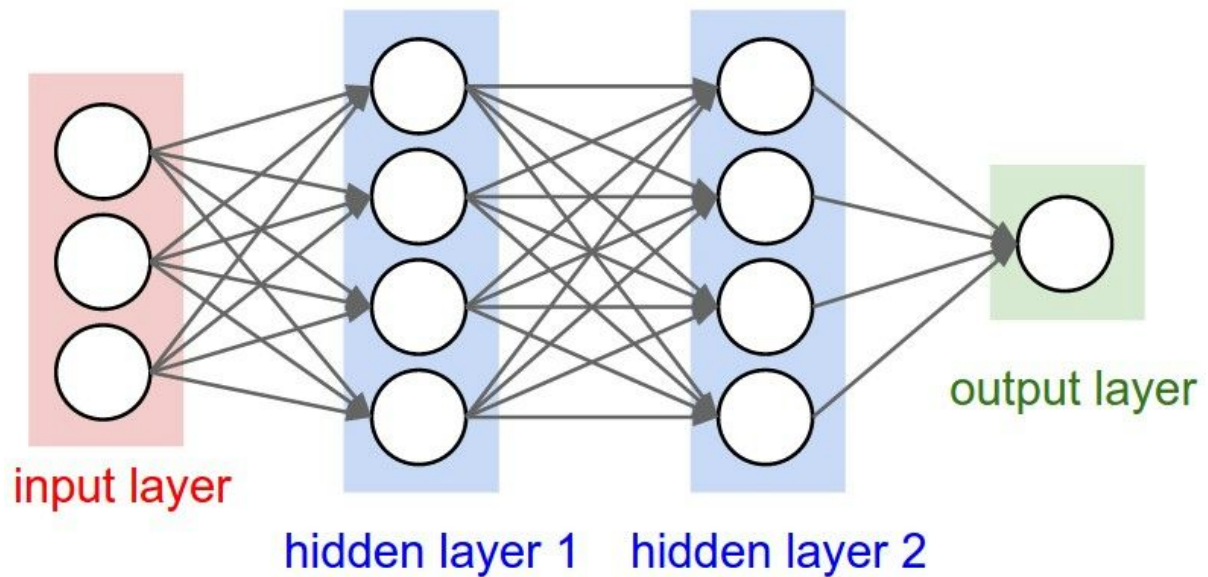
Metodología:

1. Entrenamiento de la red
 - a. Carga de dataset en el entorno de tensor flow (python como front-end y parte del backend).
 - b. Entrenamiento de red convolucional usando tensorflow.
 - c. Guardar los pesos de la red.
2. Elaboración aplicativo web en node.js para pruebas:
 - a. Creación de scripts para el manejo las redes neuronales normal y convolucional.
 - b. Carga de imagen a partir de un canvas.
 - c. Verificación de la imagen a partir de los dos modelos entrenados.
3. Toma de datos y evaluacion conceptual.
 - a. Resultados.

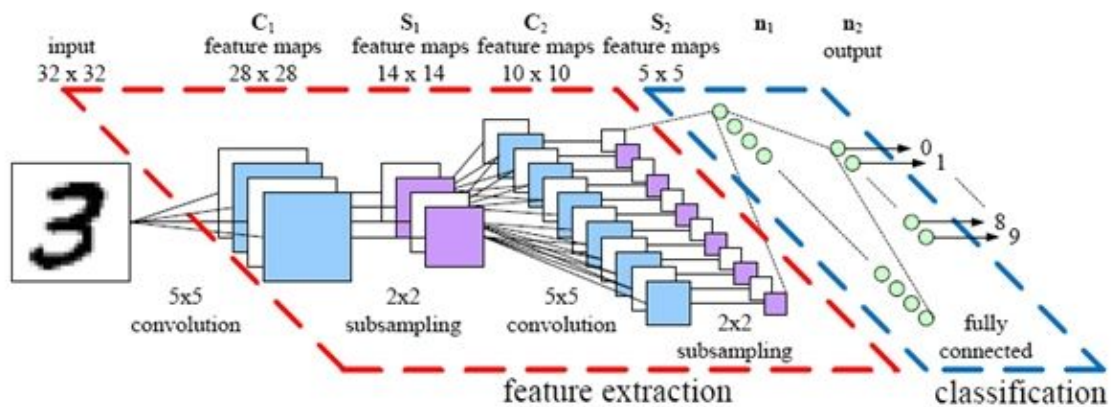
Pasos realizados:

Algoritmo de multi-clasificación usando una red neuronal.

- Esquema utilizado para algunas pruebas:



Red neuronal tradicional



Red neuronal convolucional.

Input layer: $X[m \times n]$: 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.

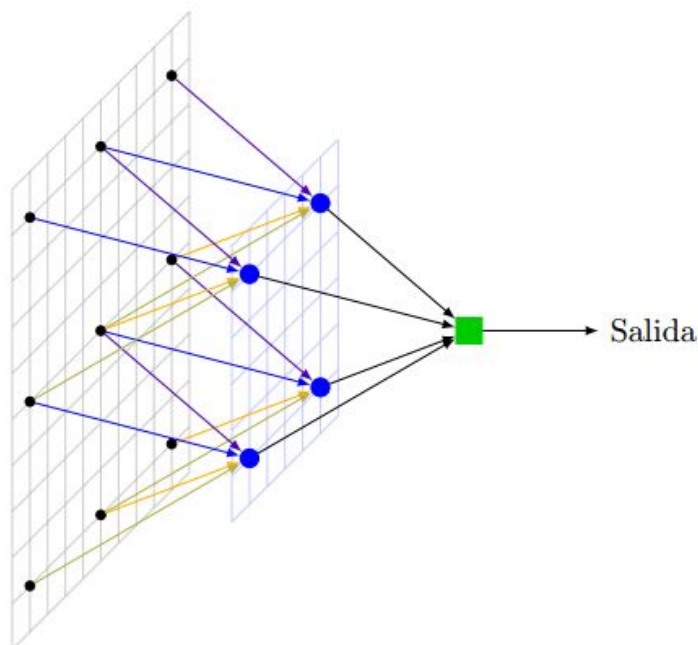
Output layer: Neuronas que contendrá el criterio de clasificación, gracias al proceso de las neuronas anteriores. Tamaño de 10 neuronas para ambos algoritmos.

- 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.

Red Neuronal Convolucional

En una red neuronal normal, se tiene una neurona para cada píxel de una imagen y después se tienen varias capas con varias neuronas, todas conectadas entre sí, para tratar de encontrar un número en una foto, por ejemplo. El problema es que no es demasiado efectivo, por ejemplo con imágenes de 1920x1080 píxeles.

La idea de las redes convolucionales es tratar de buscar características locales en pequeños grupos de entradas (en el caso de las imágenes, de píxeles), como puedan ser bordes o colores más o menos homogéneos. Se buscan características no en toda la imagen sino sólo en pequeñas regiones. Además, se busca siempre detectar la misma característica en todos los grupos, por lo que se puede repetir esa estructura y reducir los ajustes que tenemos que hacer.



Para llevar a cabo esta idea, se tiene un mismo grupo de neuronas por cada grupo de entradas (por ejemplo, un cuadrado de 3x3 píxeles en una imagen o una secuencia de 4 mediciones en un archivo de sonido). La idea es que todos los elementos que estén en la

capa (llamada capa de convolución) tienen los mismos pesos por cada entrada, y se reduce considerablemente el número de parámetros. Si metemos más capas, la red neuronal podrá descubrir más y más complejas características de la imagen: se puede empezar por colores o bordes orientados y acabar con capas que se activan con formas circulares o cuadradas, por poner un ejemplo.

Después de las capas de convolución se suele poner otra red neuronal "tradicional", que ahora tendrá más fácil el trabajo: no tiene que valorar cada píxel por separado sino que mira a un conjunto de características de alto nivel de la imagen. Ya no se trata de decidir si la imagen es un coche sabiendo que el píxel 1208 es amarillo y el 1209 es verde, sino quizás sabiendo que hay una forma rectangular en la imagen con dos formas circulares en la parte inferior. De nuevo, se trata de extraer la información "oculta" en la entrada para tratar de encontrar qué es lo que define esos datos.

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.

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
```

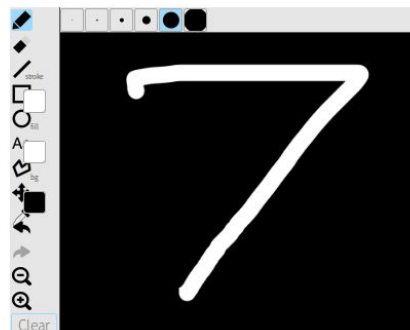
Aplicativo Web: Node.js

Dibuje un dígito del 0 hasta el 9

Es obligatorio seleccionar el "stroke" de color blanco también el "bg" de color negro. La predicción se hará en base a valores predefinidos, luego de dibujar su dígito seleccione predecir para iniciar el procesamiento de su imagen.

PREDECIR

VERIFICAR



Predicción

Según el algoritmo el número es:

Convolutional Network: 7 Normal Network: 5

VOLVER A INTENTAR

RESULTADOS







Convolutional:





N	Testing Accuracy	Learning rate	Training iters	batch_size	Tiempo empleado
1	0.976562	0.01	300000	128	1224.7647259235382
2	1.0	0.01	300000	256	1270.804370880127
3	0.832031	0.01	20000	256	7.66520118713379

Normal:












Testing accuracy	Learning rate	Training iters	n_hidden1	n_hidden2	batch_size	Tiempo empleado
0.965	0.01	9000	256	256	100	106.66971206665039

Convolutional #1 0.97

Imagen	Convolutional Network	Normal Network
	0	0
	2	2
	2	2
	3	3
	4	4
	5	5

	5	6
	3	3
	8	8
	2	3
Accuracy	6/10	7/10

Convolutional #2 1.0

Imagen	Convolutional Network	Normal Network
	0	0
	1	2
	2	2
	3	3
	4	4
	5	5
	6	6
 , 	3 , 7	3
	8	8
	9	3
Accuracy	9/10 , 10/10	7/10

Conclusiones :

1. La red neuronal convolucional con una precisión de 1.0 tiene una cantidad de aciertos mayores que la red neuronal normal.
2. 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.
3. La predicción del número 7 es la menos eficiente en ambas redes.