DEEP LEARNNING (APRENDIZAJE PROFUNDO) CON TORCH 7

Deep Learnning (Aprendizaje Profundo) con Torch 7

Israel Rivera Zárate

Instituto Politécnico Nacional-CIDETEC

irivera@ipn.mx

Miguel Hernández Bolaños

Instituto Politécnico Nacional-CIDETEC

mbolanos@ipn.mx

Patricia Pérez Romero

Instituto Politécnico Nacional-CIDETEC

promerop@ipn.mx

I. Introducción

Deep learning es por un lado un subconjunto del machine learning (aprendizaje automático) que a su vez podemos definir como: "la capacidad que tienen las computadoras de actuar o aprender sin ser explícitamente programadas para ello". Se trata de un campo de estudio entre la informática, las matemáticas, la estadística y la ingeniería, en la que se aborda la teoría del reconocimiento de patrones y de cómo podemos construir algoritmos que pueden aprender una determinada tarea directamente a partir de los datos sin necesidad de llevar a cabo una selección previa de las características que existen en los datos mismos [1].

El procedimiento habitual de las técnicas clásicas de aprendizaje máquina se basa en entrenar un modelo utilizando un conjunto de datos de entrenamiento y después utilizar ese modelo para hacer predicciones sobre un nuevo conjunto de datos que el algoritmo no ha visto previamente. Ver figura 1.

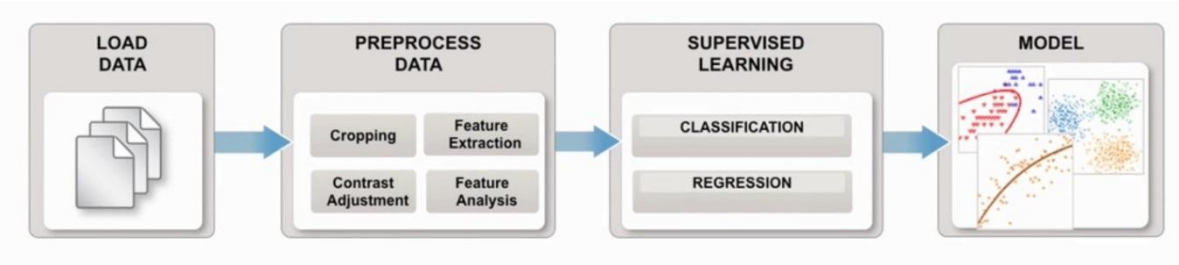


Figura 1. Generación de un modelo en el aprendizaje máquina.

En la fase de entrenamiento: Se carga un conjunto de datos, se realiza un pre procesamiento de los datos eligiendo las características relevantes y después se realiza una técnica de aprendizaje supervisado como son la clasificación o regresión para construir un modelo.

Construido el modelo, en la siguiente fase se desea hacer una predicción: donde a partir de un nuevo conjunto de datos se utiliza el mismo procesamiento empleado en la fase de entrenamiento para la extracción de características de los datos así como el modelo determinado [2]. Ver figura 2.

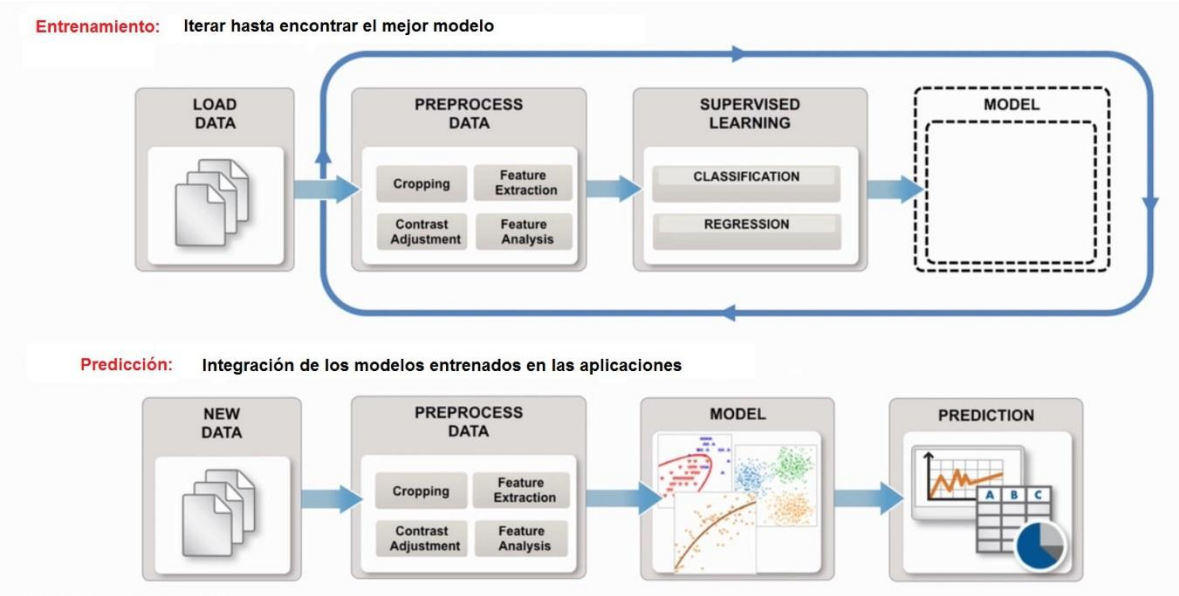


Figura 2. Etapas de entrenamiento y predicción del aprendizaje máquina.

En deep learning el entrenamiento: Parte de emplear imágenes etiquetadas por lo que se le brinda al algoritmo las características contenidas en esas imágenes y que las relacionan con tales etiquetas. El objetivo es aprender a partir de los datos con lo que se evita influir en las características y es por ello que en ocasiones se conoce también a este tipo de aprendizaje como aprendizaje de principio a fin o "end to end learning".Ver figura 3.

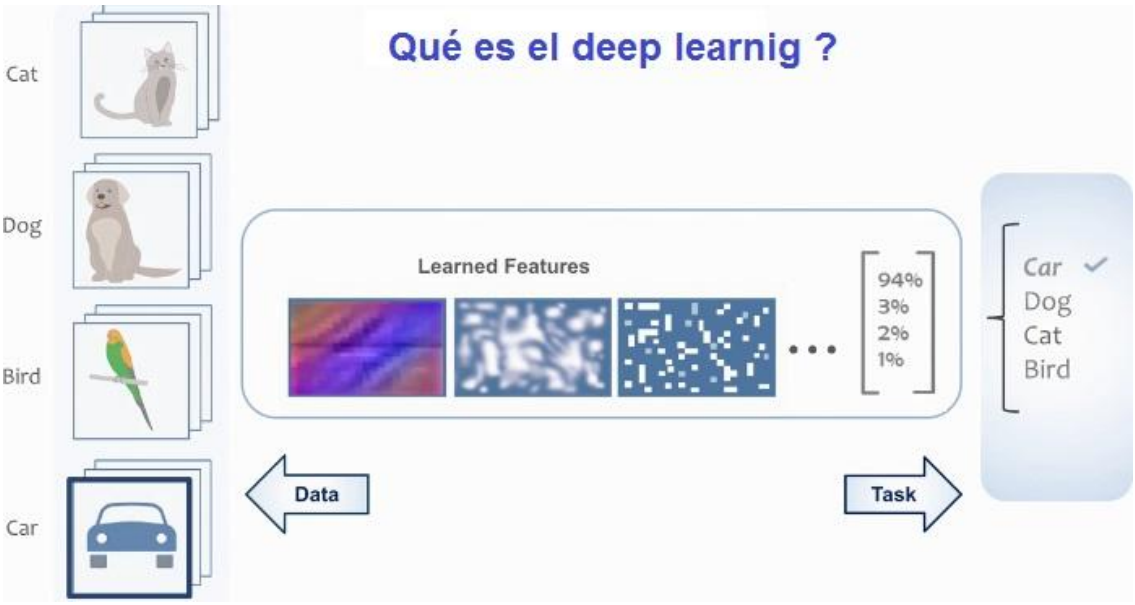


Figura 3. Características del deep learning.

Existe un conjunto de factores que facilitan llevar a cabo el aprendizaje profundo. El primero a considerar es el uso de las unidades deprocesamiento gráfico o "GPU's" que permiten entrenar los modelos mucho más rápido del orden de 60X. El segundo, que hoy en día ya existen conjuntos de datos ya etiquetados, lo cual ha sido posible gracias a la colaboración abierta entre investigadores. El último es el hecho de que el estado del arte de modelos de Deep learning está disponible para ser utilizados por cualquier persona. Ver figura 4.

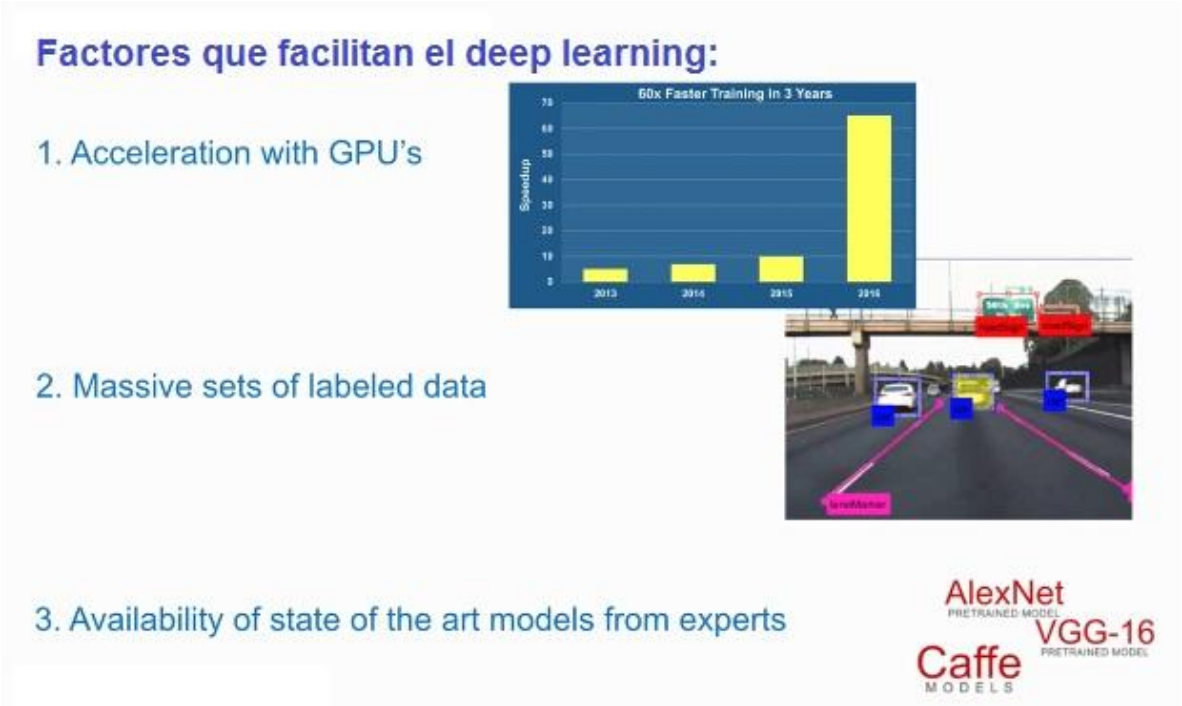


Figura 4. Factores que facilitan el deep learning.

En el aprendizaje profundo las redes multicapa cuentan con una función de transferencia la cual se adapta de acuerdo al problema por resolver. Sin embargo, uno de los principales atributos del aprendizaje que las hacen más sobresalientes que usar simplemente redes neuronales convencionales, son dos puntos muy esenciales el primero de ellos es el uso de los kernel que ayudan a modificar los datos para permitir crear características que los hagan únicos de los demás; el segundo de los puntos que caracterizan al aprendizaje profundo es el uso del pooling [3]. El pooling es una forma de simplificar una región permitiendo reducir el tamaño de la misma; dependiendo del pooling que se aplique puede ser el máximo, el mínimo o el promedio. Al salir del pooling se obtiene la representación que se asignó, provocando una reducción en el tamaño de los datos para la siguiente capa, entonces al pasar por una cantidad de capas, se reduce esa dimensión produciendo una compresión en los datos. Ver figura 5.

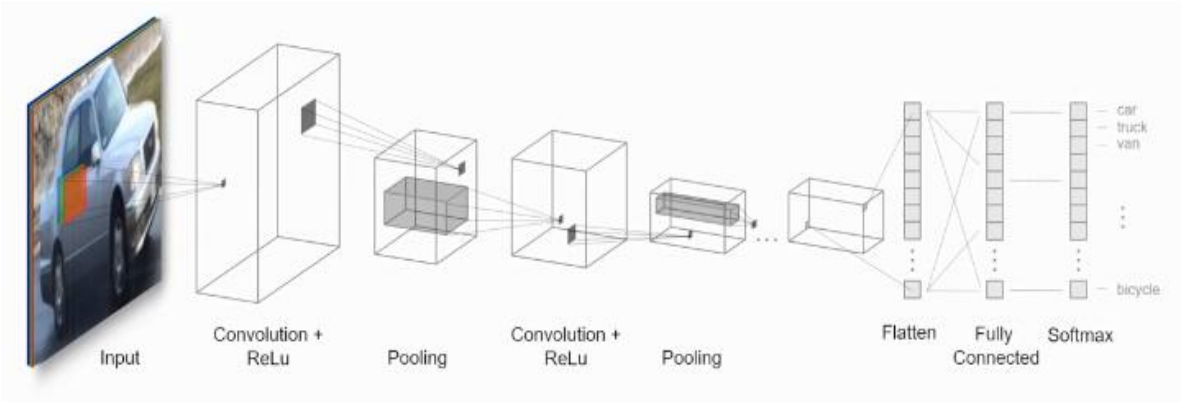


Figura 5. Etapas de procesamiento del deep learning.

II. TORCH 7: Herramienta de aprendizaje profundo

Torch 7 fue desarrollado por la universidad de Toronto en Canadá, se ha desarrollado bajo el lenguaje LUA, una de las características principales de LUA es aprovechar las arquitecturas multiprocesador que actualmente tienen los dispositivos computacionales como son los dispositivo móviles hasta grandes clústeres de supercomputadoras. Torch7 realiza las operaciones de dos tipos: la primera es utilizar el poder de la unidad central de procesamiento (CPU), la segunda es hacer uso de las unidades de procesamiento gráfico (GPU). Utilizar la GPU para tareas de aprendizaje profundo es más eficiente que hacer uso de la CPU. Torch7 utiliza el sistema operativo Ubuntu; es capaz de ser ejecutado tal como su entrenamiento además la etapa de evaluación en un dispositivo móvil como puede ser Android o IOS. También tenemos a Tensor Flow, esta herramienta fue desarrollada por Google para poder realizar tareas de aprendizaje profundo para sus herramientas; Tensor Flow utiliza el lenguaje Python para poder desempeñar sus tareas, al igual que Torch7, Tensor Flow aprovecha la GPU para poder desempeñar tareas que requieren mayor poder computacional, donde claramente al utilizar clusteres de GPU tiene mejor rendimiento que usar solamente la CPU. Ambos tanto Tensor Flow como Torch7 son herramientas con mucho potencial debido a que utilizan una forma de interactuar con los datos que son llamados Tensores, los cuales no requieren dimensionalidad y definición de tipos de objetos [4].

a). Primeros pasos

Como primer paso debemos de realizar la respectiva instalación, primero es entrar al sitio oficial de Torch7, la página nos ofrece la descripción de los requerimientos que se necesita para poder hacer uso del entorno, además es recomendable realizar la instalación con un equipo CUDA para aprovechar el máximo potencial.

Una vez que se realizó la instalación de Torch7, se realiza la verificación de la respectiva ejecutando el comando en una terminal de Ubuntu: **th**, al realizar el comando se desplegará la palabra Torch7 en la terminal con eso garantizamos que la instalación se realizó de manera exitosa. Ahora tenemos listo nuestro entorno de trabajo preparado para comenzar a ejecutar programas de Torch7.

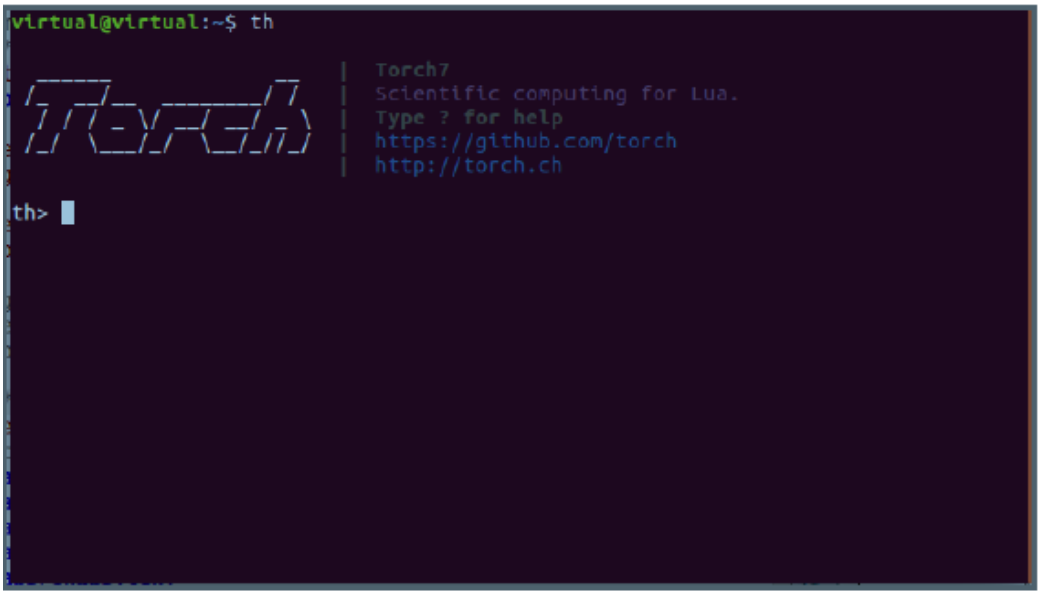


Figura 6. Entorno de desarrollo de la herramienta Torch.

El primer programa en Torch7 es el “hola mundo” de la página oficial de Torch7, se muestra en el siguiente código.

```
torch.manualSeed (1234)
N = 5
A = torch . rand (N, N)
A = A * A : t ( )
A: add ( 0 . 0 0 1 , torch . eye (N) )
b = torch . rand (N)
function J ( x )
return 0.5 * x : dot (A * x ) - b : dot ( x )
end
```

print (J (torch . rand (N)))

Como primer paso se debe crear un archivo asignándole un nombre con la extensión **.lua**; nosotros creamos el archivo HolaMundo cuya extensión debe de ser del archivo LUA, debido a que Torch7 hace uso de este entorno de ejecución, LUA no es un compilador, sino que ejecuta cada línea en tiempo de ejecución. En el siguiente ejemplo se describirá una red neuronal multicapa, especialmente se busca resolver el problema de la XOR utilizando una red neuronal. El problema consiste en encontrar los pesos necesarios para poder realizar una XOR, cuya tabla de verdad es la siguiente.

Tabla 1. Tabla funcional operación XOR.

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

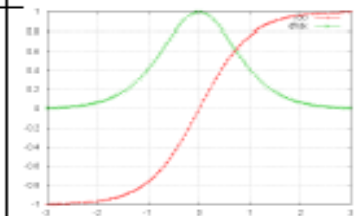
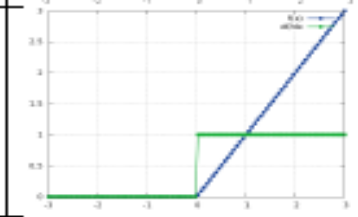
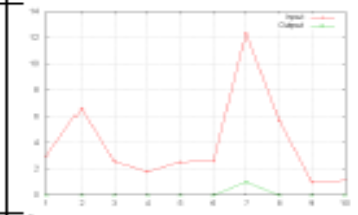
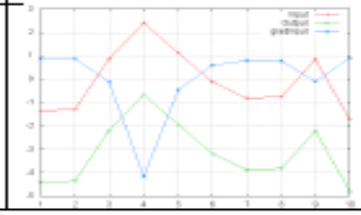
Se requiere utilizar el paquete que guarda todas las llamadas a funciones para poder utilizarse en las redes neuronales que es: **nn**. En este empaquetado se define el tipo de modelo que se implementará debido a que Torch7 tiene 3 diferentes estructuras de modelos: secuencial, concatenado y paralelo.

El primero, el modelo secuencial como lo indica su nombre los pasos se deben de seguir de forma secuencial, donde la salida de la primer capa es la entrada de la siguiente capa hasta que ya no se tengan capas a realizarse; el segundo modelo es el modelo concatenado, el cual se asimila como al tener dos modelos con características diferentes pero al final de cada uno de los modelos el resultado es unido, por lo que un modelo A puede tener un número de entrada a con capas diferentes al modelo B, al igual que sus salidas, pueden tener cada modelo diferente número de capas, así como de diferentes funciones de transferencia, pero al final ambos modelos el resultado es unido; por último, se tiene el modelo paralelo, como su nombre lo indica un número indeterminado de modelos se ejecutan totalmente de forma paralela lo que nos indica que cada modelo es independiente entre ellos, así como su salida. Para este ejemplo se utiliza el modelo secuencial.

Debemos de definir las formas de clasificar nuestra red neuronal, vamos a definir el número de nuestras entradas que será de dos, el número de salidas para este ejemplo es de una, también es una red multicapa. Se define el número de neuronas ocultas para este ejemplo, definiendo 20 neuronas, cabe mencionar que el número de neuronas no está definido bajo un criterio, debido a que el ejemplo puede funcionar mejor o menor rendimiento si incrementamos o disminuimos el número de neuronas en la capa oculta.

Debido a que tenemos un conjunto con dos clases de entrada, cuyos valores son un número 0 o un número 1, así como de una salida que es 1 o 0, se ha comprobado que utilizar una función de transferencia lineal es suficiente para resolver el problema de la XOR utilizando redes neuronales, para definir la función de transferencia, ésta se encuentra definida dentro del paquete nn, por lo que al consultar la documentación se requiere una entrada así como de una salida, debido a que es la primer capa debemos de especificar como 2 entradas, se tiene una capa de redes neuronales ocultas de 20 neuronas, posteriormente para simplificar aún más el entrenamiento se añade la función de transferencia que es la tangente hiperbólica; por último se busca clasificar la salida, añadiendo nuevamente una función de transferencia lineal, pero ahora la entrada es el número de neuronas de la capa oculta, para analizar la salida es la que se requiere, para este ejemplo es sólo una salida, ver tabla 2.

Tabla 2. Funciones de transferencia más empleadas.

Función de transferencia	Gráfica
Tanh	
ReLU	
Softmax	
LogSoftmax	

La función SoftMAx está definida como:

$$f_i(x) = \frac{e^{x_i - \max x_i\,(x_i)}}{e^{x_j - \max x_i\,(x_i)}}$$

Donde x es cada muestra de la imagen
La función Log SoftMAx está definida como:

$$f_i(x) = \log\left(\frac{1}{a * e^{x_i}}\right)$$

Donde x es cada muestra de la imagen.

Se cuenta con lo necesario para poder ejecutar el modelo de redes neuronales utilizando Torch7, pero a lo largo del estudio de las redes neuronales es necesario utilizar funciones para tomar criterios con los pesos para validar si son correctos, se debe apoyar con funciones que permitan hacer esta comparación, en Torch7 se tiene implementadas funciones que son conocidas como criterios de evaluación; en la documentación de Torch7 nos despliegan los diferentes criterios que nos ofrece, además es un software libre de implementar en cualquier criterio que sea de gran utilidad para nuestro desarrollo. Se sabe que el criterio más común para utilizar es el MSE (Mean Square Error), por lo que haremos uso de este criterio [5]. Para el criterio MSE su evaluación tiene un método que es llamado forward, que lo podemos traducir como la operación que realiza la diferencia de error entre la evaluación del modelo con el valor deseado. Debido a que se utiliza el back propagation para regresar toda la red de acuerdo a la entrada con respecto a la salida, posteriormente se aplica el cambio hacia atrás, se debe de definir la frecuencia de muestra para realizar el aprendizaje, todo esto último debe de ejecutarse en un ciclo for dependiendo del número de iteraciones que queremos comprobar. En este ejemplo realizaremos la ejecución con 10 evaluaciones de entrenamiento para validar su eficacia, posteriormente observemos con 100, 1000 y 5000 muestras de entrenamiento.

El ejemplo de una red neuronal multicapa nos permite entender el manejo de utilizar Torch7 para implementar tareas de máquinas de aprendizaje para poder explotar su potencial. Ahora es necesario realizar redes neuronales convolucionales, se debe realizar la manipulación de una imagen en Torch7, debido a que una imagen mantiene datos en una matriz. Ver tabla 3.

Tabla 3. Resultados del entrenamiento para una operación XOR.

A	B	10	100	1000	5000
0	0	0	0	0	0
0	1	0	0	0	1
1	0	0	0	0	1
1	1	0	0	0	0

La lectura de imágenes es en formato png y jpg porque son las imágenes que soporta Torch7, en caso de realizar la lectura de otro tipo de formato es necesario leer el manual del paquete image para saber si nuestro formato es soportado. Con base en la documentación que nos ofrece Torch7, se observa que al leer una imagen se almacena en un tensor, una vez almacenado en la variable la imagen tiene 3 canales que representan a los colores rojo, verde y azul; en inglés representa al conjunto de los colores RGB, por lo tanto, la imagen debe contener un plano para cada color mencionado. Además, otros datos representativos son las dimensiones de la misma imagen. En Torch7 se tiene el manejo de tensores, no es necesario definir las dimensiones del elemento que ocupará en la memoria. En la siguiente línea de código se describe la distribución de la composición de los valores que conforman a la imagen.

Imagen [{ { canal } ,{ ancho} , { a l t u r a } }]

Con esta información se conoce la distribución de los bytes que componen a la imagen de acuerdo a su posición de cada uno de los respectivos 3 canales que corresponden a una imagen a color. El saber la distribución nos permite crear los conjuntos de imágenes que se requieren para poder realizar el entrenamiento de una red neural de cualquier tipo.

****
Para poder definir la longitud que le corresponde a la variable que almacena todas las imágenes, se define con un cuarto valor que se le asigna la longitud de todas las imágenes almacenadas. La representación se asimila al almacenamiento de arreglos.

Imagen [{ { p o s i c i ó n } , { canal } , { ancho } , { a l t u r a } }]

Con esta información define las variables que se encargan de almacenar todas las imágenes que componen a nuestro entrenamiento, así como de nuestro conjunto de pruebas para saber si nuestro entrenamiento es correcto.

b). Redes Convolucionales

Se ha definido la aplicación para realizar la implementación de una red neuronal multicapa. Ahora, se describirá el desarrollo de una red neuronal convolucional utilizando el aprendizaje profundo. El aprendizaje profundo hace uso de dos características principalmente que lo diferencia de una red neuronal multicapa, una diferencia es usar convoluciones y la segunda es utilizar funciones de agrupación (pooling), lo que permite realizar una reducción entre capas, lo que Refleja como si los datos analizados son reducidos a través de las capas hasta que el final del modelo, solamente se muestra el conjunto de datos que hace diferentes entre diferentes clases, esto último depende de la aplicación de usar aprendizaje profundo [6].

Para este ejemplo se utiliza el método **SpatialConvolution**, la cual pertenece al paquete de redes neuronales de Torch7. Para utilizar la convolución se debe indicar al menos cuatro

atributos que son necesarios para la convolución que son: el número de entradas, el número de características a obtener, así como del tamaño del kernel que se toma para obtener las características. El kernel es la vecindad o el conjunto de pixeles que se toman por cada pixel que representa a la imagen a convolucionar. Las características son generadas a partir de aplicar variaciones en los análisis que conforman al kernel. El objetivo del kernel es determinar las limitaciones entre los rasgos que diferencian a cada una de las imágenes para poder separar los datos en regiones, lo principal es buscar las derivadas que permiten encontrar los rangos más representativos entre imágenes, las cuales permite definir zonas que limitan entre diferentes áreas, un kernel muy representativo es el kernel de Sobel [7] que se muestra en la tabla 4.

Tabla 4. Kernel Sobel.

	1	
1	-4	1
	1	

El kernel es conformado con una matriz de 3x3, el kernel es aplicado a cada uno de los pixeles, como la interacción entre pixeles es independiente entre ellos es posible explotar este comportamiento utilizando las tarjetas gráficas porque de acuerdo a las características del problema se acelera el procesamiento de los datos. Cuando se especifica el número de características lo que se genera son propuestas de variaciones de kernel para poder determinar un número de características que varíen con base a todo un conjunto de datos. Un kernel también es conocido como filtro, porque trata de agrupar o enfatizar más los datos. Una vez que se haya establecido la convolución para poder explotar más lo que es el aprendizaje profundo es hacer uso del pooling. El pooling es la parte de agrupación, se tienen diferentes tipos de pooling cuya características es que la agrupación se realizar de diferentes formas [8]. Un ejemplo es utilizar el maxpooling, donde se especifica que la agrupación de un grupo de pixeles se tomará el máximo. En la figura 4 se define de forma más gráfica en que consiste el maxpooling.

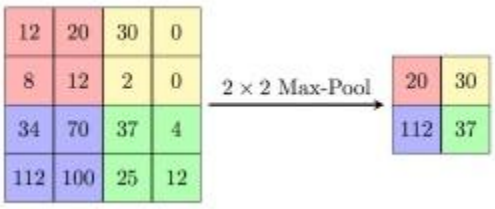


Figura 4. Representación gráfica del Maxpooling.

Además de la agrupación por máximos, también se tiene la agrupación por mínimos y por promedio de la vecindad, podemos usar este comportamiento para reducir los datos que pasan a la siguiente capa, en caso de que requiera realizar hasta un número de capas. La convolución

y la agrupación son las características más importantes al implementar un modelo utilizando el aprendizaje profundo. Al igual que las redes neuronales multicapa, se requiere un optimizador como el gradiente descendiente, además del back propagation para mantener la integración de las redes neuronales. Prácticamente, los cambios importantes son la implementación del modelo, debido a que también se requiere un entrenamiento para realizar ajustes, así como de un optimizador para poder acelerar la convergencia entre las diferentes funciones de transferencia. Ver figura 5.

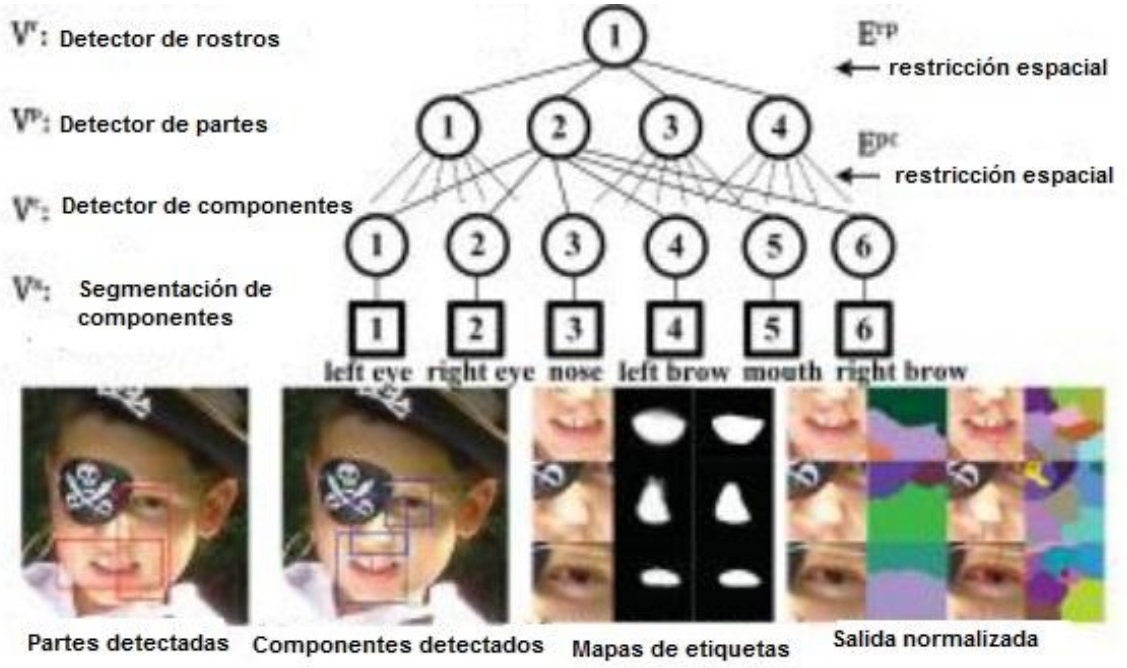


Figura 5. Funciones asociadas a las diferentes capas en una red de aprendizaje profundo.

III. Conclusiones

El presente artículo sentó las bases teóricas que brindan un panorama general en el tema del aprendizaje profundo entendiendo que forma parte de los esquemas de aprendizaje máquina más novedosos. Se señaló que el aprendizaje profundo comienza a integrar funciones psicológicas distintivas como son la memoria, el razonamiento, la atención, la motivación y las emociones y que sus aplicaciones son cada vez más demandadas por el mundo empresarial, algunos campos en los que se ha desarrollado son: los traductores inteligentes, el reconocimiento de voz, la interpretación semánticas donde las máquinas son capaces de entender comentarios y conversaciones, así como el reconocimiento de imágenes que permite reconocerlas y clasificarlas cualitativamente de forma eficaz.

Cabe mencionar que con esta información se pretende presentar un segundo artículo en el cual se definirán hasta 10 posibles clases que podemos encontrar de diferentes tipos de sillas, tres diferentes tipos de mesas así como una puerta, el monitor de una computadora y hasta una ventana. Una vez definidas las clases se analizaran para empezar a desarrollar nuestro modelo para ser utilizado por el módulo de aprendizaje profundo de nuestra arquitectura. Se estudiará y analizaran los diferentes modelos que se utilizan para la clasificación de objetos que pertenecen a una clase.

Una vez teniendo las clases determinaremos el ángulo de acuerdo a la posición del usuario que observa al objeto a partir del entrenamiento de las diferentes características que definan a los objetos ubicados en nuestro escenario de cada una de las clases. A partir de la perspectiva del usuario se proyecta un modelo tridimensional en el dispositivo móvil, si sabemos que estamos viendo una silla para poder desplegar el modelo se determinara la posición a partir del ángulo con el que se esté observando, si se mira de frente a la silla se desplegará el objeto modificando su matriz de pose para desplegar el modelo 3D de frente, pero si esa misma silla se está mirando su parte trasera, la matriz de pose se rotará 180 grados para girar el modelo 3D para proyectarlo al mismo ángulo con respecto al observador.

IV. Referencias.

[1] Learning Deep Architectures for AI, Yoshua Bengio, https://www.iro.umontreal.ca/~lisa/pointeurs/TR1312.pdf, 28 de Marzo de 2016.
[2] DenseNet: Implementing Efficient ConvNet Descriptor Pyramids, Forrest Iandola, Matt Moskewicz, Sergey Karayev, http://arxiv.org/pdf/1404.1869.pdf, 28 de Marzo de 2016.
[3] Image Classification with Pyramid Representation and Rotated Data Augmentation on Torch7, Keven (Kedao) Wang, http://cs231n.stanford.edu/reports/kvw_cs231n.pdf, 28 de Marzo de 2016.
[4] Hierarchical Face Parsing via Deep Learning, Ping Luo, http://www.ee.cuhk.edu.hk/~xgwang/papers/luoWTcvpr12.pdf, 28 de Marzo de 2016.
[5] Página personal de Paul Milgram, http://etclab.mie.utoronto.ca/people/Paul.html, 28 de Marzo de 2016.
[6] Página personal de Fumio Kishio, http://www-human.ist.osaka-u.ac.jp/~kishino/index_eng.html, 28 de Marzo de 2016.
[7] Página personal de Ronald Azuma,http://www.ronaldazuma.com/,28 de Marzo de 2016
[8] Torch7, http://torch.ch/, 28 de Marzo de 2016