

Intro a Redes Neuronales con Keras



Rodolfo Ferro.



¡Qué onda!

Soy Rodo Ferro

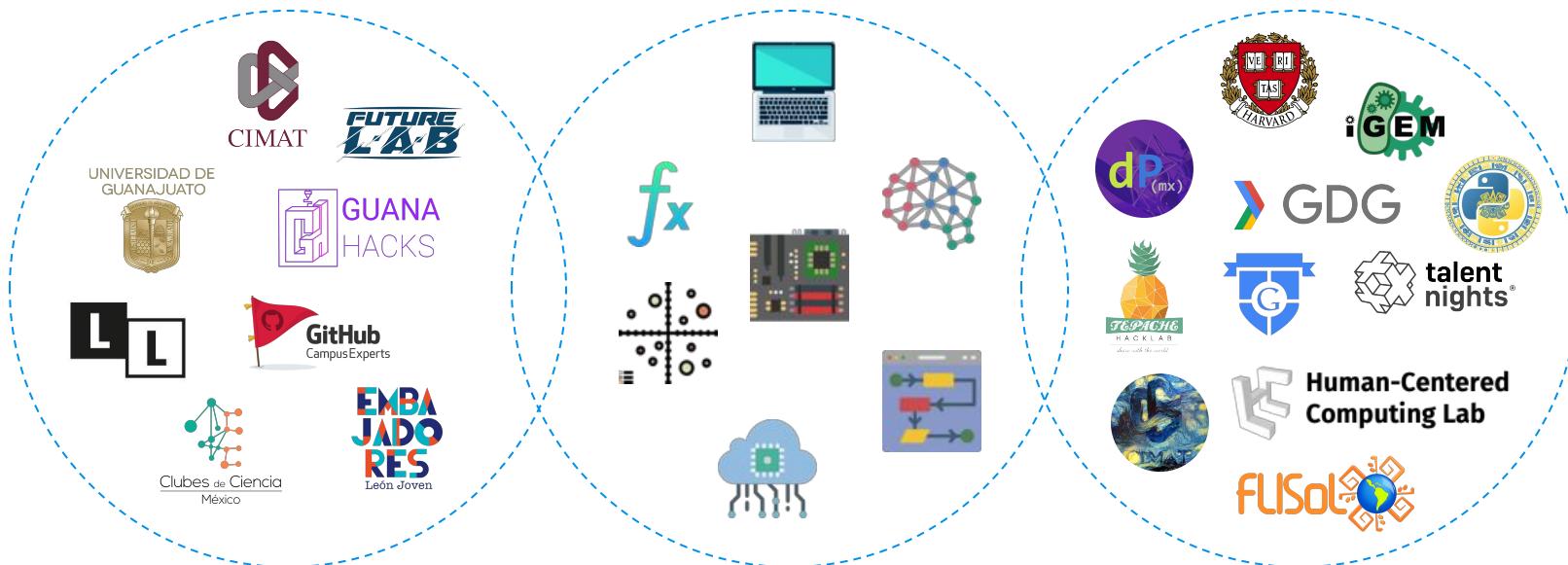
Computer-
Mathemagician $f[\square]$
GitHub Campus Expert

Encuéntrame:

 @FerroRodolfo
 @rodo_ferro



Lo que hago:



Objetivos:

- Contextualizar sobre IA
 - Entender bases de redes neuronales
 - Conocer Keras, framework para desarrollar redes neuronales con Python
 - Desarrollar un modelo de clasificación
-
- + Mostrar un servicio web que utiliza una red neuronal entrenada...



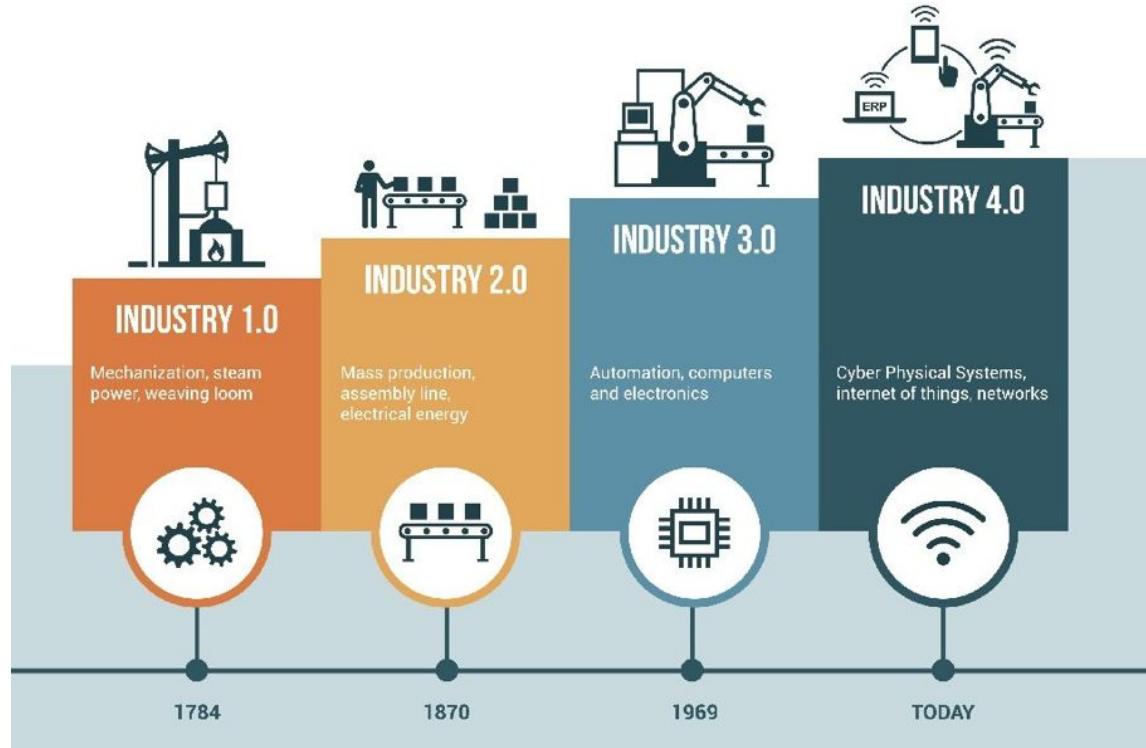


1.

Contexto

Situación actual y relevancia

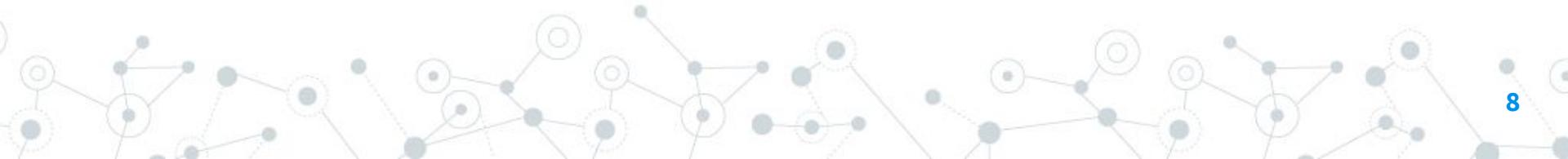
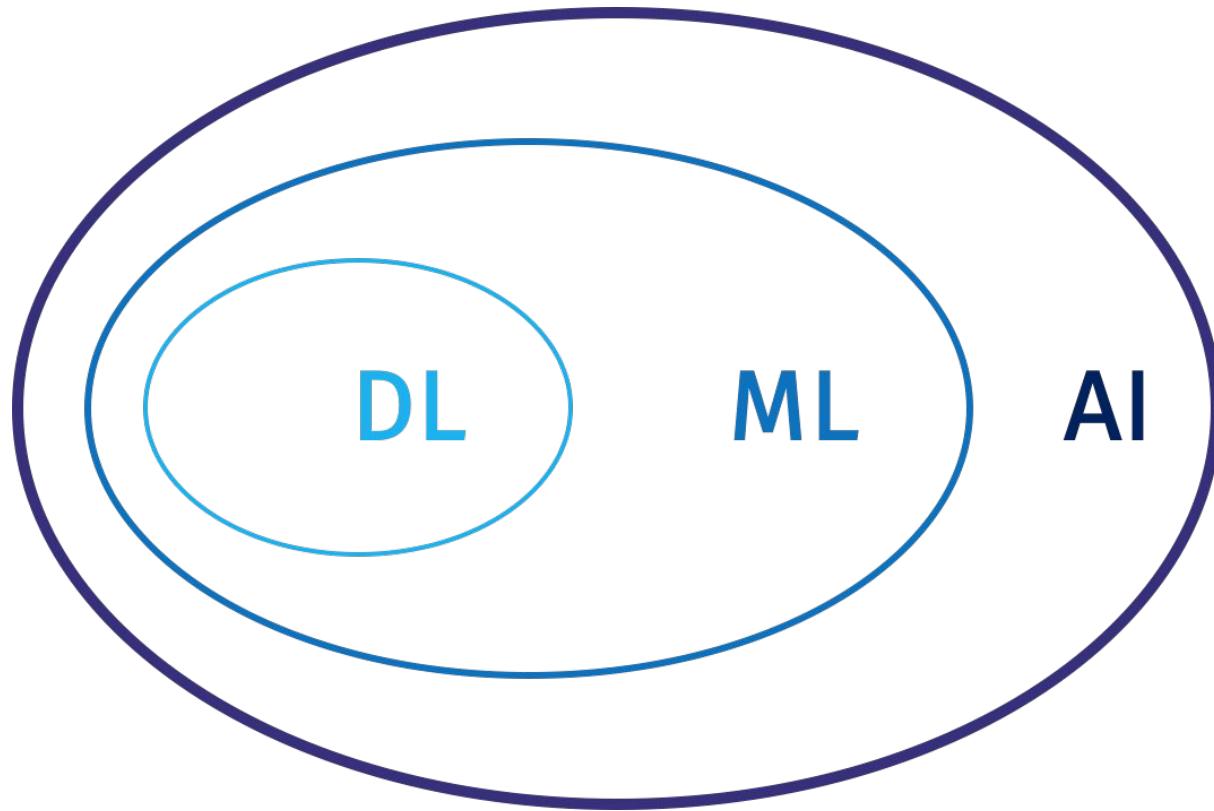
RI4.0



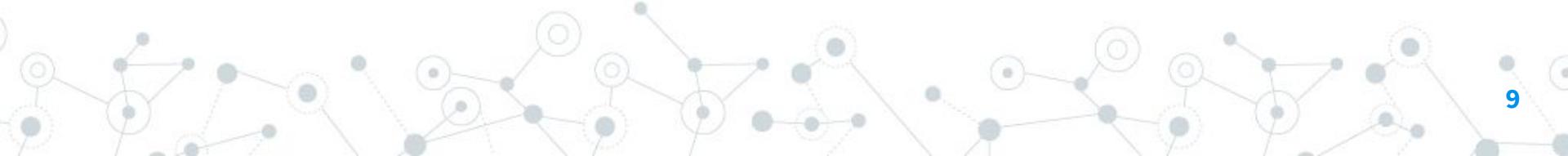
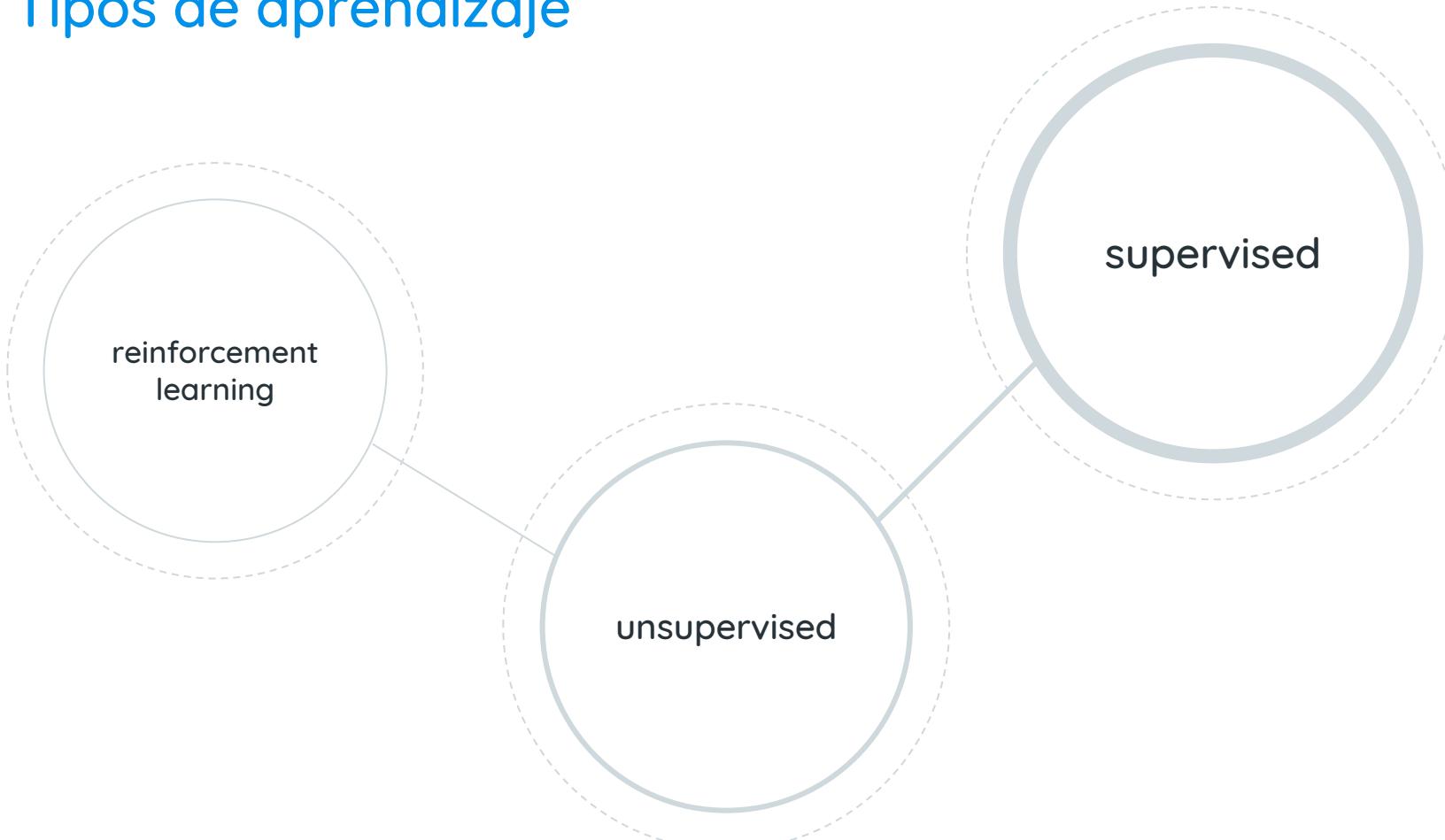
Fuente: Industry 4.0 and Industrial IoT in Manufacturing:
A Sneak Peek, Aberdeen Essentials.



AI, ML, DL

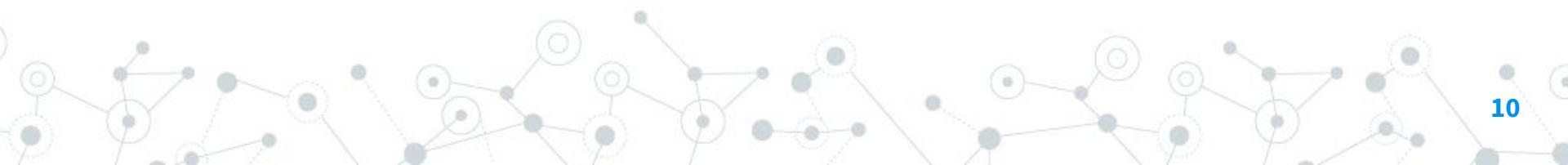
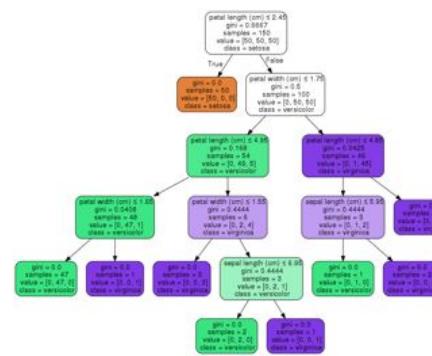
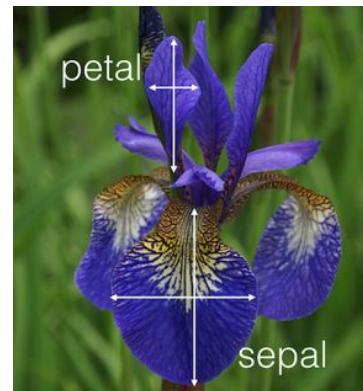
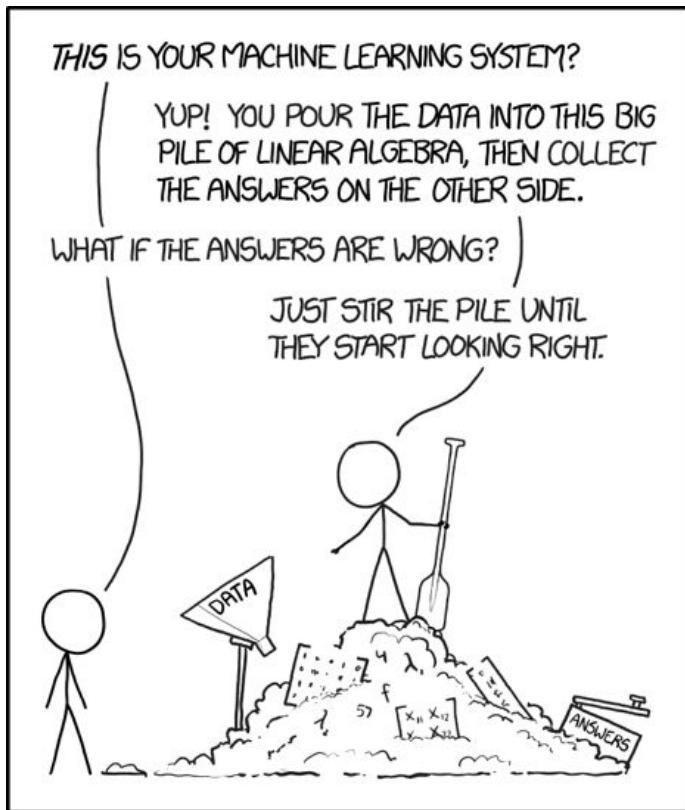


Tipos de aprendizaje



Distintos modelos:

Probabilidad, estadística y álgebra lineal con esteroides



A faint, light-gray network of numerous small circles of varying sizes and thin connecting lines forms a complex web across the entire slide.

2.

Redes neuronales

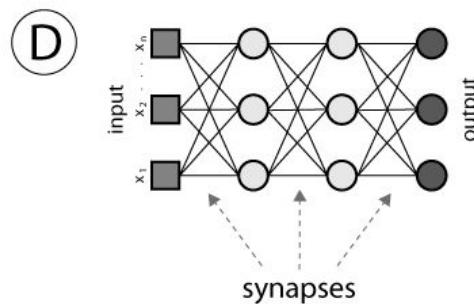
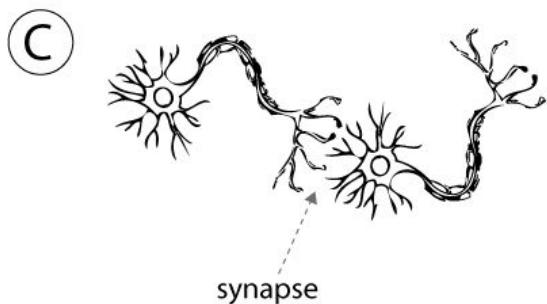
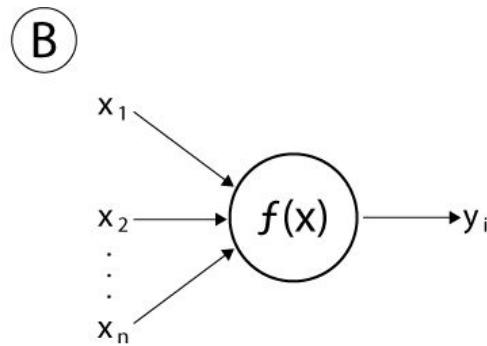
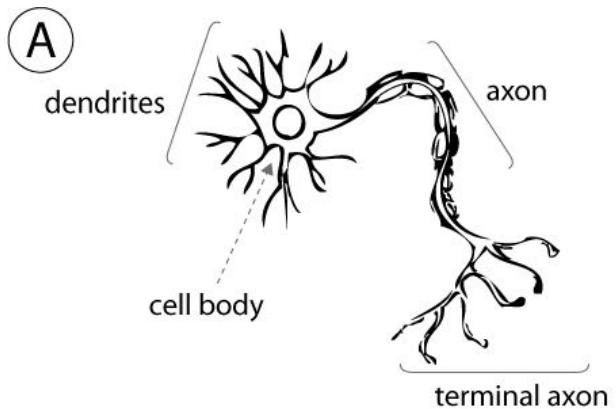
Aprendizaje inspirado en el
funcionamiento cerebral



“

*Las ANNs son modelos computacionales **bio-inspirados** por el sistema nervioso de los seres vivos. Las ANNs pueden ser definidas como un conjunto de unidades de procesamiento (neuronas artificiales), interconectadas por matrices de pesos sinápticos (sinapsis artificiales), que pueden ser implementadas por vectores y matrices de pesos sinápticos (da Silva, Spa , Flauzino, Liboni, & dos Reis Alves, 2017).*

Modelo bio-inspirado



Fuente: da Silva, Spa , Flauzino, Liboni, & dos Reis Alves, 2017.

Estructura

Una ANN puede ser dividida en tres partes llamadas capas, que son conocidas como:

1. *Input layer (capa de entrada)*

Esta capa recibe información, la cual puede ser señales, características, imágenes crudas o medidas del entorno. Estos inputs usualmente son normalizados. Esta normalización tiene como resultado mejores resultados de precisión numérica en las operaciones realizadas por la red.

Estructura

Una ANN puede ser dividida en tres partes llamadas capas, que son conocidas como:

2. *Hidden layers (capas ocultas)*

Estas capas están compuestas por neuronas que responden extrayendo patrones asociados con el proceso o sistema analizado. Estas capas realizan la mayoría del procesamiento interno de una ANN.

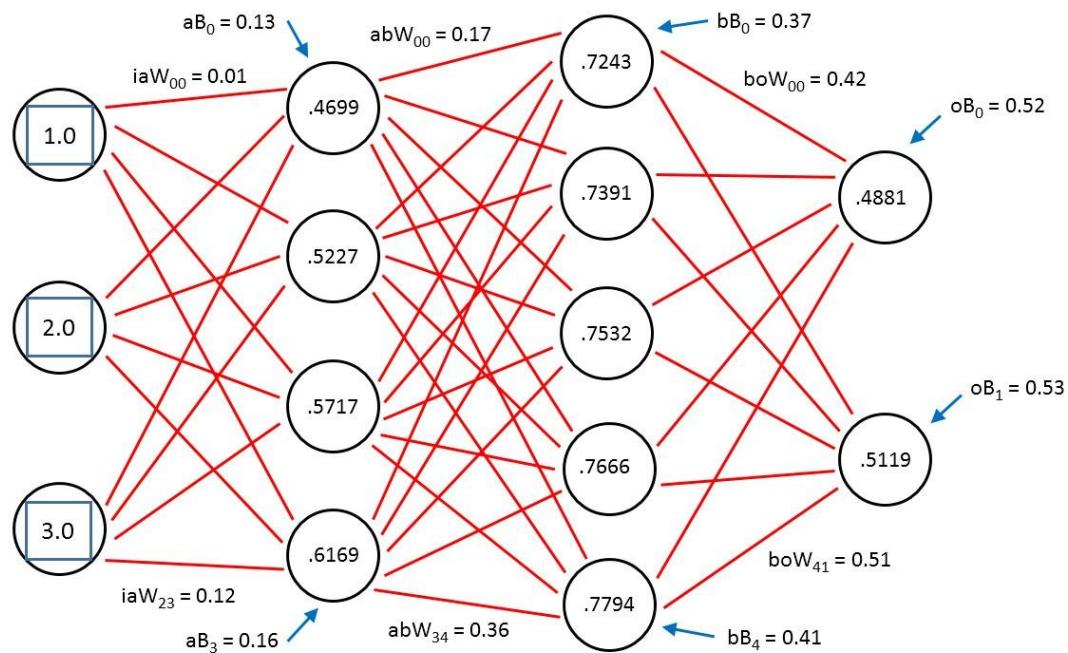
Estructura

Una ANN puede ser dividida en tres partes llamadas capas, que son conocidas como:

3. *Output layer (capa de salida)*

Esta capa compuesta también de neuronas es la responsable de producir y presentar las salidas finales de la red, que resultan del procesamiento realizado por las capas previas.

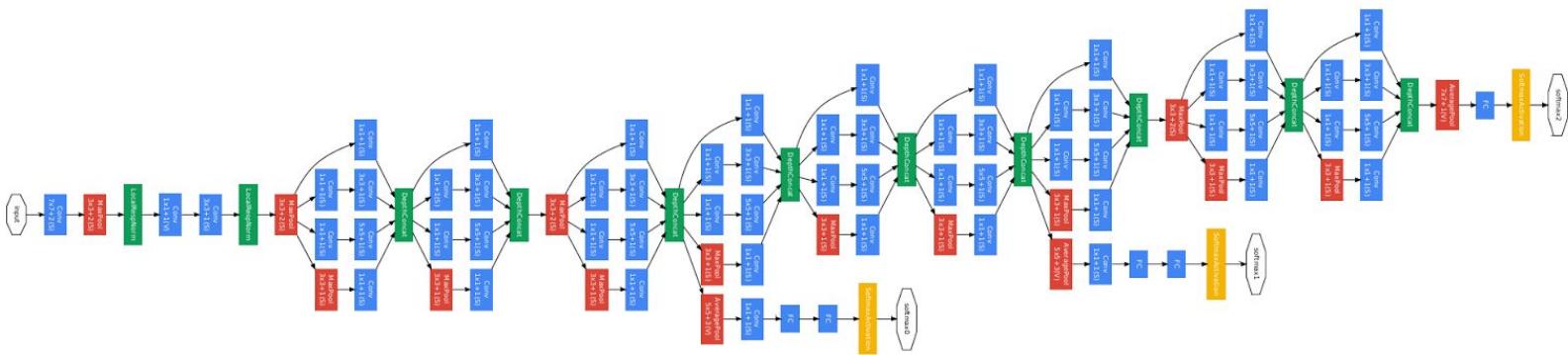
Capas, como las cebollas:



Fuente: “DEEP NEURAL NETWORKS: A GETTING STARTED TUTORIAL, PART #1”, Sergey Golubev, 2014.



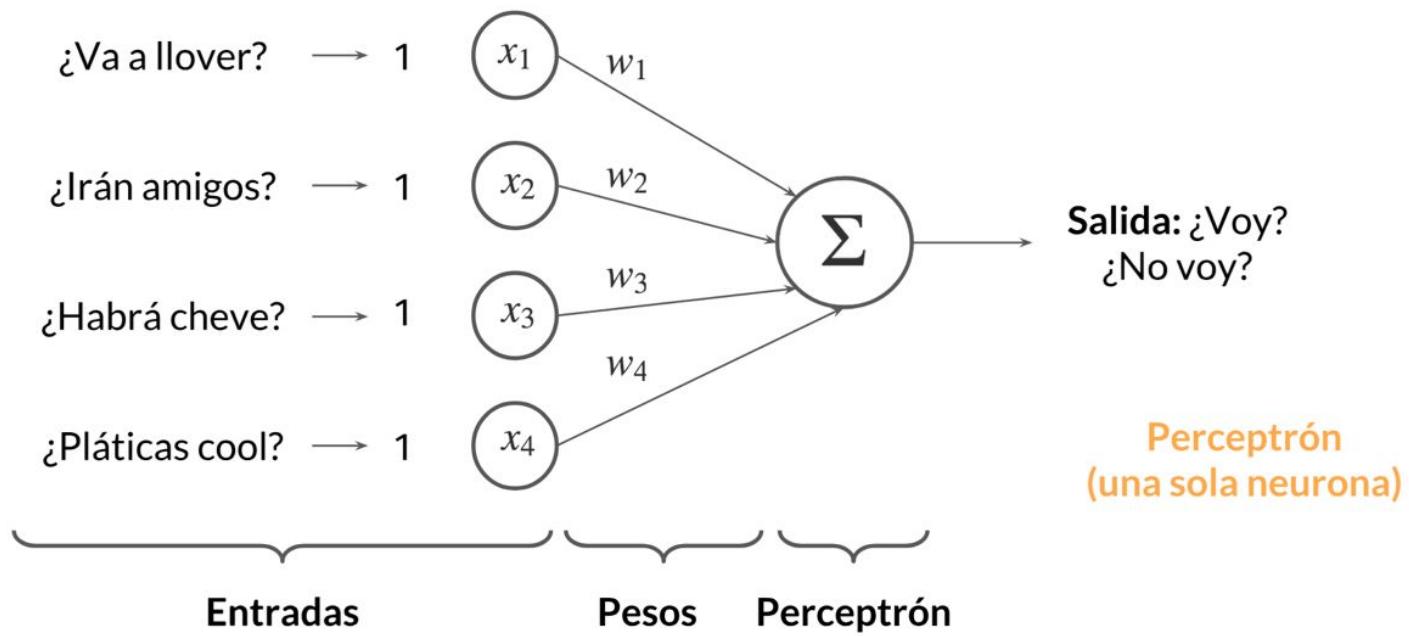
Capas, como las cebollas:



Fuente: “Going Deeper with Convolutions”,
Szegedy et al., 2014.



¿Voy al CONISOFT 2018?



¿Cómo se programa?

```
import numpy as np

class Perceptron():
    def __init__(self, entradas, pesos):
        """Constructor de la clase."""
        self.n = len(entradas)
        self.entradas = np.array(entradas)
        self.pesos = np.array(pesos)

    def voy_no_voy(self, umbral):
        """Calcula el output deseado."""
        si_no = (self.entradas @ self.pesos) >= umbral
        if si_no: return "Sí voy."
        else: return "No voy."
```

```

import numpy as np

class SigmoidNeuron():
    def __init__(self, n):
        np.random.seed(123)
        self.synaptic_weights = 2 * np.random.random((n, 1)) - 1

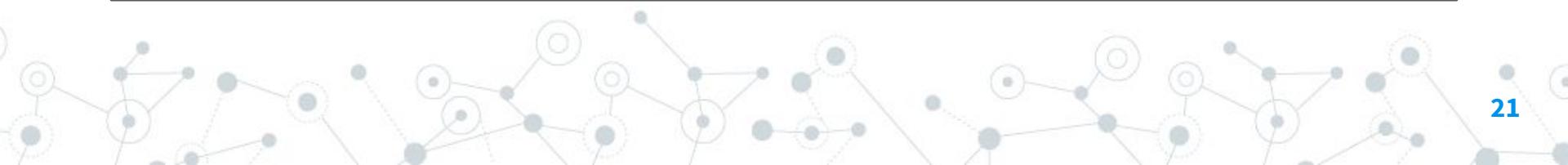
    def __sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def __sigmoid_derivative(self, x):
        return x * (1 - x)

    def train(self, training_inputs, training_output, iterations):
        for iteration in range(iterations):
            output = self.predict(training_inputs)
            error = training_output.reshape((len(training_inputs), 1)) - output
            adjustment = np.dot(training_inputs.T, error *
                                self.__sigmoid_derivative(output))
            self.synaptic_weights += adjustment

    def predict(self, inputs):
        return self.__sigmoid(np.dot(inputs, self.synaptic_weights))

```



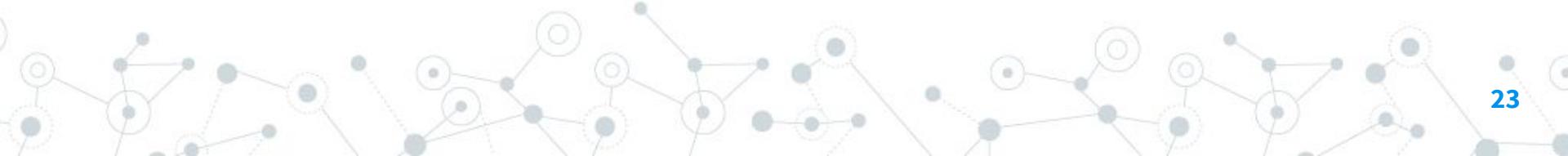
La magia detrás:

Operaciones que
ayudan a
comprender la
esencia de las cosas.

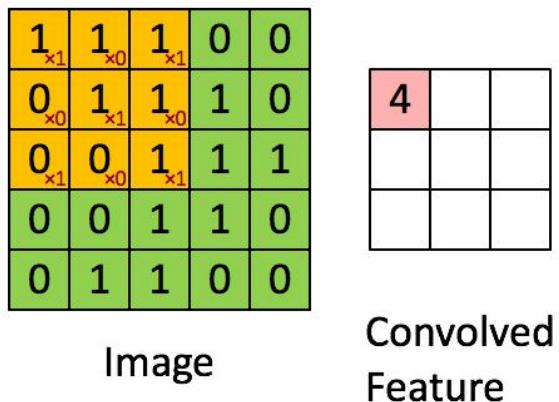


Operaciones mágicas:

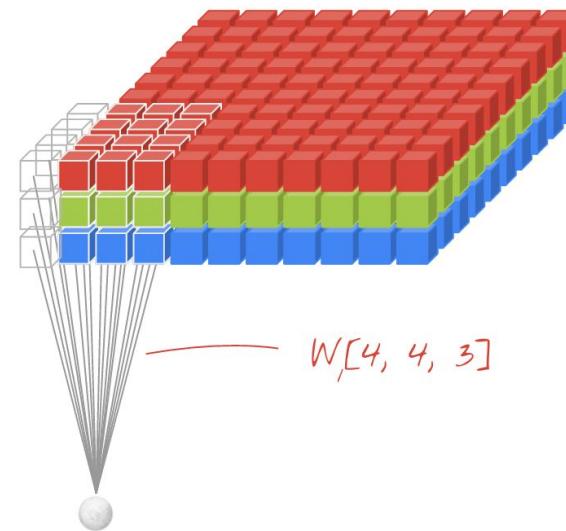
$$\begin{aligned}(f * g)(t) &\stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau \\&= \int_{-\infty}^{\infty} f(t - \tau)g(\tau) d\tau.\end{aligned}$$



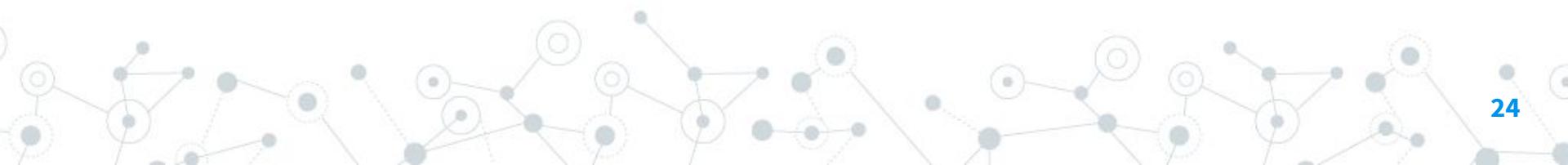
Operaciones mágicas:



Fuente: “Convolution sliding window”, NVIDIA Blog.



Fuente: “2D Convolutional Animation”, Martin Görner.



Operaciones mágicas:

“Representation Learning”.

Representación = Feature

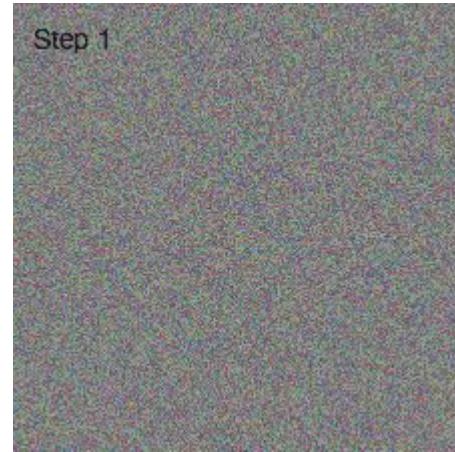


Input - Una imagen

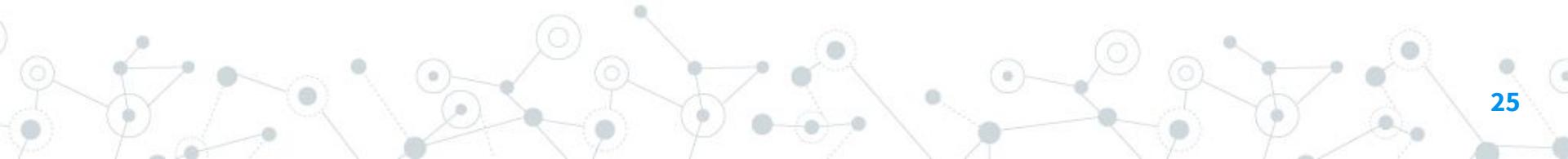
Representación - No hay esquinas

Modelo - Crea una representación y aplica la regla para detectar la forma.

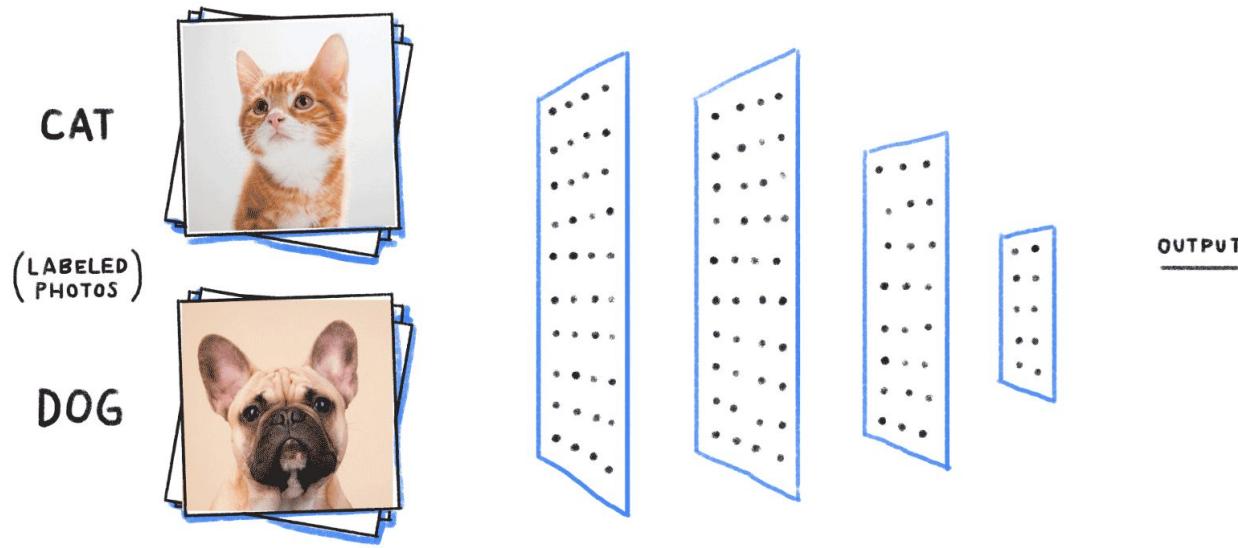
Fuente: “What is representation learning in deep learning?”, Quora.



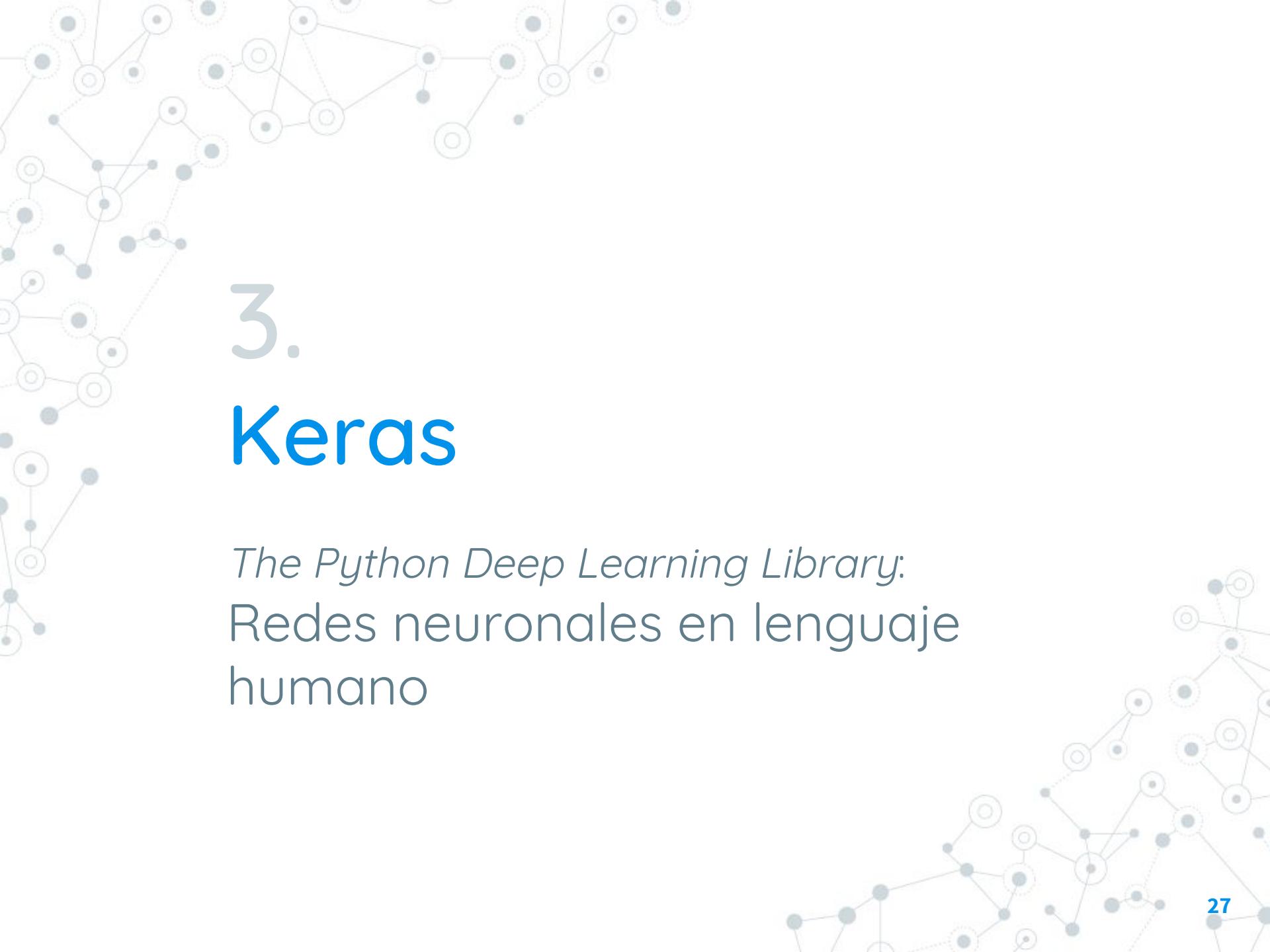
Fuente: “Keras Visualization Toolkit”, Raghavendra Kotikalapudi.



Operaciones mágicas:



Fuente: "Simple Image Classification using Convolutional Neural Network—Deep Learning in python.", Venkatesh Tata.



3. Keras

The Python Deep Learning Library:
Redes neuronales en lenguaje
humano

François Chollet



A screenshot of François Chollet's Twitter profile. It features a circular profile picture of him with glasses and dark hair, set against a blue background. To his right is a large, abstract, glowing green sphere composed of many smaller nodes and lines, set against a dark background. A quote from Arthur Schopenhauer is visible on the right side of the sphere: "what we see now is like a dim image in a mirror". Below the image are three icons: a gear, a bell with a plus sign, and a blue button labeled "Following".

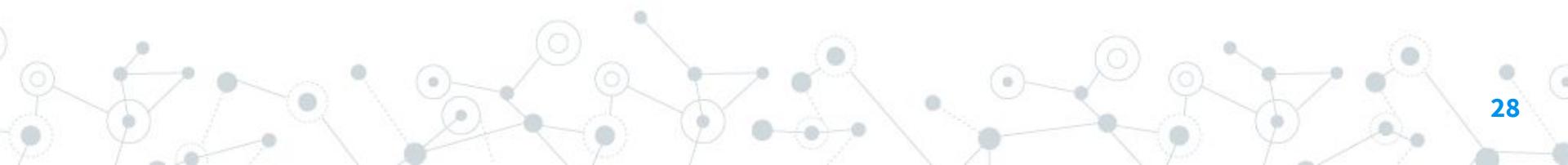
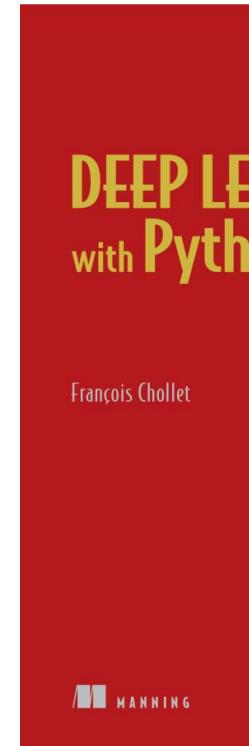
François Chollet 

@fchollet

Deep learning [@google](#). Creator of Keras, neural networks library. Author of "Deep Learning with Python". All opinions are my own (strong but weakly held).

📍 Mountain View, CA 🌐 [keras.io](#)

565 Following **106.4K Followers**





François Chollet

@fchollet

Following



Photoshop is basically like deep learning:
just add more layers

6:02 PM - 9 Feb 2018

140 Retweets 625 Likes



20



140



625



Tweet your reply



Max Woolf @minimaxir · Feb 9

Replying to @fchollet

Deep learning is Shrek cuz it has layers



2



2



19

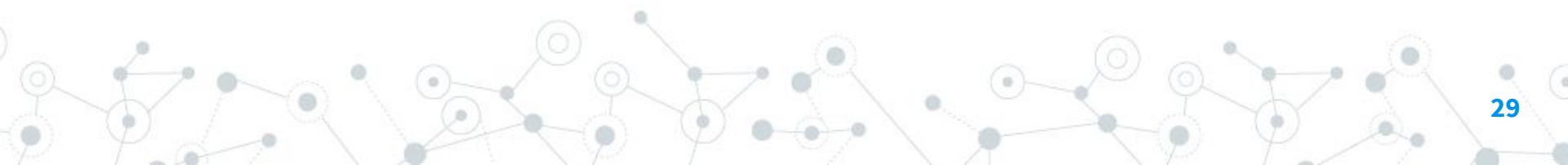


Les Guessing @LesGuessing · Feb 10

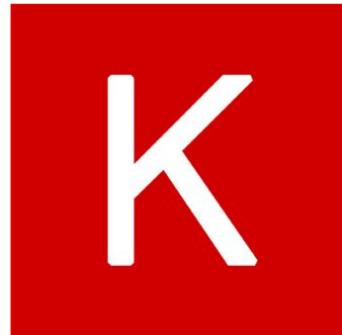
Everybody loves a parfait.



Fuente: <https://twitter.com/fchollet/status/962144774802227205>



Keras: The Python Deep Learning library

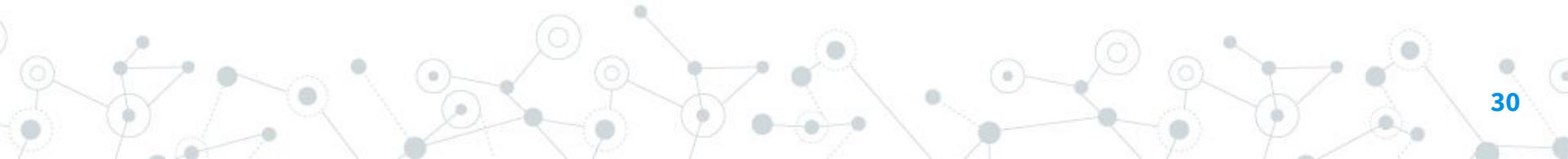


Keras

You have just found Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of [TensorFlow](#), [CNTK](#), or [Theano](#). It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

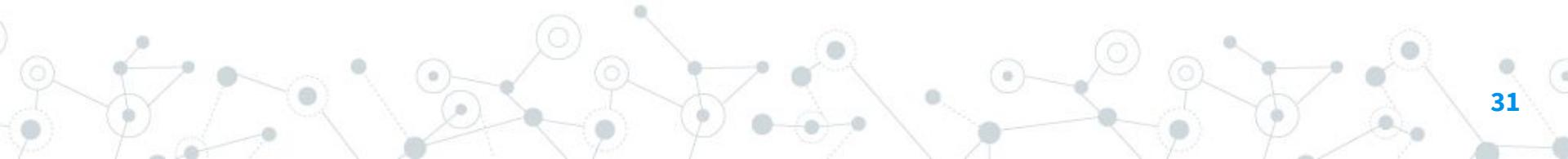
Source: <https://keras.io/>



Guiding principles

- **User friendliness.** Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.
- **Modularity.** A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.
- **Easy extensibility.** New modules are simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.
- **Work with Python.** No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

Source: <https://keras.io/>



Keras en 1 minuto:

La estructura principal de Keras es un modelo, que es una manera en la que organizamos las capas. El modelo más sencillo es el modelo **Sequential**, una pila lineal de capas. Para modelos más elaborados, usamos **Model**, con la programacion funcional del API de Keras.

```
# Import the Sequential model:  
from keras.models import Sequential  
  
# Create a new Sequential model:  
model = Sequential()
```

Keras en 1 minuto:

Añadir capas es tan sencillo como utilizar `.add()`.

```
# Import the Dense Layer class for FC:  
from keras.layers import Dense  
  
# Add FC Layers:  
model.add(Dense(units=64, activation='relu', input_dim=100))  
model.add(Dense(units=10, activation='softmax'))
```

Keras en 1 minuto:

Una vez se tenga el modelo, configuramos su proceso de aprendizaje utilizando `.compile()`.

```
# Compile model:  
model.compile(loss='categorical_crossentropy',  
              optimizer='sgd',  
              metrics=['accuracy'])
```

Keras en 1 minuto:

Una vez compilado el modelo, puede procederse a iterar en los lotes de entrenamiento usando `.fit()`.

```
# Train the model:  
# x_train and y_train are Numpy arrays  
# --just like in the Scikit-Learn API.  
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

Keras en 1 minuto:

Finalmente se puede evaluar el desempeño, o generar predicciones de nuevos datos:

```
# Evaluate the model:  
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)  
  
# Make predictions:  
classes = model.predict(x_test, batch_size=128)
```

Keras Layers:

- Core:
 - Dense
 - Activation
 - Dropout
 - Flatten
 - Input
 - Reshape
 - Permute
 - ...
- Convolutional:
 - Conv1D
 - Conv2D
 - Conv3D
 - ...
- Pooling:
 - MaxPooling1D
 - MaxPooling2D
 - MaxPooling3D
 - AveragePooling1D
 - AveragePooling2D
 - AveragePooling3D
 - GlobalMaxPooling1D
 - ...
- Locally-connected
- Recurrent
- Embedding
- Merge
- Normalization
- ...

Keras Preprocessing:

- Sequence preprocessing
- Image preprocessing
- Text preprocessing

...

*Metrics, Losses, Activations, Optimizers,
Initializers, Regularizers, Datasets...*

¿Y si quisiera hacer un módulo *inception*?

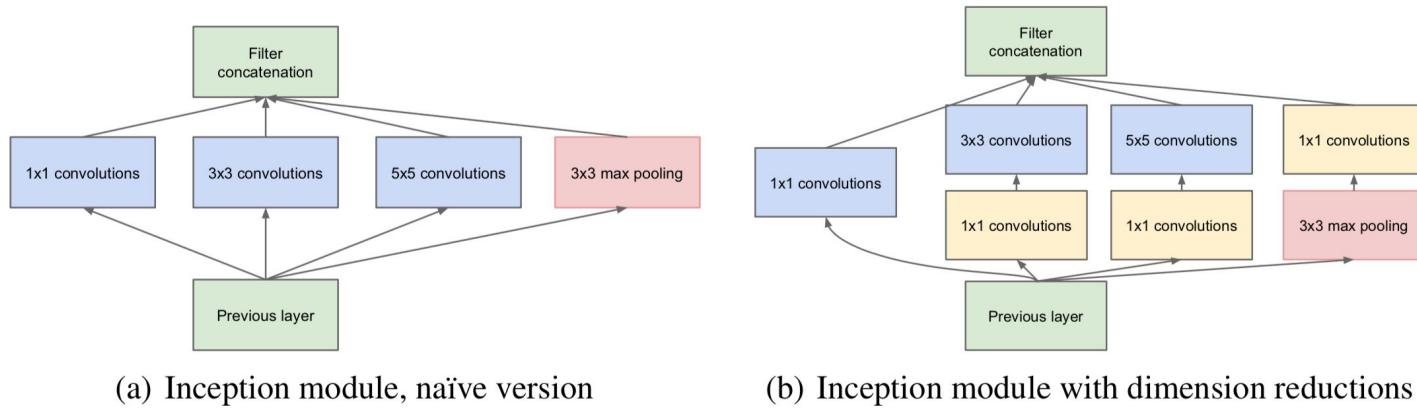
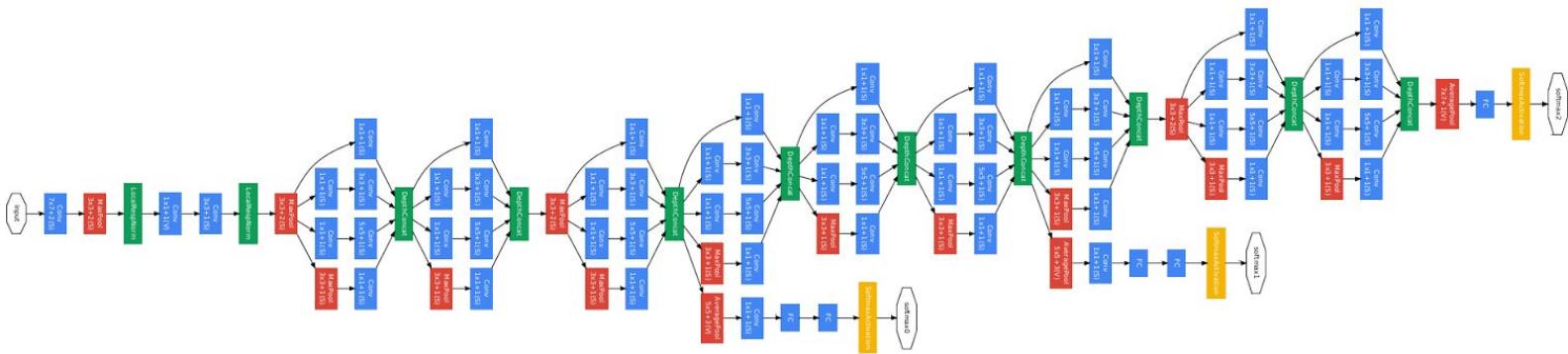


Figure 2: Inception module

Fuente: “ Going Deeper with Convolutions”, Szegedy *et al.*, 2014.

Capas, como las cebollas:



Fuente: “Going Deeper with Convolutions”,
Szegedy et al., 2014.



Keras Functional API:

La API funcional de Keras es el camino a seguir para definir modelos complejos, como modelos de múltiples salidas, gráficos acíclicos dirigidos o modelos con capas compartidas.

Keras Functional API:

```
from keras.layers import Conv2D, MaxPooling2D, Input

input_img = Input(shape=(256, 256, 3))

tower_1 = Conv2D(64, (1, 1), padding='same', activation='relu')(input_img)
tower_1 = Conv2D(64, (3, 3), padding='same', activation='relu')(tower_1)

tower_2 = Conv2D(64, (1, 1), padding='same', activation='relu')(input_img)
tower_2 = Conv2D(64, (5, 5), padding='same', activation='relu')(tower_2)

tower_3 = MaxPooling2D((3, 3), strides=(1, 1), padding='same')(input_img)
tower_3 = Conv2D(64, (1, 1), padding='same', activation='relu')(tower_3)

output = keras.layers.concatenate([tower_1, tower_2, tower_3], axis=1)
```

Módulo *inception*:

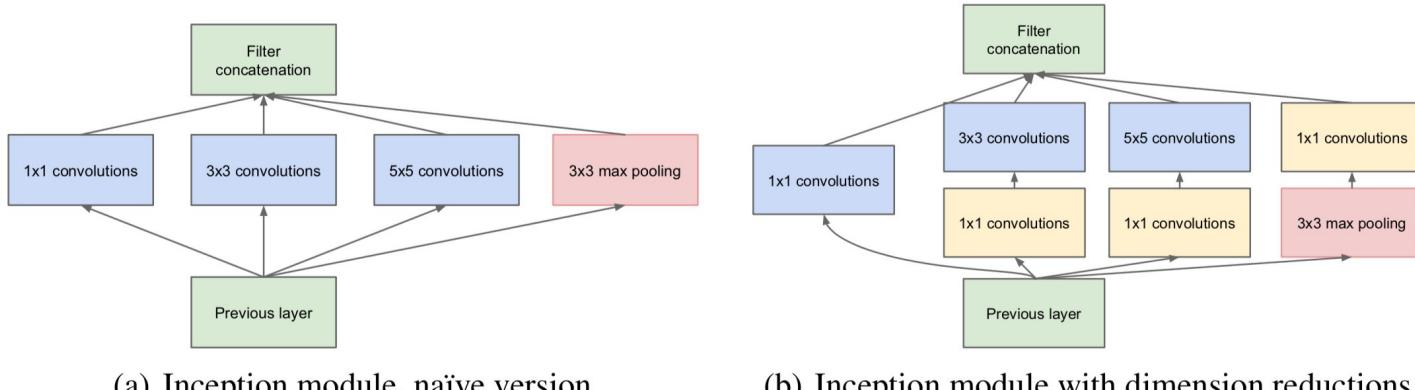


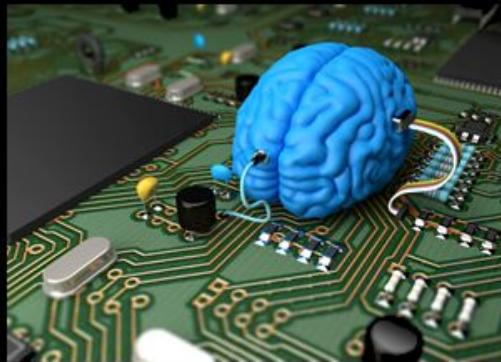
Figure 2: Inception module

Fuente: “ Going Deeper with Convolutions”, Szegedy *et al.*, 2014.

Deep Learning



What society thinks I do



What my friends think I do



What other computer
scientists think I do



What mathematicians think I do



What I think I do

```
In [1]:  
  
import keras  
  
Using TensorFlow backend.
```

What I actually do



4.

Manos a la obra

A escribir código...

Antes de continuar...



<https://github.com/RodolfoFerro/CONISOFT2018>

Fork it, clone it, cut it, paste it, load it, check it, quick, rewrite it...

- *Daft Punk (ft. Rodolfo Ferro)*

Antes de continuar...

The logo for colab, featuring the word "colab" in a bold, sans-serif font. The letters are composed of overlapping colored segments: 'c' is yellow, 'o' is yellow and orange, 'l' is orange, and 'ab' is orange.

<https://colab.research.google.com>



Reto 1:

Completar el notebook.

Reto 2:

Experimentar con la arquitectura dada.

Reto 3:

Experimentar con hiperparámetros.

*Reto 4:

Graficar el entrenamiento.

**Reto 5:

Añadir módulos inception.



30 NOV - 1 DIC, GDL.
<http://pythonday.mx/>

5.

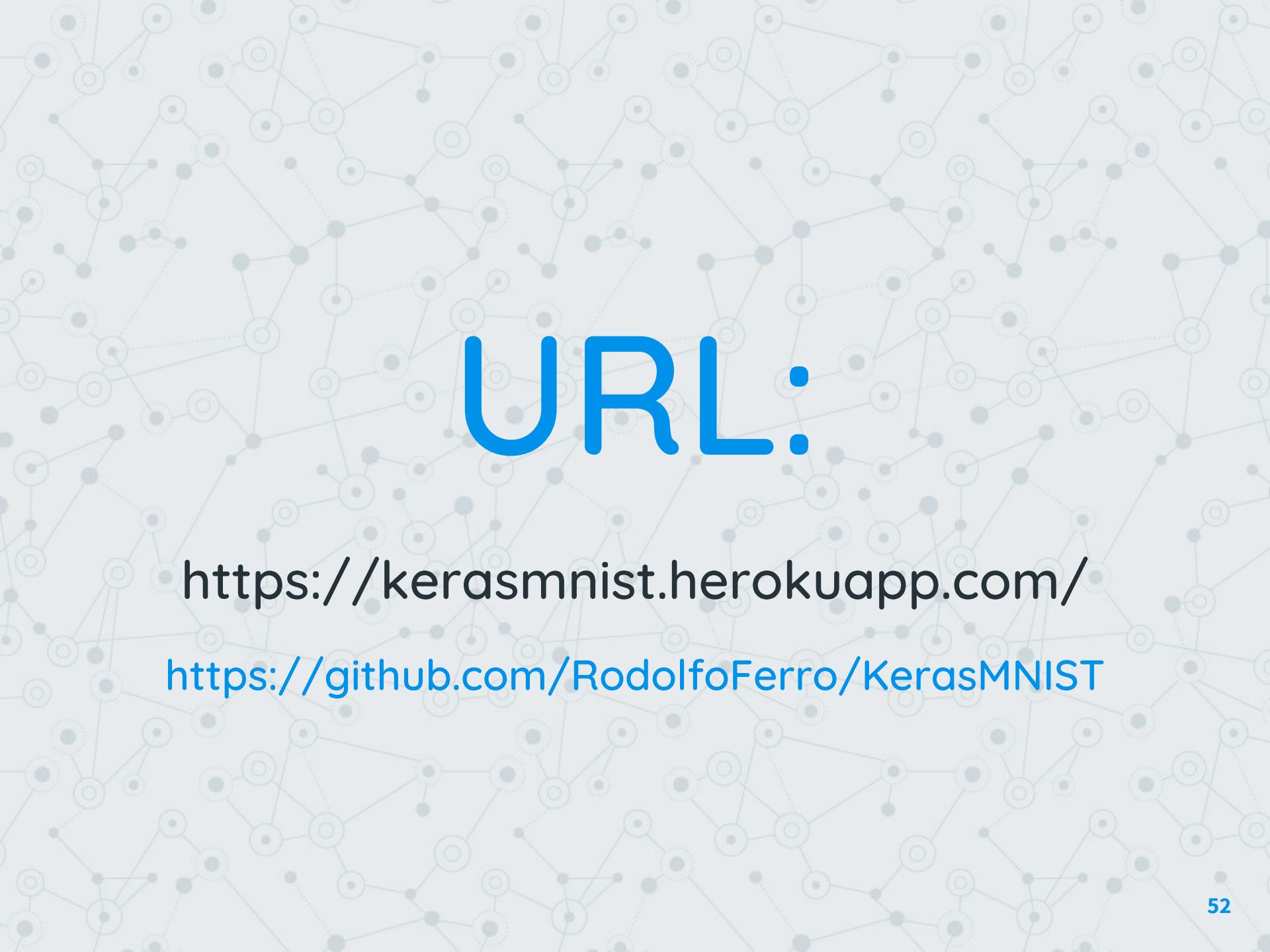
Aplicación

Un ejemplo aplicado a través
de un servicio



Servicio web

Un ejemplo sencillo de aplicación directa para clasificación de dígitos escritos a mano.

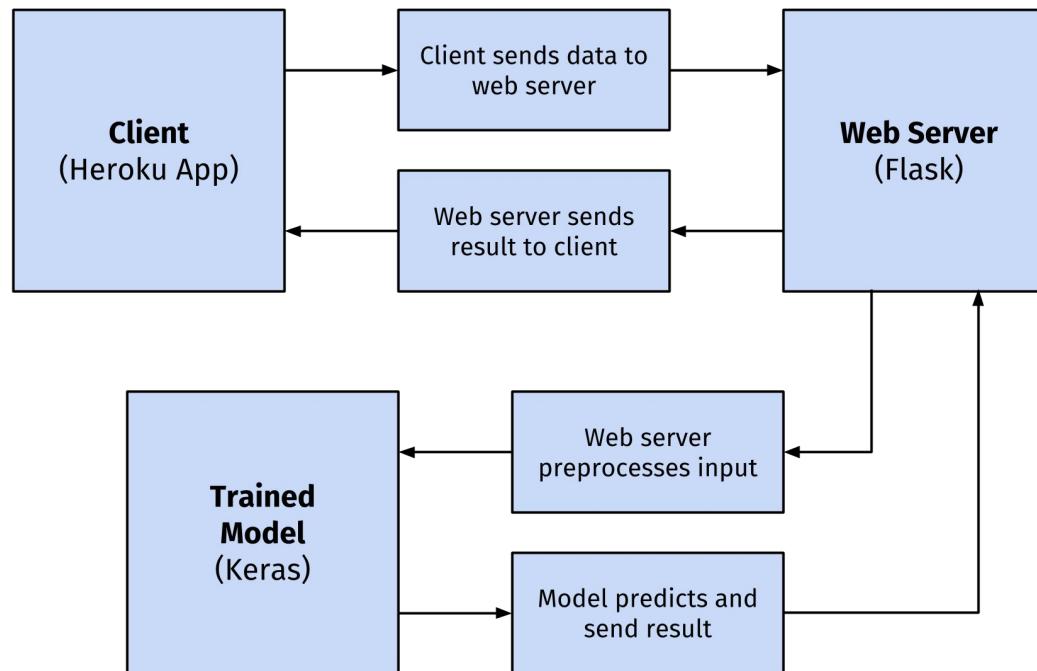


URL:

<https://kerasmnist.herokuapp.com/>

<https://github.com/RodolfoFerro/KerasMNIST>

Sobre el servicio...



¡Gracias!

¿Preguntas?

Encuéntrame:

 @FerroRodolfo

 @rodo_ferro

<https://rodolfoferro.xyz/>

