

# Seminario de deep learning y redes neuronales

## Deep Learning I

Carlos Andrés Delgado S. Msc

Universidad San Buenaventura, Cali

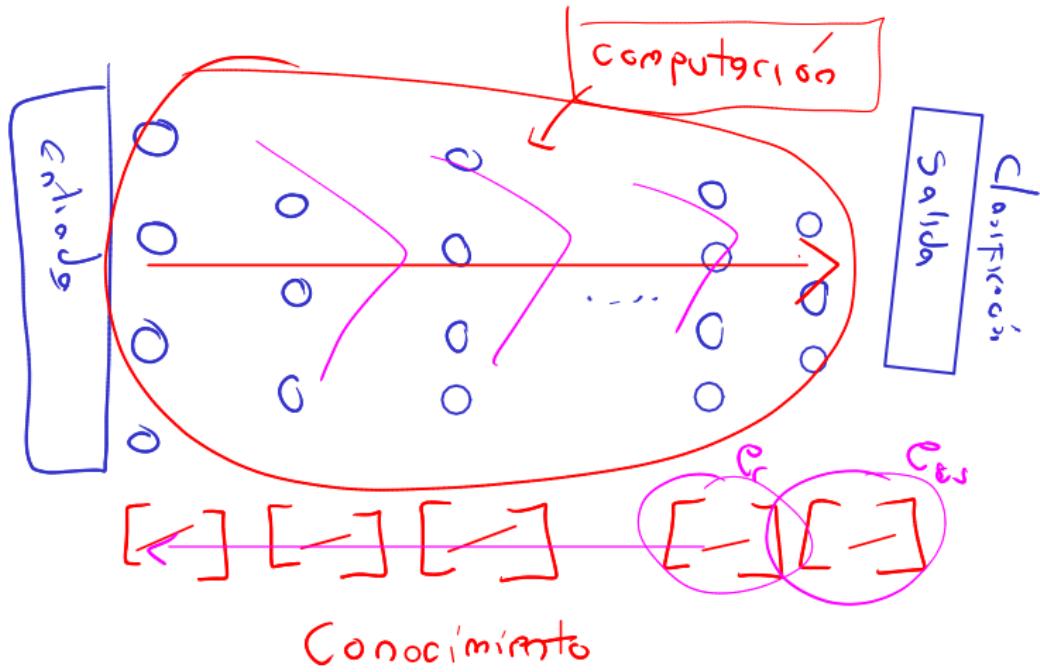
Febrero de 2021

# Contenido

- 1** Introducción
- 2** Auto-codificadores (Autoencoders)
- 3** Redes neuronales convolucionales
- 4** Librerías
  - Tensorflow
  - Keras
- 5** Diseño de aplicaciones con CNN

# Contenido

- 1** Introducción
- 2** Auto-codificadores (Autoencoders)
- 3** Redes neuronales convolucionales
- 4** Librerías
  - Tensorflow
  - Keras
- 5** Diseño de aplicaciones con CNN



# Introducción

## Definiciones

El Deep-Learning (Aprendizaje profundo) es un conjunto de técnicas de aprendizaje de máquina que se basa en aprendizaje de datos.

- Se busca solucionar problemas muy difíciles (muchas características y clases), en el menor tiempo posible
- Existe un gran conjunto de técnicas de Deep-Learning: Regresión, clusterización, predictores, entre otros. clases, f109c105
- Es utilizado ampliamente en sistemas de recomendación. Por ejemplo, el reconocimiento de rostros al momento de tomar fotos
- Gracias al Deep-Learning las redes neuronales se han convertido en una técnica ampliamente utilizada.

# Introducción

## Definiciones

Sin embargo, hasta el momento conocemos los problemas de las redes neuronales

- Si el problema tiene muchas características, la red neuronal será muy grande
- Si la red neuronal tiene muchas entradas y muchas capas, el entrenamiento podría ser computacionalmente inviable
- En algunos casos las características de un problema son dependientes entre sí y las redes neuronales no tienen la capacidad de decisión sobre si alguna de ellas se puede descartar

# Introducción

## Definiciones

### El enfoque de Deep-Learning

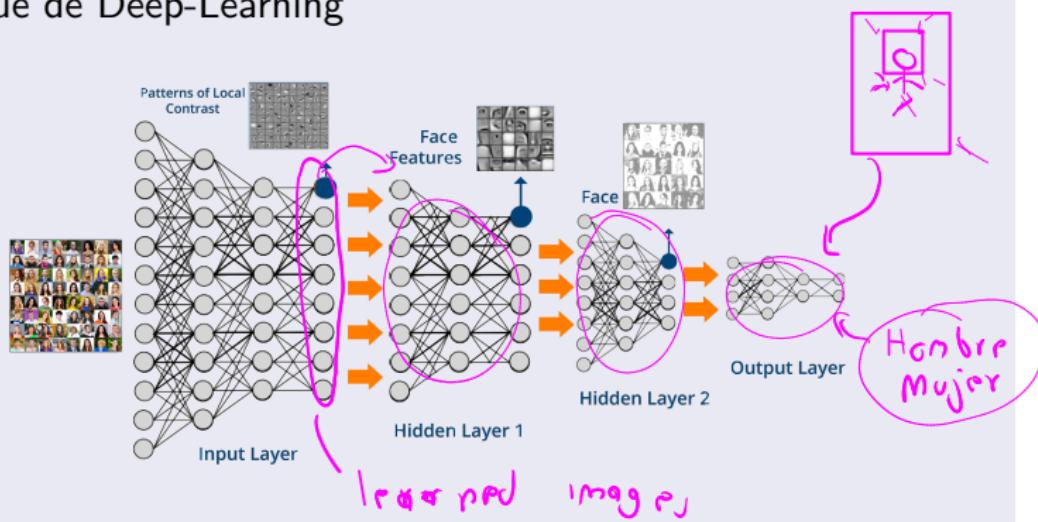


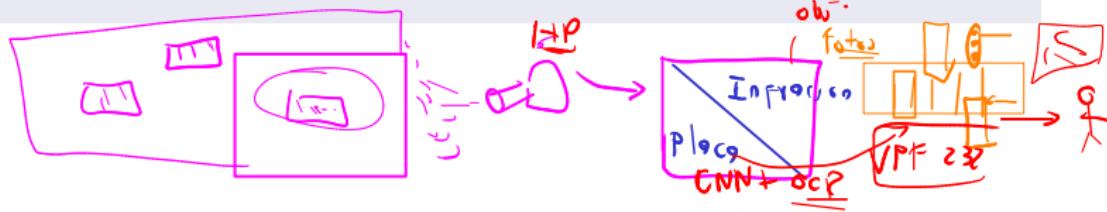
Figura 1: Tomado de <https://cdn.edureka.co/>

# Introducción

## Aplicaciones

Bajo el enfoque de Deep-Learning se han implementado varias

- Traducciones instantáneas (Google)
- Reconocimiento de rostros (Celulares)
- Sistemas de recomendación (Facebook - Amazon)
- Fotomultas ) (Reconocimiento de imágenes)



# Introducción

## Aplicaciones

Debido a que las redes neuronales matemáticamente se pueden representar como matrices de pesos, se pueden utilizar GPUs para procesarlas de forma más eficiente que con CPUs.

Recuerden que las GPUs procesan transformaciones no lineales de matrices (que representan las proyecciones de un información de imágenes de 3D a 2D) por lo que son el elemento ideal para procesamiento de algoritmos de Deep-Learning

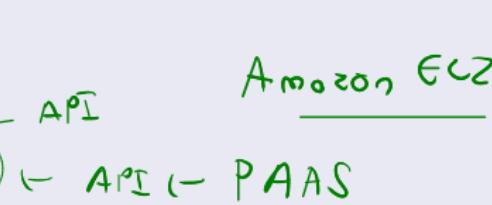
$$O(n^3)$$

# Introducción

## Definiciones

### Herramientas gratuitas

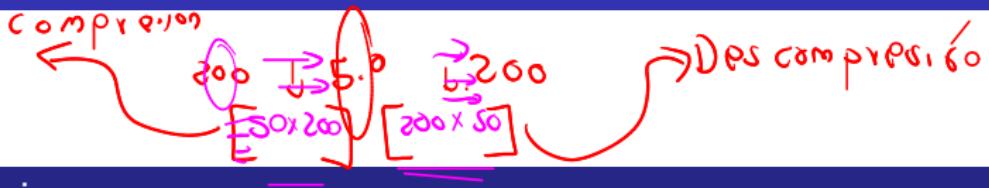
- Tensorflow
- Pytorch
- Keras
- Google ML
- IBM Watson



# Contenido

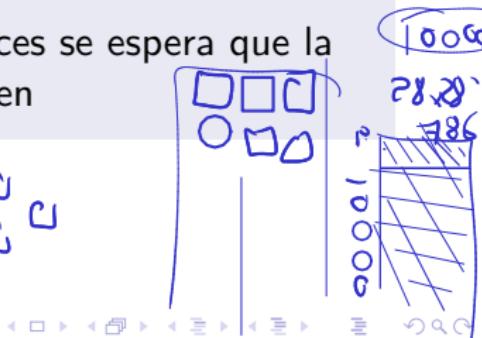
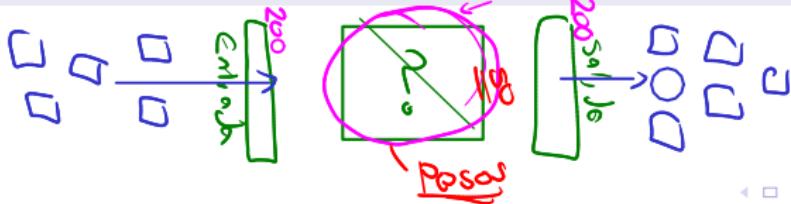
- 1 Introducción
- 2 Auto-codificadores (Autoencoders)
- 3 Redes neuronales convolucionales
- 4 Librerías
  - Tensorflow
  - Keras
- 5 Diseño de aplicaciones con CNN

# Autoencoders



## Definiciones

- Suelen ser redes de 3 capas
- Se aprende a producir la salida que sea exactamente igual que la entrada
- Las capas de entrada deben tener el mismo número de neuronas de salida
- Ejemplo, si se recibe una imagen, entonces se espera que la red aprenda a reproducir la misma imagen



# Autoencoders

## Definiciones

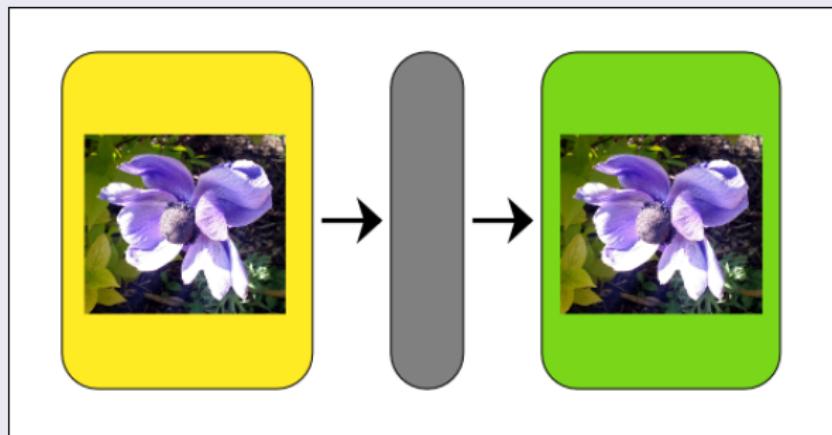


Figura 2: Tomado de <https://rubenlopezg.files.wordpress.com>

# Autoencoders

## Definiciones

Sin embargo...

Esto es inútil a simple vista ¿Que ganamos?

# Autoencoders

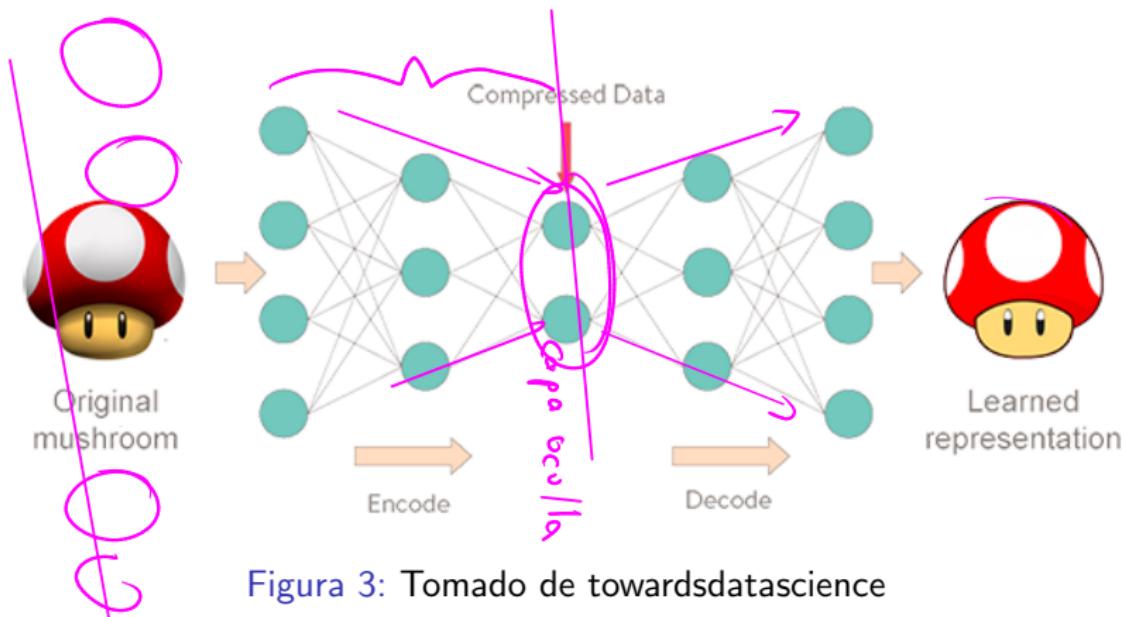


Figura 3: Tomado de towardsdatascience

# Autoencoders

## Definiciones

- La clave está en la capa oculta
- Suponga que la capa oculta tiene menos neuronas que las capas de entrada y salida
- Entonces: **En esta capa ganamos compresión de la información**
- Además: **La capa oculta deberá comprimir y descomprimir la información**

# Autoencoders

## Definiciones

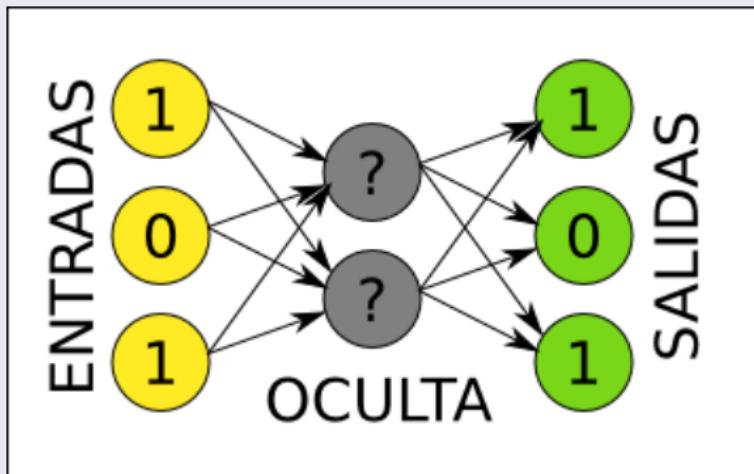


Figura 4: Tomado de <https://rubenlopezg.files.wordpress.com>

# Autoencoders

## Definiciones

- Una vez entrenada la red se puede dividir en dos
  - La primera que utiliza la capa oculta como salida
  - La segunda que utiliza la capa oculta como entrada
- Tenemos una red para comprimir y otra para descomprimir

# Autoencoders

## Definiciones

- Sin embargo, comprimir información no es el uso habitual de estas redes
- ¿Que pasa si la capa oculta tuviste más neuronas?
- Pero existe un riesgo: Que la red se limite a copiar la entrada y la salida
- Entonces, vamos a forzar que las neuronas de la capa oculta se activen pocas veces, así la red tendría que aprender un código alternativo y además disperso

# Autoencoders

## Definiciones

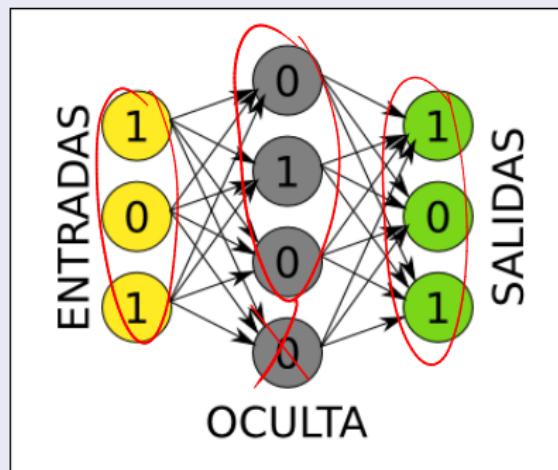
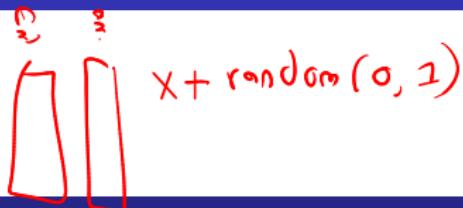


Figura 5: Tomado de <https://rubenlopezg.files.wordpress.com>

# Autoencoders



## Definiciones

Existen varias formas de conseguir que la red no se limite a copiar la información:

- Añadir a la hora de calcular el error, un factor de dispersión.  
Si este es bajo añade mucho al error y si es alto poco.
- Introducir un poco de ruido a los vectores de entrada y que la salida no tenga el ruido. Así obligamos a la red a generalizar

# Autoencoders

## Ejemplo

Vamos a observar un sistema de reconocimiento de dígitos escritos a mano:

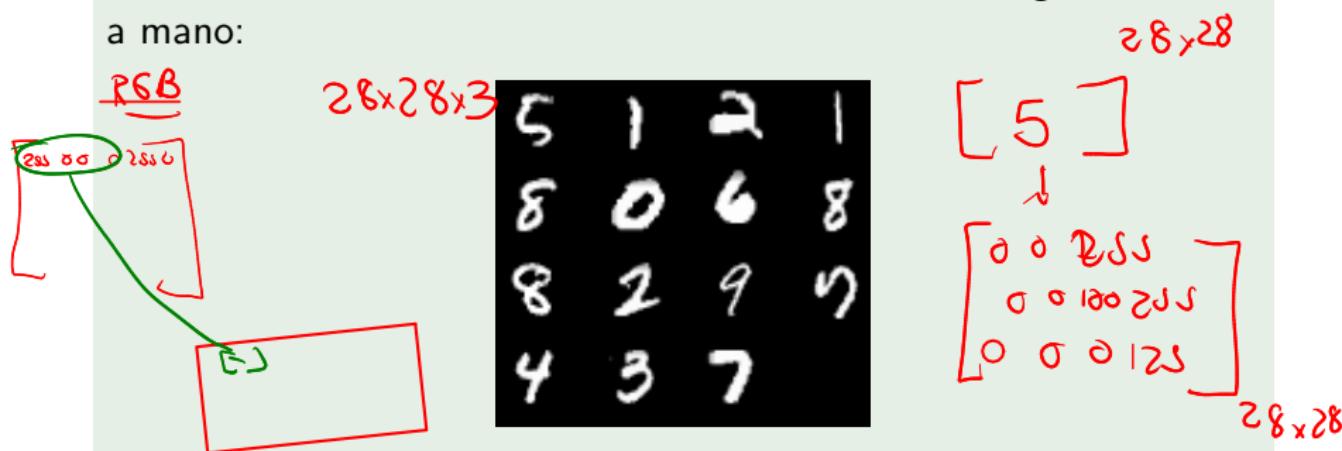
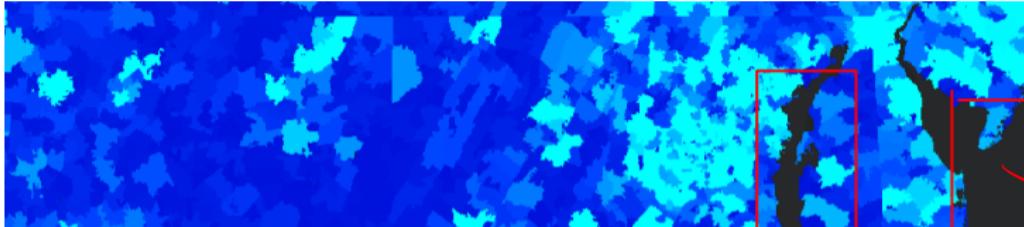
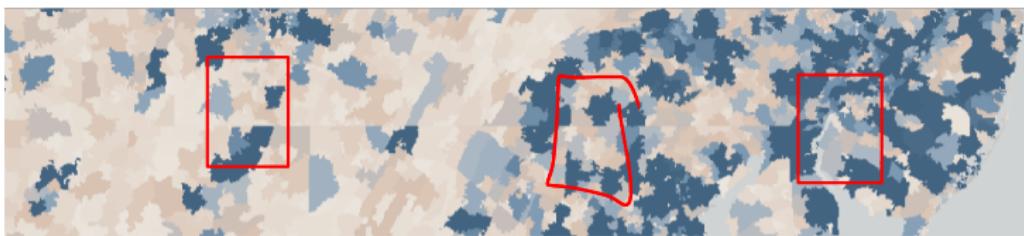


Figura 6: Tomado de <https://rubenlopezg.files.wordpress.com>

$$(0.3 * R) + (0.59 * G) + (0.11 * B)$$



Profundidae



# Autoencoders

## Definiciones

- Para entrenar vamos a tomar imágenes de la base de datos Minst
- Se tienen 60000 imágenes de entrenamiento y 10000 imágenes de prueba de 28x28
- Dado que necesitamos codificar las imágenes como un vector de entrada unidimensional, este tendrá 784.

# Autoencoders

## Ejemplo

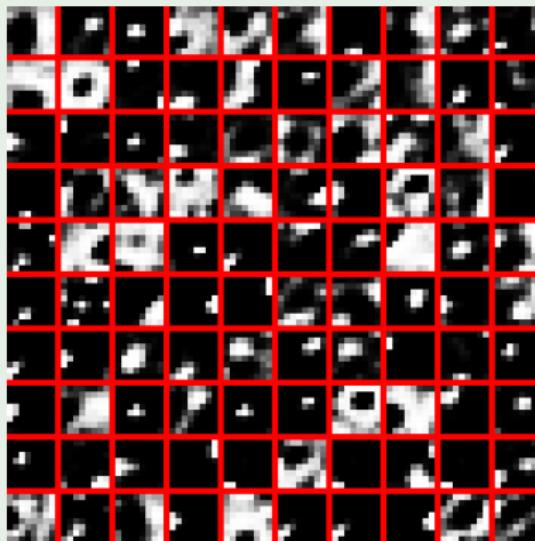
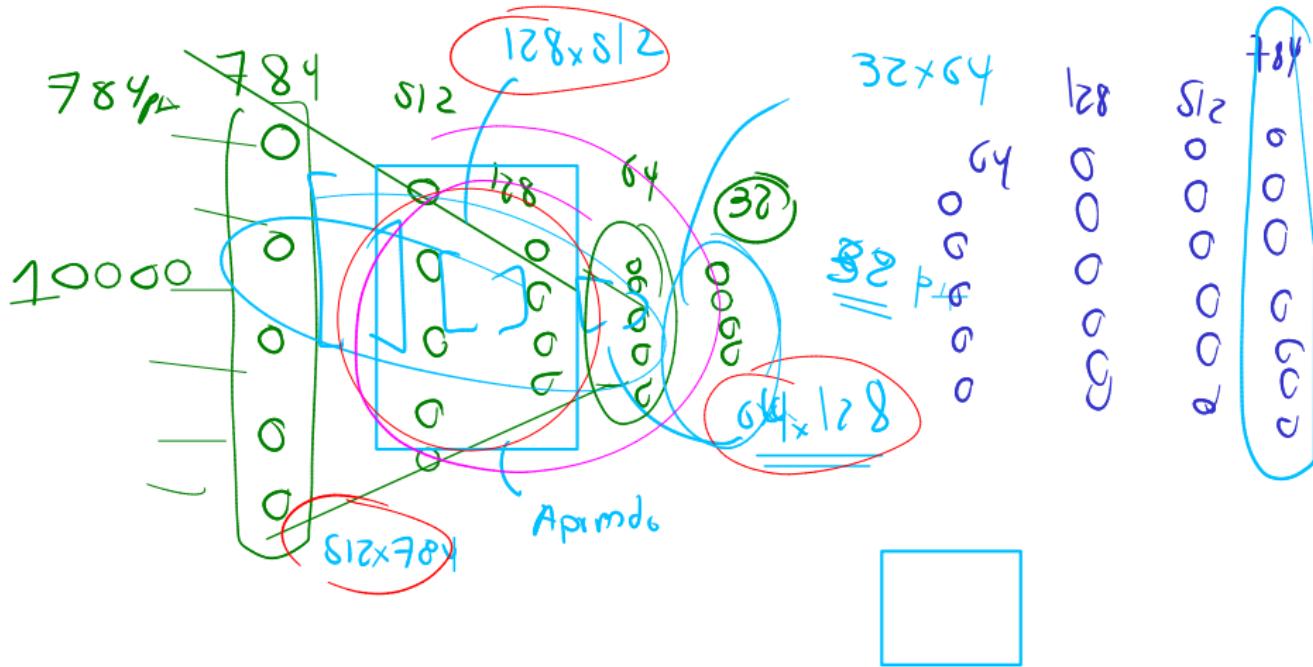


Figura 7: Tomado de <https://rubenlopezg.files.wordpress.com>

# Autoencoders

## Definiciones

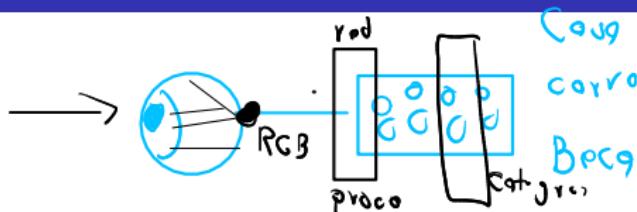
- La idea de realizar Deep-Learning con autoencoders es usar varios codificadores
- Cada codificador se encarga de un conjunto de entradas o de características
- Estos enconders realizan el proceso del aprendizaje de los datos en los algoritmos de Deep-Learning



# Contenido

- 1 Introducción
- 2 Auto-codificadores (Autoencoders)
- 3 Redes neuronales convolucionales
- 4 Librerías
  - Tensorflow
  - Keras
- 5 Diseño de aplicaciones con CNN

# Redes neuronales convolucionales



## Definiciones

- Está inspirado en la corteza visual
- Es una variación del perceptrón multicapa
- Su aplicación es realizada en matrices bidimensionales
- Son redes de múltiples capas de filtros convencionales de una o más dimensiones

# Redes neuronales convolucionales

## Estructura redes convolucionales

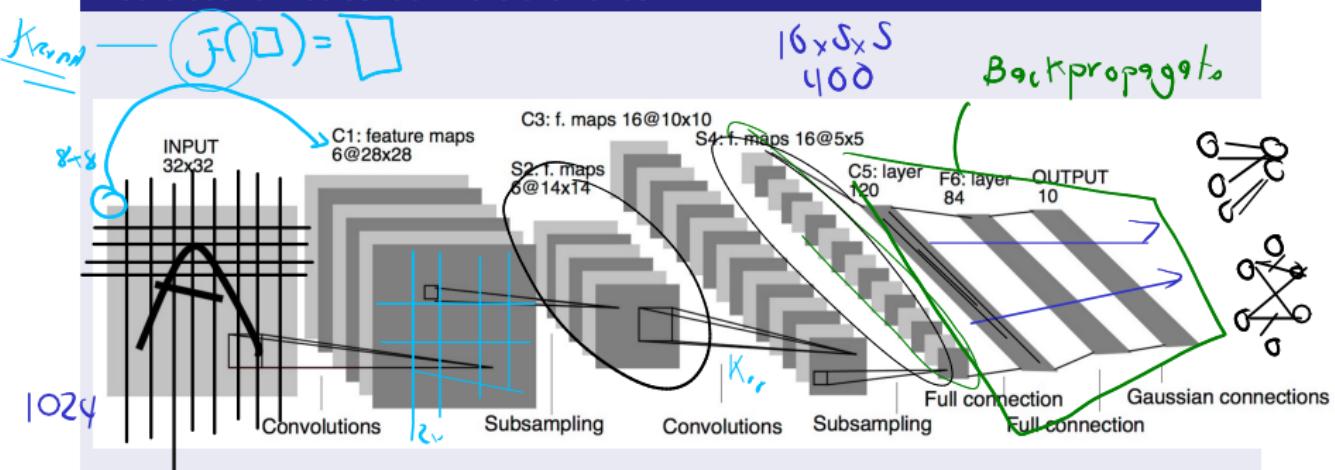


Figura 8: Tomado de LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition.

# Redes neuronales convolucionales

## Definiciones

- Las capas entre sí están conectadas a través de funciones no lineales
- Estas redes realizan la extracción y reducción de las características a medida que se avanza en las capas ocultadas
- Estas redes permiten reducir la dimensionalidad de los problemas (número de parámetros) pero se son más sensibles a cambios en las entradas

# Redes neuronales convolucionales

## Definiciones

- la convolución es un proceso que consiste en filtrar una imagen usando una máscara
- Diferentes mascaras retornan diferentes resultados.

# Redes neuronales convolucionales



Figura 9: Tomado de <https://ccc.inaoep.mx>

# Redes neuronales convolucionales

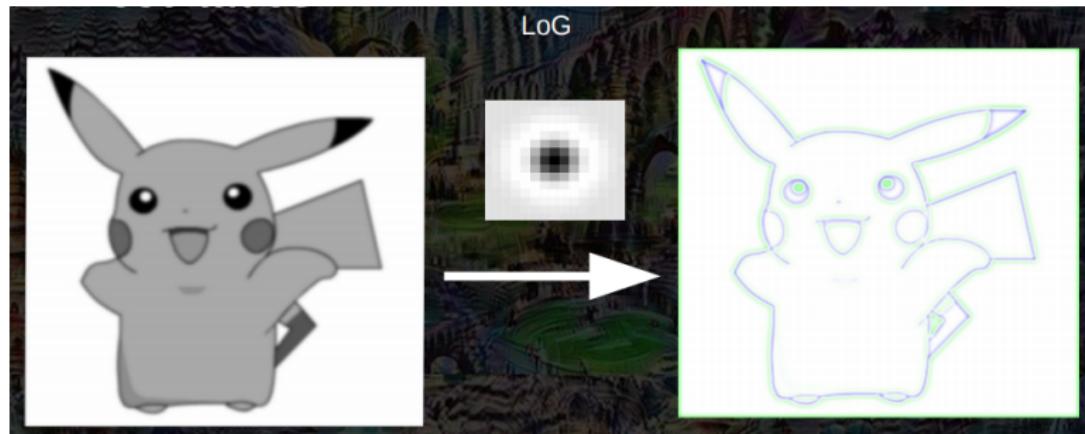


Figura 10: Tomado de <https://ccc.inaoep.mx>

# Redes neuronales convolucionales

## Definiciones

En la convolución cada píxel es una combinación lineal de los píxeles de entrada

- Se tiene una ventana deslizante a través de toda la imagen (zona que tomamos como entrada)
- Las máscaras son representados con los pesos de la red

pesos de la red  
peso

# Redes neuronales convolucionales

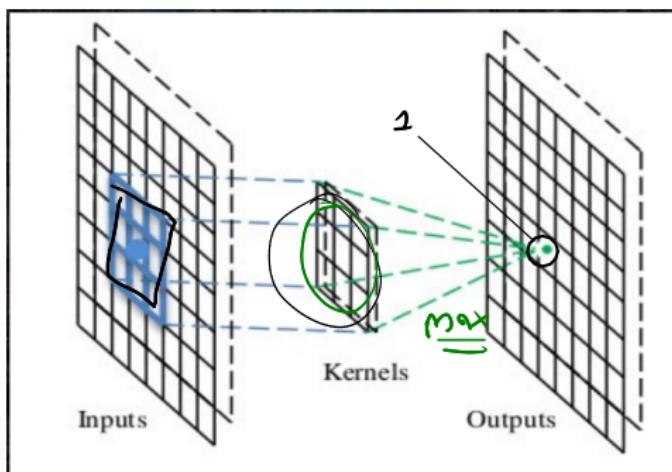


Figura 11: Tomado de <https://ccc.inaoep.mx>

# Redes neuronales convolucionales

## Definiciones

- En estas redes cada capa es un volumen de neuronas en 3D
- Por lo tanto, cada capa tiene su alto, ancho y profundidad

# Redes neuronales convolucionales

## Definiciones

Las redes tienen 3 tipos de capas

- Capas convolucionales: Requieren el uso de mascaras
- Capas de pooling: La salida es máximo de la entrada en una ventana
- Capas totalmente conectadas: Se aplican al final, se pierde información.

# Redes neuronales convolucionales

## Definiciones

Existen dos tipos de arquitecturas. La primera son las redes convolucionales:

- Entrega una salida para toda la imagen
- Salida totalmente conectada

# Redes neuronales convolucionales

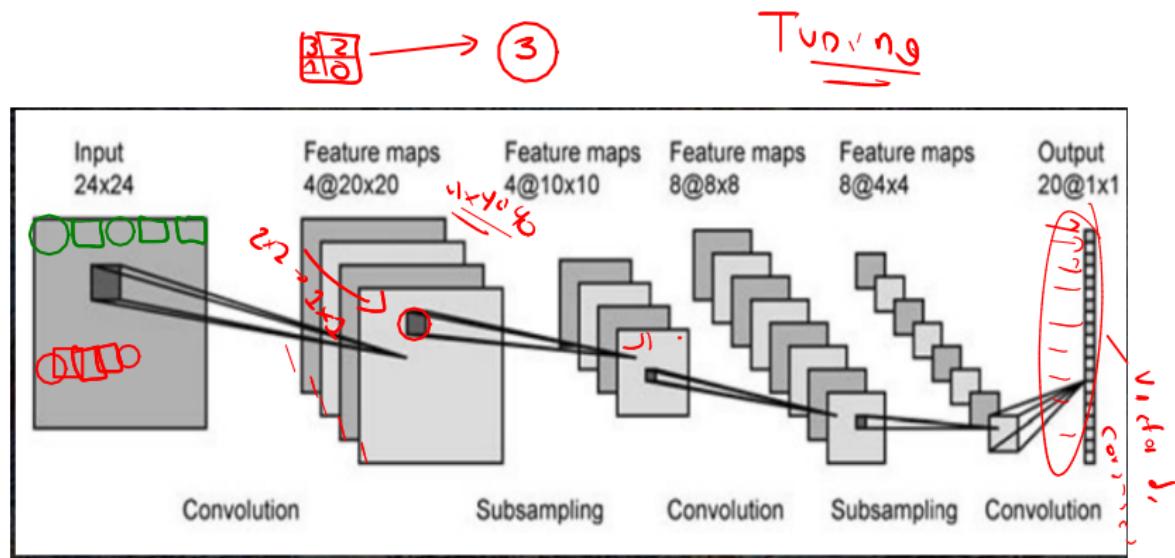


Figura 12: Tomado de <https://ccc.inaoep.mx>

# Redes neuronales convolucionales

## Definiciones

Existen dos tipos de arquitecturas. La segunda son las convolucionales completas:

- Tienen un decodificador(encoder) y un codificador(decoder)
- Comprimen la información
- Entregan una salida por Píxel

# Redes neuronales convolucionales

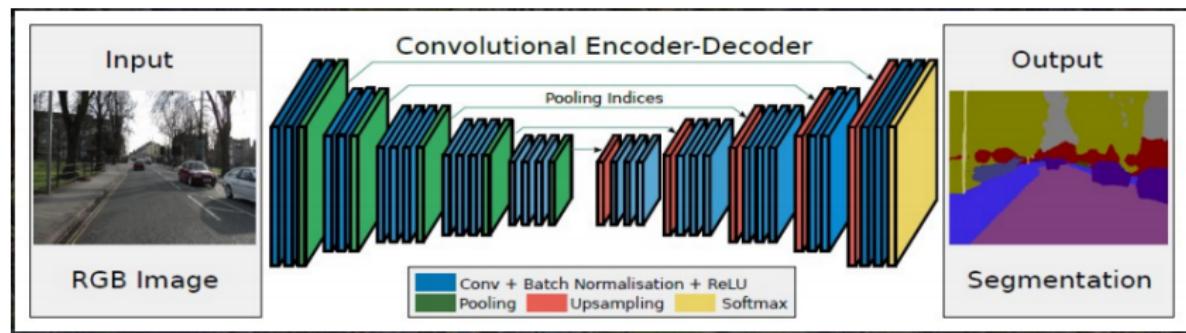
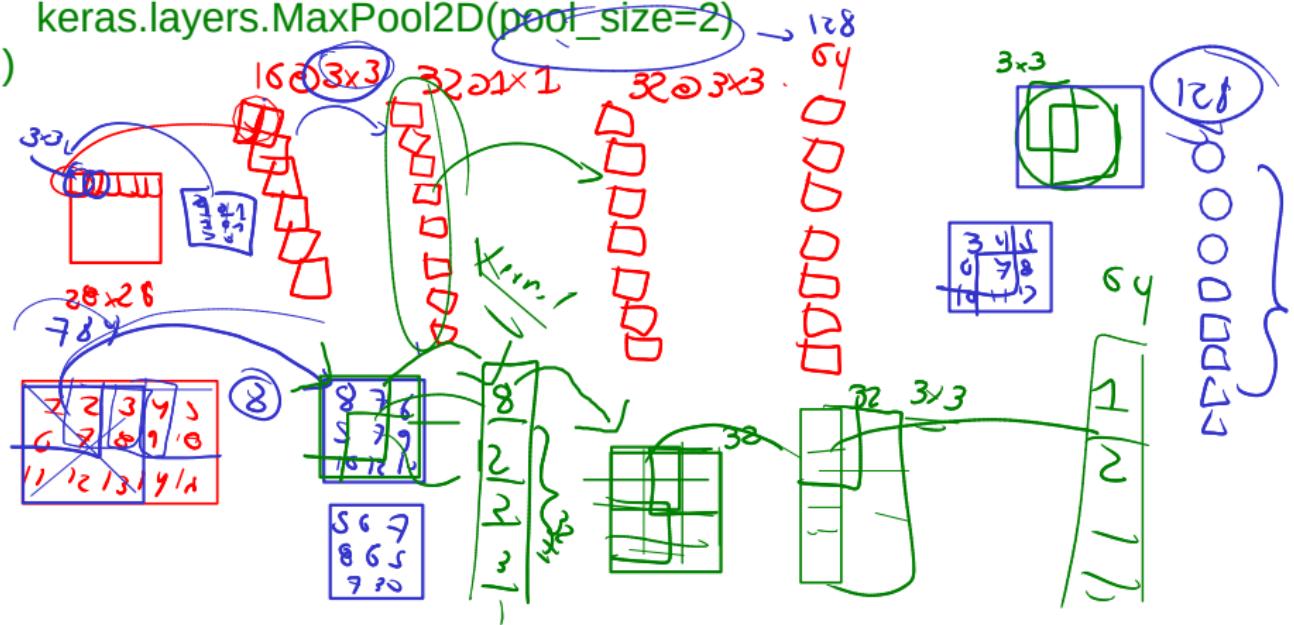


Figura 13: Tomado de <https://ccc.inaoep.mx>

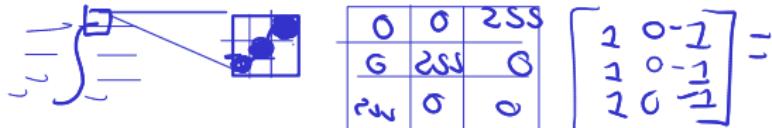
```

encoder = keras.models.Sequential([
    keras.layers.Reshape([28, 28, 1], input_shape=[28, 28]),
    keras.layers.Conv2D(16, kernel_size=(3, 3), padding="same", activation="relu"),
    keras.layers.MaxPool2D(pool_size=2),
    keras.layers.Conv2D(32, kernel_size=(3, 3), padding="same", activation="relu"),
    keras.layers.MaxPool2D(pool_size=2),
    keras.layers.Conv2D(64, kernel_size=(3, 3), padding="same", activation="relu"),
    keras.layers.MaxPool2D(pool_size=2)
])

```



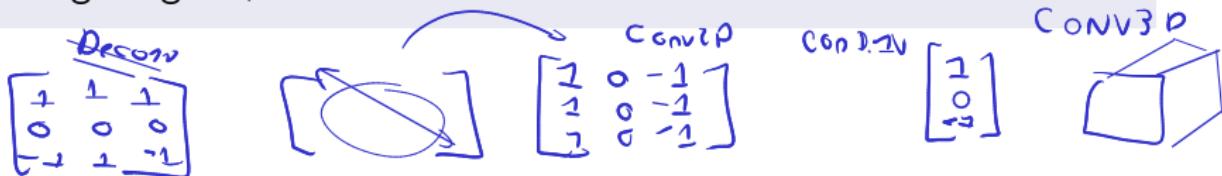
# Redes neuronales convolucionales



## Definiciones

Las redes convolucionales son apetecidas en el momento:

- Solucionan problemas difíciles de reconocimiento de imágenes reduciendo la dimensionalidad del problema
- Se pueden implementar en GPU (Dado su estructura matricial)
- Se adaptan a diferentes problemas: Reconocimiento de anomalías en imágenes médicas, identificación de zonas geológicas, entre otras.



# Contenido

- 1 Introducción
- 2 Auto-codificadores (Autoencoders)
- 3 Redes neuronales convolucionales
- 4 Librerías
  - Tensorflow
  - Keras
- 5 Diseño de aplicaciones con CNN

# Contenido

- 1 Introducción
- 2 Auto-codificadores (Autoencoders)
- 3 Redes neuronales convolucionales
- 4 Librerías
  - Tensorflow
  - Keras
- 5 Diseño de aplicaciones con CNN

# Tensorflow

## Definiciones

Tensorflow = Tensor + Flow = Data + Flow

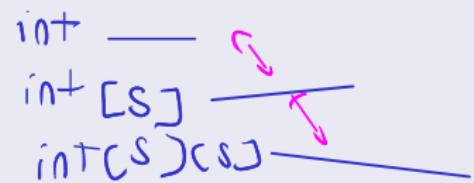
- Es una librería de código abierto para Deep Learning y Machine Learning
- Desarrollada por el Google Brain Team en Noviembre de 2015
- Es utilizada para **clasi**cación****, percepción, entendimiento, descubrimiento, predicción y creación.

# Tensorflow

## ¿Qué es un tensor?

- 0-d tensor: Escalar (número)
- 1-d tensor: Vector
- 2-d tensor: Matriz
- n-d tensor: Matriz enésima

int —————  
int [S] —————  
int(S)(S) —————

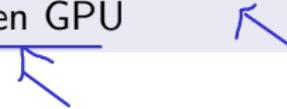


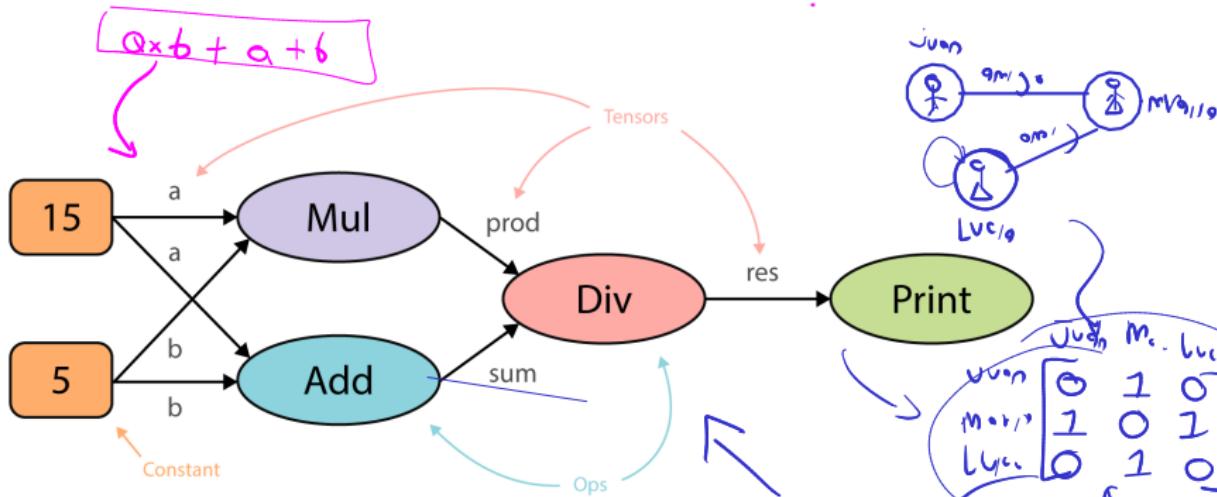
# Tensorflow

## Grafos de flujo

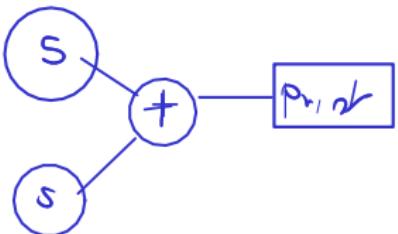
- Son un modelo para computación paralela
- Tensorflow usa un grafo de flujo para representar su computación

Esta estrategia permite parallelizar la ejecución de la librería y optimizar para ejecutar en GPU





$S + S$



# Tensorflow

## Servicios con tensorflow

- Google Cloud Speech
- Google Photos
- Google Search

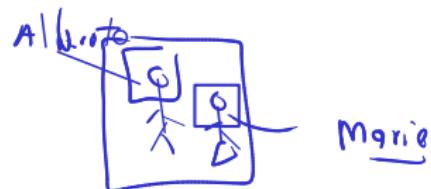
# Contenido

- 1** Introducción
- 2** Auto-codificadores (Autoencoders)
- 3** Redes neuronales convolucionales
- 4** Librerías
  - Tensorflow
  - Keras
- 5** Diseño de aplicaciones con CNN

## Definiciones

- Es una librería para construir redes neuronales con bloques
- Trae soporte para redes neuronales concurrentes y recurrentes
- Es bastante utilizada por su facilidad de instalación en diferentes sistemas.

# Keras



## Aplicaciones

- Teléfonos móviles
- Sistemas de recomendación en Facebook y Amazon

# Contenido

- 1** Introducción
- 2** Auto-codificadores (Autoencoders)
- 3** Redes neuronales convolucionales
- 4** Librerías
  - Tensorflow
  - Keras
- 5** Diseño de aplicaciones con CNN

## Datos

- Se requiere imágenes con el mismo tamaño
- Estas son matrices, donde cada pixel tiene un valor (blanco y negro) y tres valores (a color)
- Se deben preprocesar:
  - Normalizar los datos entre 0 y 1
  - Si hay 3 canales, transformar a blanco y negro o tomar un sólo canal (Excepto si los colores tienen significado)

# Diseño de aplicaciones con CNN

## Convoluciones

- Consiste en tomar grupos de pixeles y representarlos como un solo dato
- Para esto se usa una función denominada Kernel.

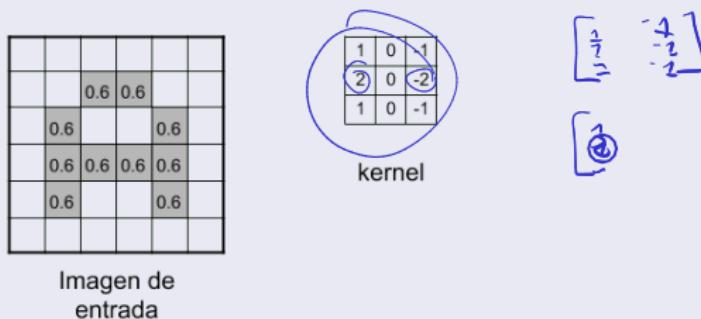


Figura 14: Tomado de [www.aprendemachinelearning.com](http://www.aprendemachinelearning.com)

## Filtros: Conjuntos de Kernels

- En realidad no se aplica un sólo kernel, si no muchos
- Por ejemplo, podríamos tener 32 filtros, lo que corresponde a 32 matrices de salida
- En total tendríamos  $28 * 28 * 1 * 32 = 25088$  neuronas en la primera capa oculta

# Diseño de aplicaciones con CNN

## Filtros: Conjuntos de Kernels

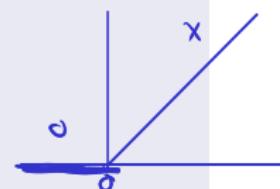
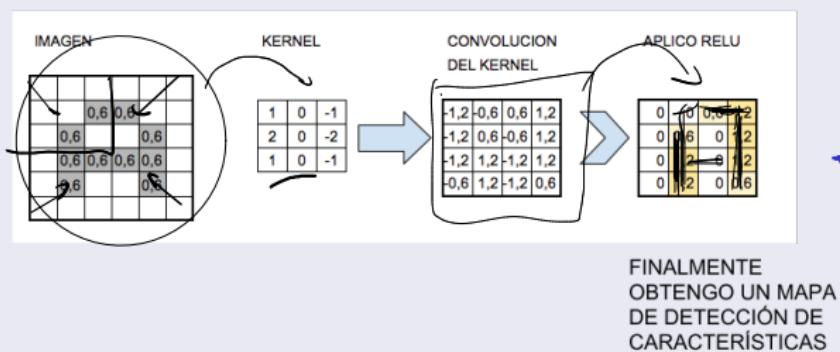


Figura 15: Tomado de [www.aprendemachinelearning.com](http://www.aprendemachinelearning.com)

# Diseño de aplicaciones con CNN

## Filtros: Conjuntos de Kernels

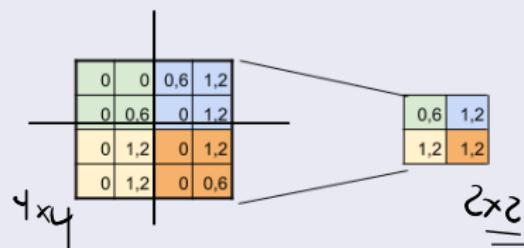
La función de activación más utilizada aquí es Relu,  
 $f(x) = \max(0, x)$ .

## Subsampling

- Tenemos una capa oculta con 25088 neuronas
- Si se hace una nueva convolución, aumenta exponencialmente el número de neuronas
- Para reducir este tamaño, se hará subsampling para reducir el tamaño de las imágenes filtradas

# Diseño de aplicaciones con CNN

## Subsampling



SUBSAMPLING:  
Aplico Max-Pooling de  $2 \times 2$   
y reduzco mi salida a la mitad

Figura 16: Tomado de [www.aprendemachinelearning.com](http://www.aprendemachinelearning.com)

# Diseño de aplicaciones con CNN

## Primera convolución

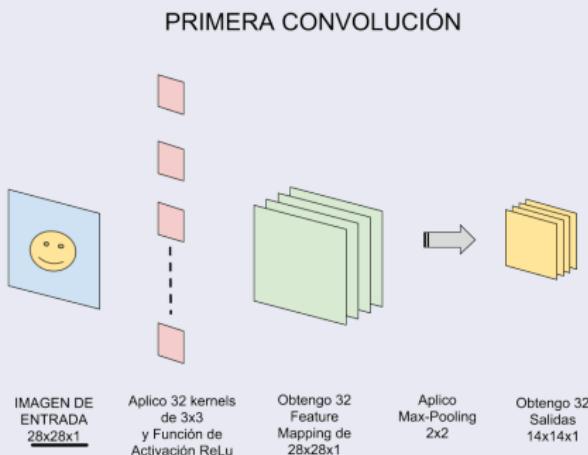


Figura 17: Tomado de [www.aprendemachinelearning.com](http://www.aprendemachinelearning.com)

# Diseño de aplicaciones con CNN

## Segunda convolución

### SEGUNDA CONVOLUCIÓN (y sucesivas)

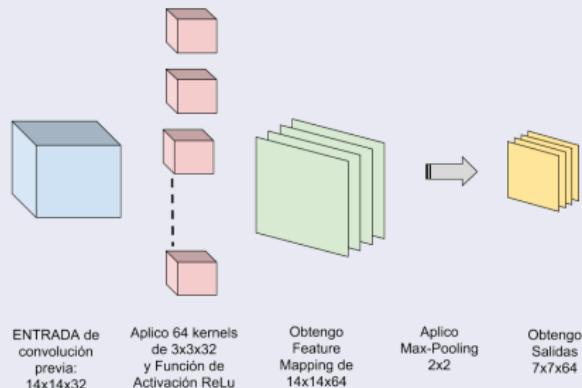


Figura 18: Tomado de [www.aprendemachinelearning.com](http://www.aprendemachinelearning.com)

# Diseño de aplicaciones con CNN

## Red totalmente conectada

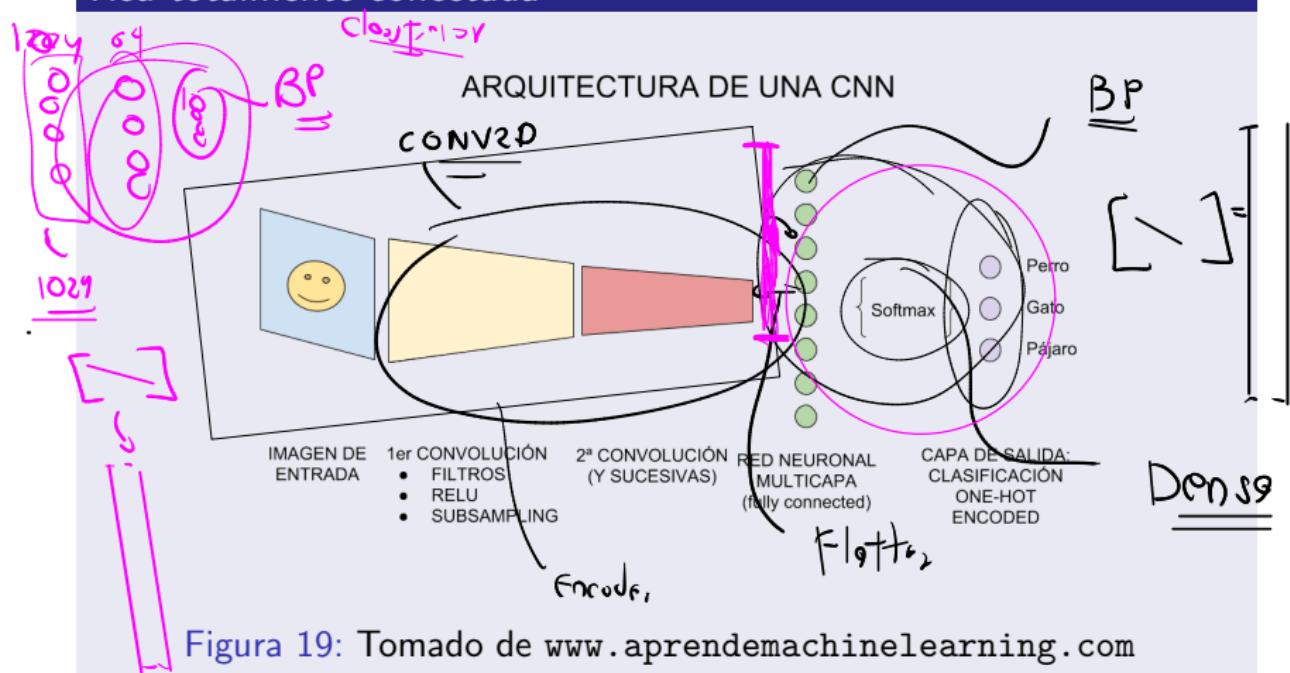
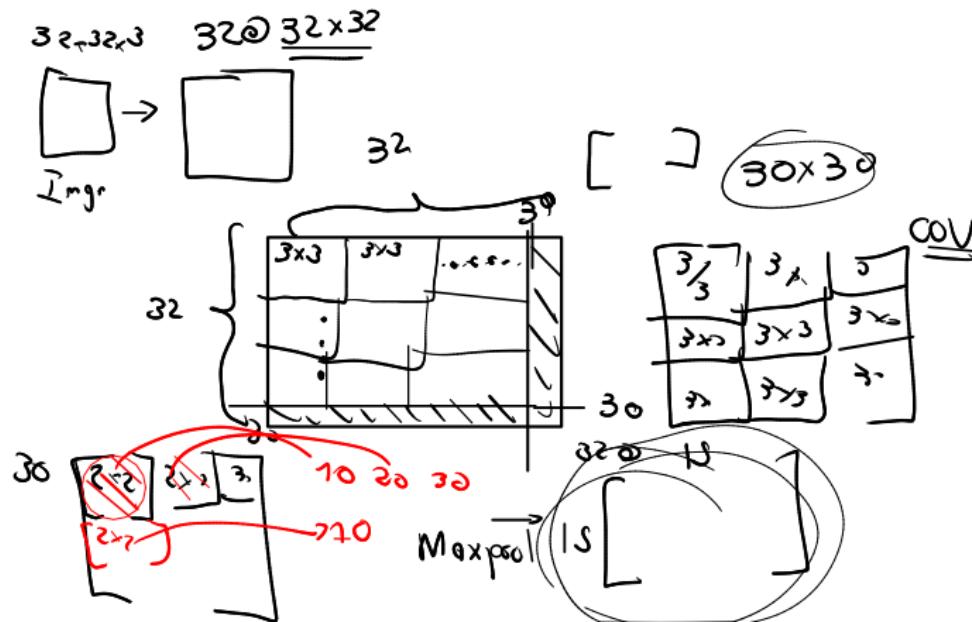


Figura 19: Tomado de [www.aprendemachinelearning.com](http://www.aprendemachinelearning.com)

```
model = models.Sequential()
```

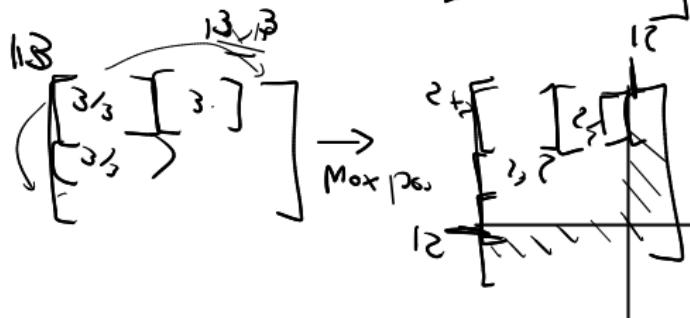
```
model.add(layers.Conv2D(32, (3, 3), activation='relu'))  
input_shape=(32, 32, 3)))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(64, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

32 map



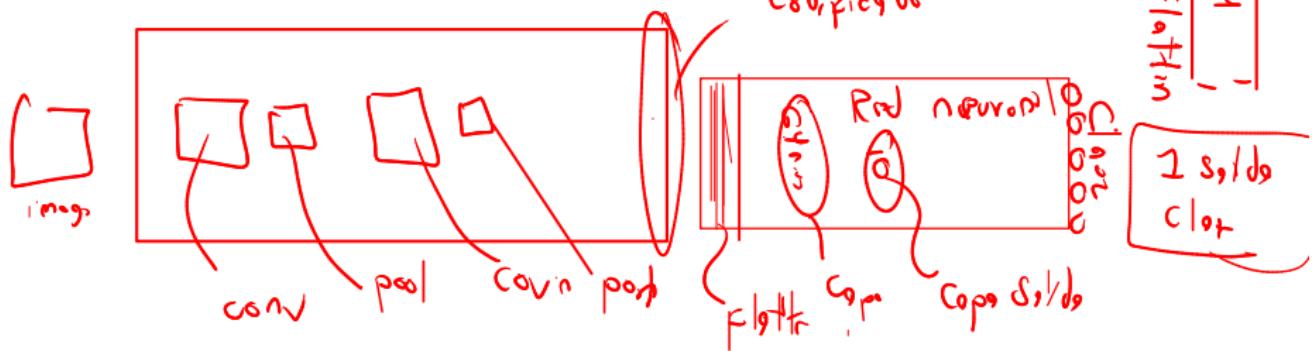
32@

IS

64@  $|S \times S|$ 

Imag  
Config  
ds

$$\begin{bmatrix} 4 \times 4 \times 6 \end{bmatrix} = \begin{bmatrix} - & - \\ - & - \\ - & - \\ - & - \end{bmatrix}$$



# Referencias I

-  Curso de modelos computacionales.  
[http://www.lcc.uma.es/~munozp/.](http://www.lcc.uma.es/~munozp/)  
Accessed: Ocubre-2017.
-  Du, K.-L. and Swamy, M. N. S. (2010).  
*Neural Networks in a Softcomputing Framework*.  
Springer Publishing Company, Incorporated, 1st edition.
-  Heaton, J. (2008).  
*Introduction to Neural Networks with Java*.  
Heaton Research.
-  Li, H. (2000).  
*Fuzzy Neural Intelligent Systems: Mathematical Foundation and the Applications in Engineering*.  
CRC Press.

# ¿Preguntas?

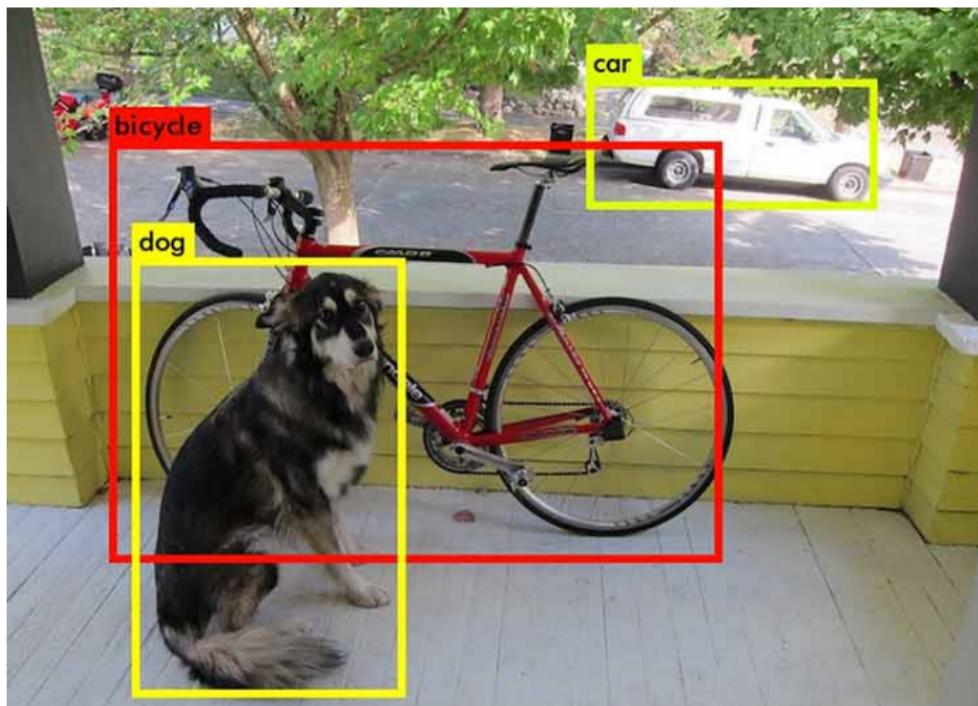


Figura 20: Tomado de [www.medium.com](http://www.medium.com)