Curso de

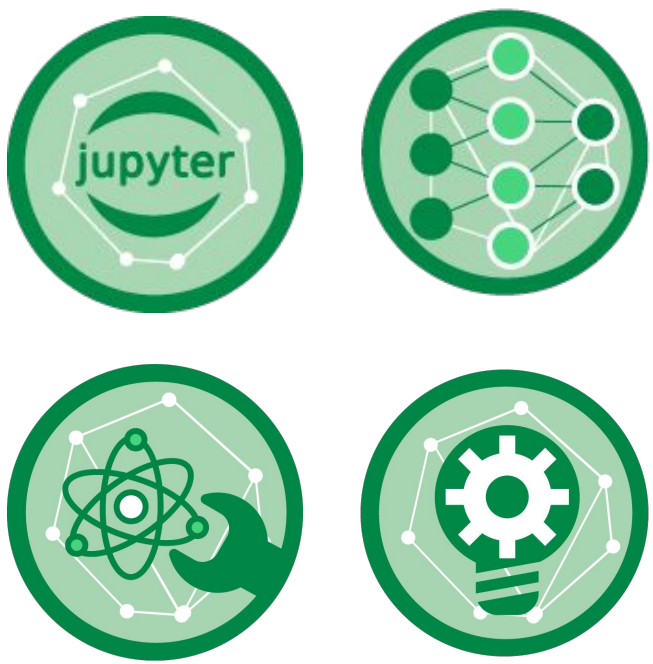
Redes Neuronales con TensorFlow

Adonaí Vera
Co-Founder y CTO en SwitchAI

www.switchai.co

Requisitos del curso

- Fundamentos de redes neuronales.
- Uso de Jupyter Notebooks y Google Colab.
- Creación de proyectos de ciencia de datos e inteligencia artificial.



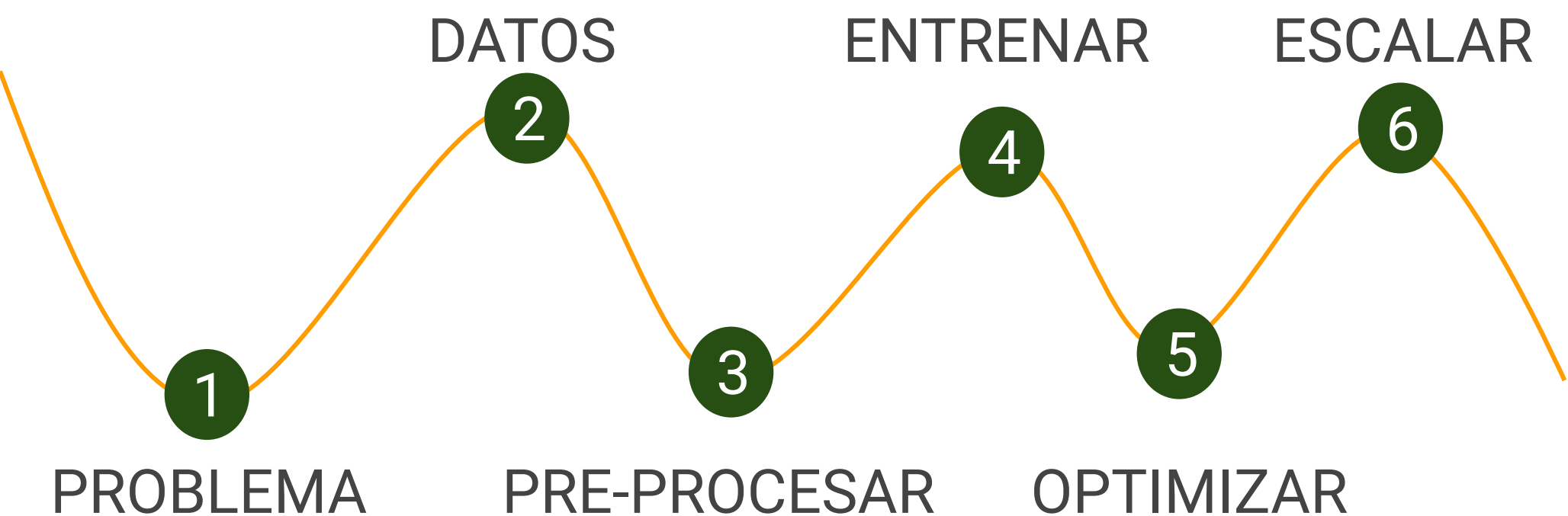
Por qué debes tomar el curso

- Tienes un proyecto de la universidad en donde aplicarás redes neuronales.
- En tu empresa están trabajando con redes neuronales y te asignaron el nuevo proyecto.
- Curioso por nuevas tecnologías.

Por qué debes tomar el curso

- Quieres aprender cómo aplicar inteligencia artificial a imágenes.
- Tienes las bases pero quieres volverte un pro configurando redes neuronales.
- Quieres crear tu propio tu propio modelo.

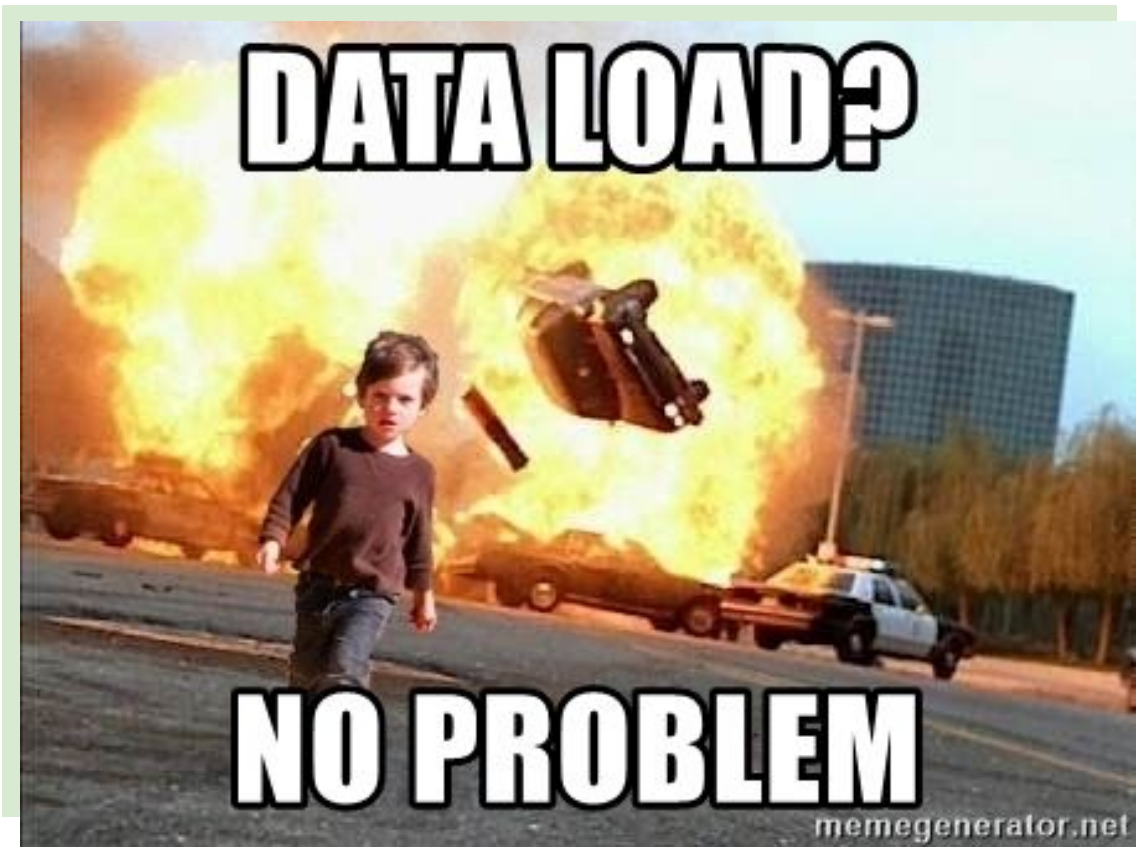
Ciclo de la IA



Redes neuronales

Llevar redes neuronales a la práctica.

Cargar bases de datos



Optimizar modelos

¿Cómo evitar el overfitting y underfitting?

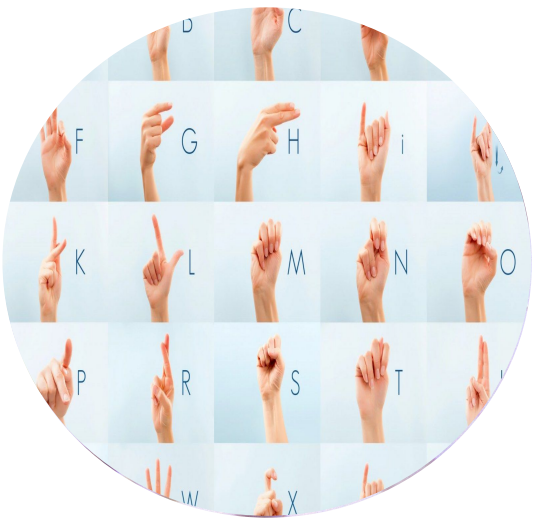
Transferencia de aprendizaje

Transfer learning be like

Almacenar y cargar modelos

Proyecto del curso

- 27455 imágenes
- Escala de grises
- 28 x 28
- 24 clases (J - Z)
- JPEG
- TecPerson - Kaggle
- C0: Public Domain



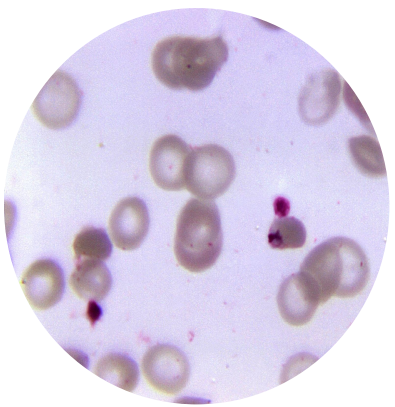
Lenguaje de señas

Proyectos

Tom & Jerry

Tumor

Lenguaje de Señas



Malaria

```
# Bases de datos Malaria
!wget --no-check-certificate \
    https://storage.googleapis.com/platzi-tf2/MalariaCells.zip \
    -O /tmp/sign-language-img.zip


# Bases de datos Tumor
!wget --no-check-certificate \
    https://storage.googleapis.com/platzi-tf2/TumorClassification.zip \
    -O /tmp/sign-language-img.zip

# Bases de datos Tom y Jerry
!wget --no-check-certificate \
    https://storage.googleapis.com/platzi-tf2/TomAndJerry.zip \
    -O /tmp/sign-language-img.zip
```

¿Qué aprenderás?

- Cómo cargar tus propias bases de datos.
- Cargar bases de datos en formatos como CSV, JSON, BASE64, imágenes.
- Aplicar técnicas para optimizar tus modelos.

¿Qué aprenderás?

- Agregar métricas en el entrenamiento de tus modelos.
- A cargar y guardar modelos.
- Autotunner de Keras para encontrar las mejores variables.

¿Qué aprenderás?

- Bases de aprendizaje por transferencia.
- Uso de TensorBoard y cómo mostrar tu proyecto al mundo entero.
- A tener tu modelo listo para utilizarlo como inferencia.

Introducción a TF2.0

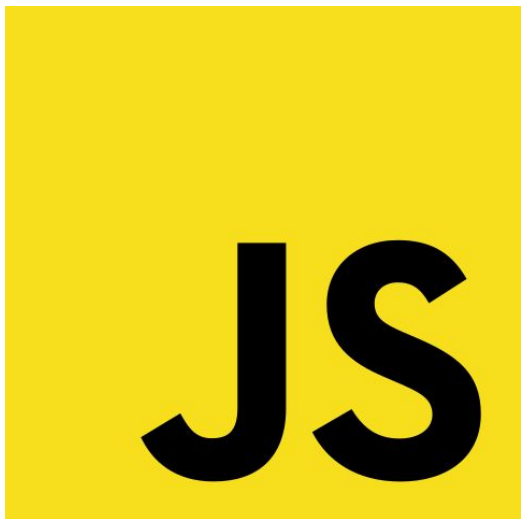
Librerías de deep learning

Caffe

Microsoft
CNTK

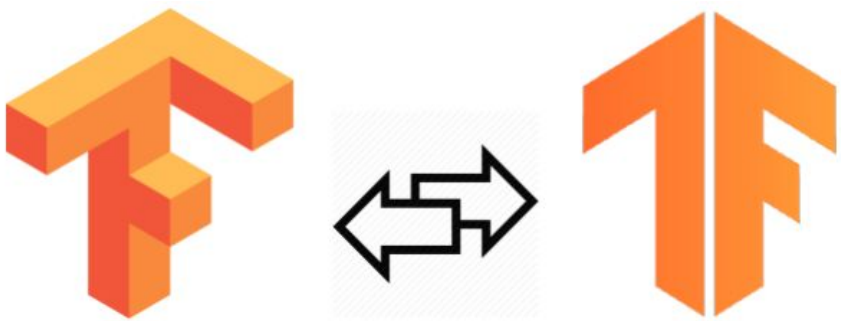
GLUON

Lenguajes de programación



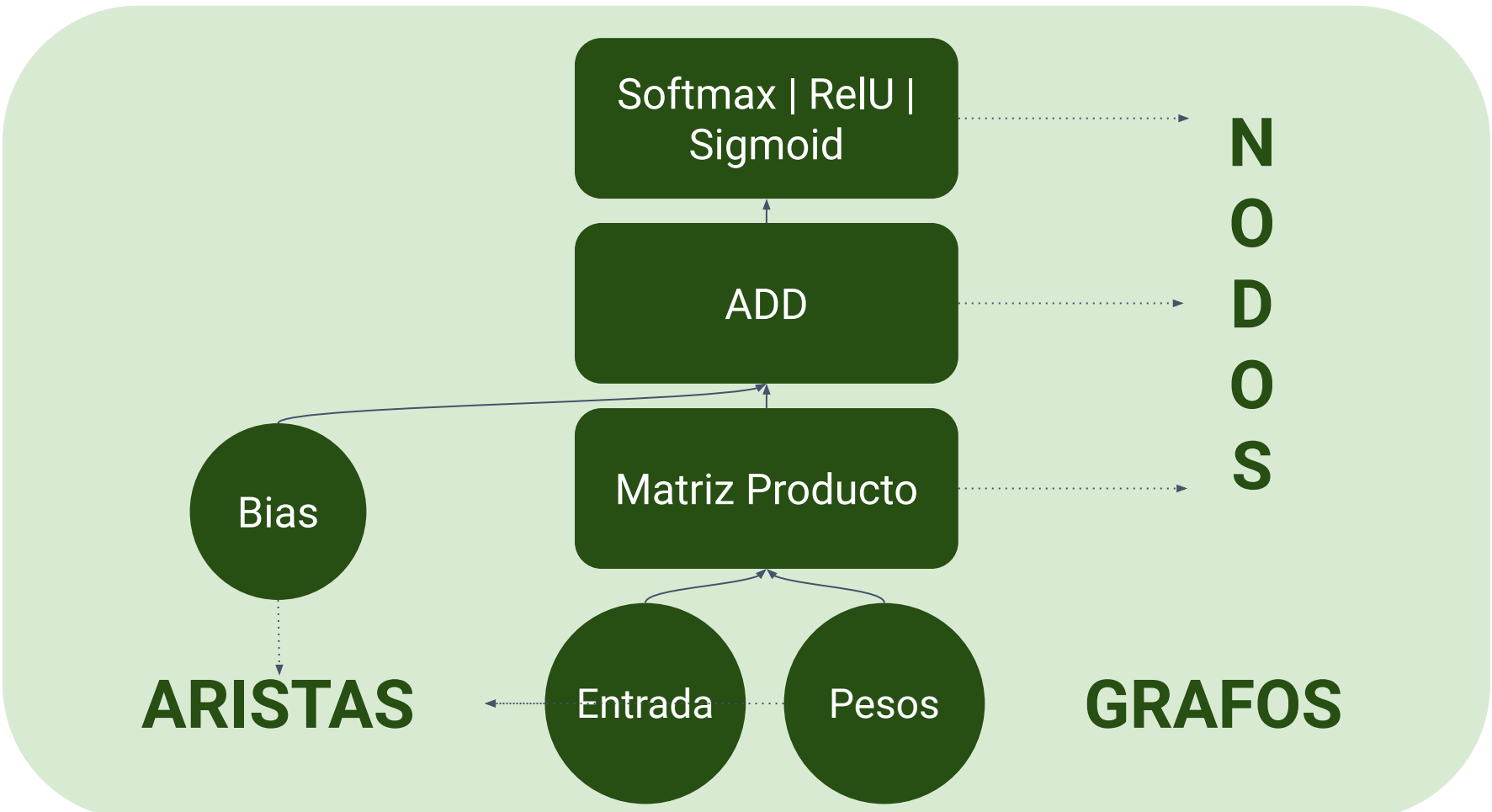
TensorFlow 2 vs. TensorFlow1

Symbolic | Imperative API



- API CLEAN UP
- NO MORE GLOBAL FUNCTIONS
- NOT SESSIONS

Lógica de TensorFlow



Diferencias en código

```python
import tensorflow as tf
import tensorflow.compat.v1 as v1
import tensorflow_datasets as tfds

g = v1.Graph()

with g.as_default():
  in_a = v1.placeholder(dtype=v1.float32, shape=(2))
  in_b = v1.placeholder(dtype=v1.float32, shape=(2))

  def forward(x):
    with v1.variable_scope("matmul", reuse=v1.AUTO_REUSE):
      W = v1.get_variable("W", initializer=v1.ones(shape=(2,2)),
                          regularizer=lambda x:tf.reduce_mean(x**2))
      b = v1.get_variable("b", initializer=v1.zeros(shape=(2)))
      return W * x + b

  out_a = forward(in_a)
  out_b = forward(in_b)
  reg_loss=v1.losses.get_regularization_loss(scope="matmul")

with v1.Session(graph=g) as sess:
  sess.run(v1.global_variables_initializer())
  outs = sess.run([out_a, out_b, reg_loss],
                  feed_dict={in_a: [1, 0], in_b: [0, 1]})
```

```python
import tensorflow as tf
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test,  y_test, verbose=2)
```

TensorFlow + Keras

Código abierto
Google
2015
Escrita en Python
API alto nivel


El ecosistema de TensorFlow
ARDUINO
Google Cloud
API
TensorFlow
Extended
TensorFlow
Lite
TensorFlow
.JS
Flask

Comunidad de TF

- **56,000** preguntas en StackOverflow.
- **109** ML Google Developer Experts.
- **46** TensorFlow user groups en todo el mundo.
- **25** publicaciones no blog de TensorFlow.
- **~ 70,000** commits.
- **> 2,200** contribuyentes.

Comunidad de TF

Parte de **Google ML Community – 25 grupos**

TensorFlow Colombia

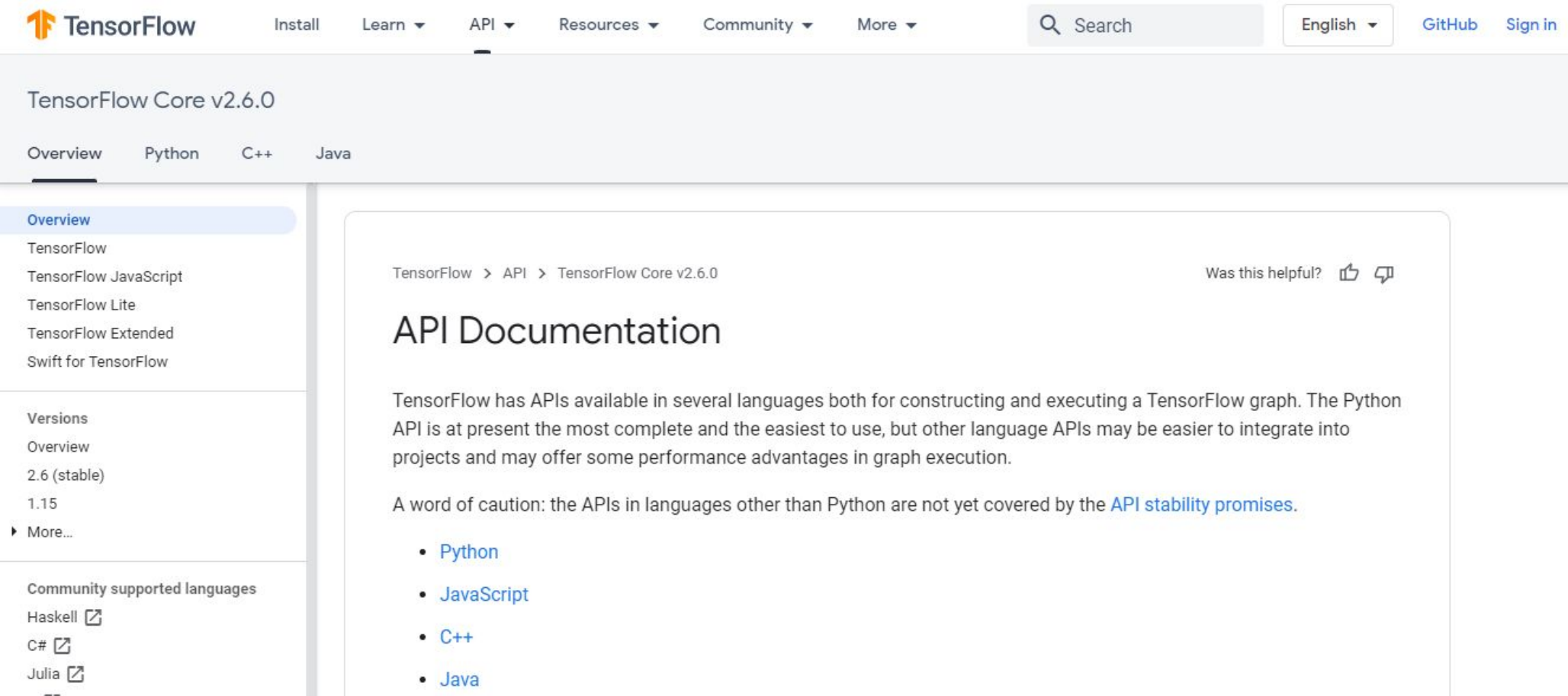
Medellín, Colombia

1677 miembros · Grupo público

Organizado por **Yeis Taborda H.** y otras 4 personas

Compartir:

Documentación adicional



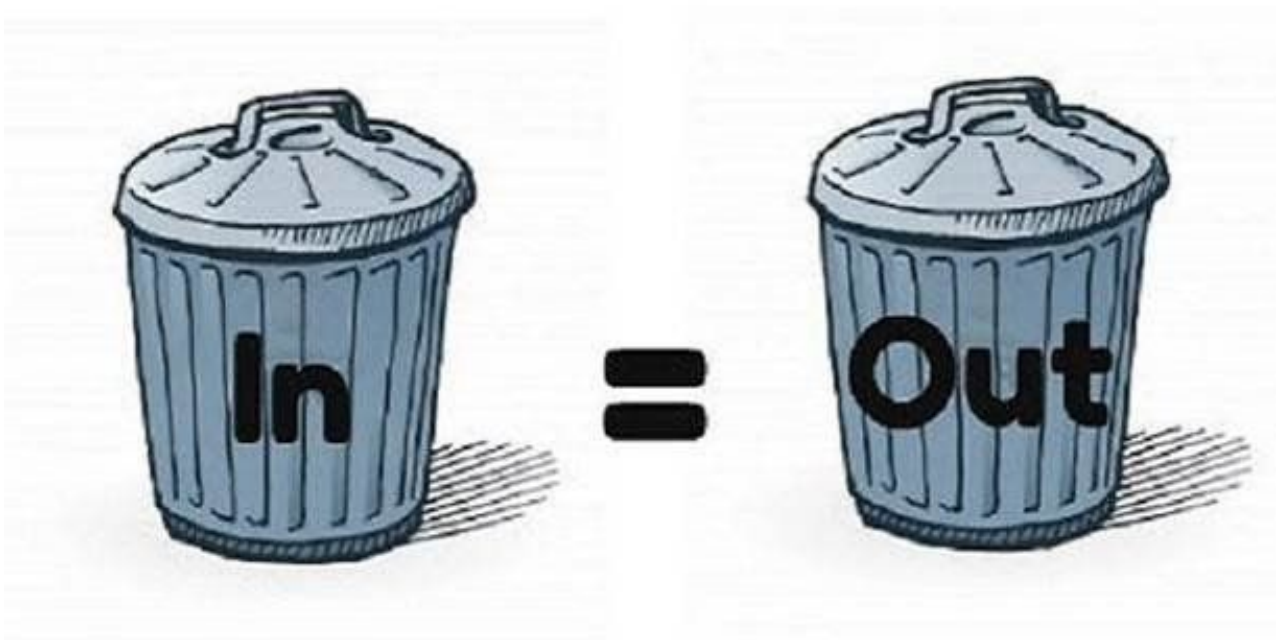
Uso de data pipelines

En este módulo aprenderemos

- Cómo cargar bases de datos en formatos como CSV, JSON, Base64.
- Pre-procesar los datos.
- Cómo cargar dataset de Keras.
- Dataset generators.
- Cómo cargar tu propios dataset con TF.data.
- Cómo distribuir los datos.

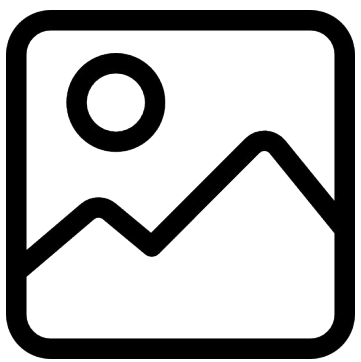
Motor de inteligencia artificial

Basura que entra
= Basura que sale

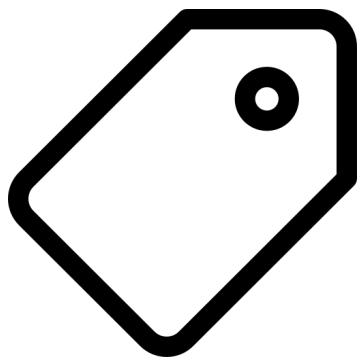


Pre-procesar la información

¿Qué compone un dataset?



Imágenes



Labels

Cargar bases de datos JSON con URL a la nube de GCP

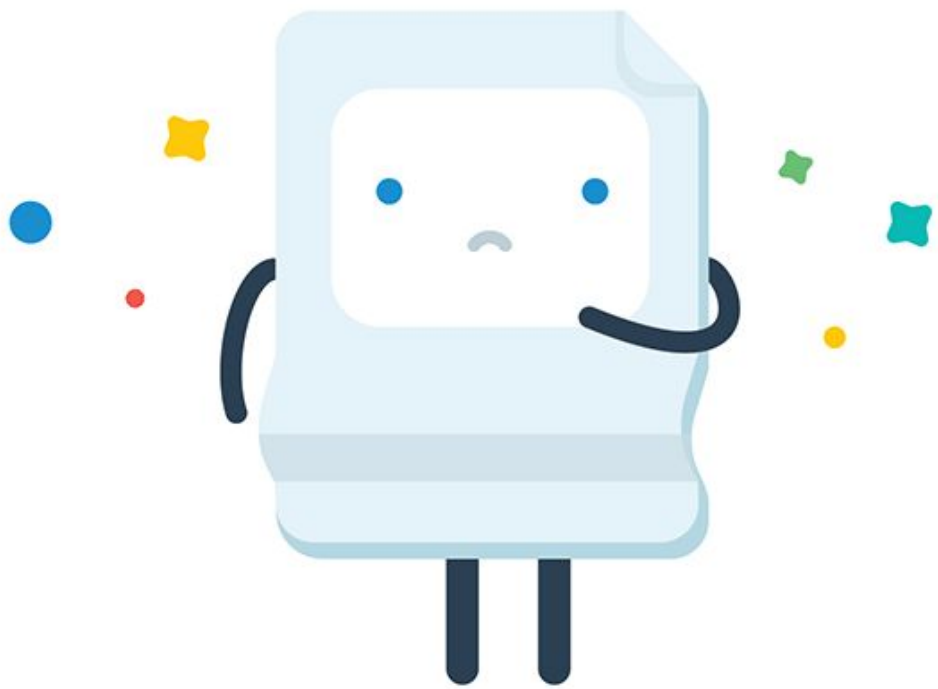
Cargar bases de datos Json Base64 y .CSV

Preprocesamiento de datos

Keras datasets

Dataset Generators

Memoria limitada



Generator en Python

Es una función que devuelve un objeto, el cual te permite iterar en el mismo, sin almacenarlos a la vez en memoria.

Aprende a buscar bases de datos

¿Donde buscar?

kaggle

Google Dataset Search Beta

Cómo distribuir los datos

Distribución de bases de datos

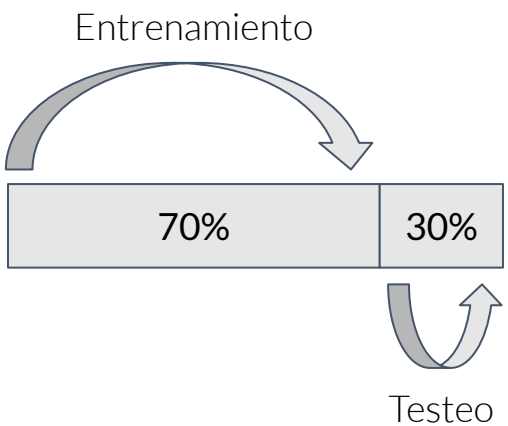
TRAINING

VALIDATION

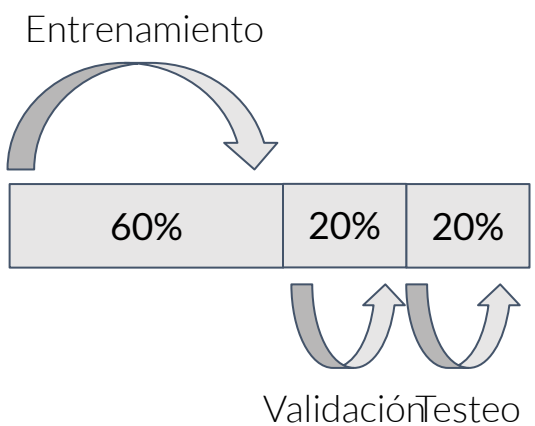
TEST

Distribución de los datos según casos

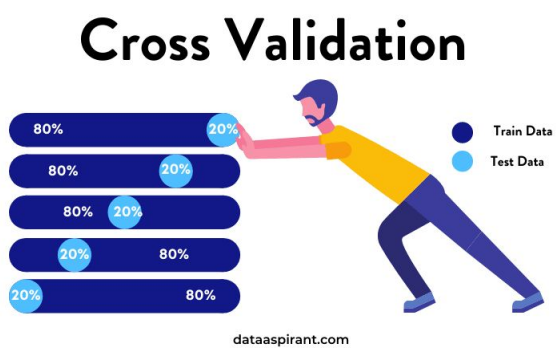
CONFIGURACIÓN PROMEDIO



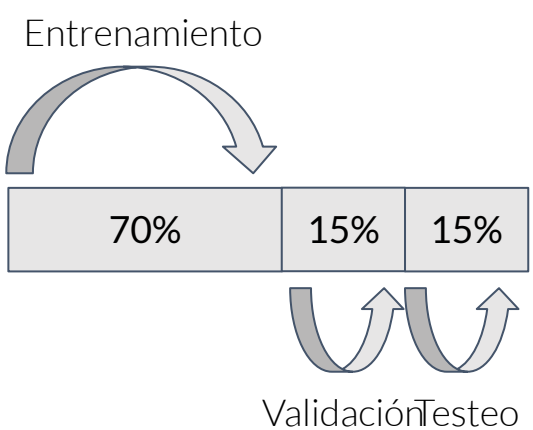
ANDREW NG



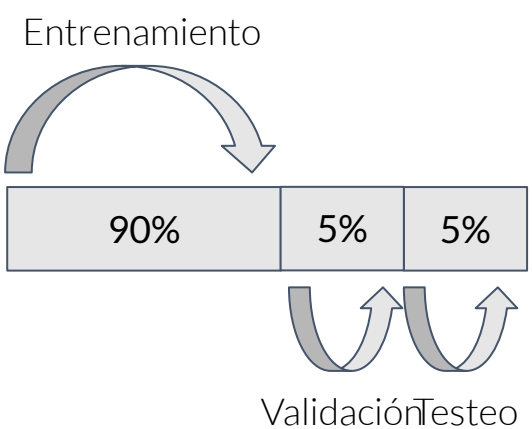
POCOS DATOS



CONFIGURACIÓN RECOMEDADA



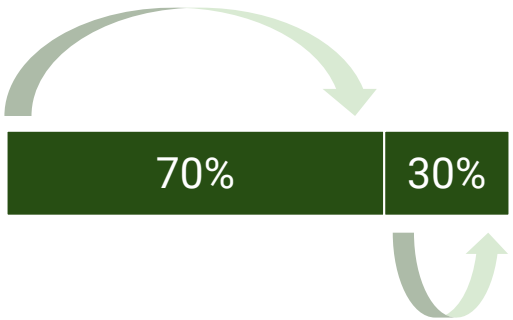
MUCHOS DATOS



Distribución de los datos según casos

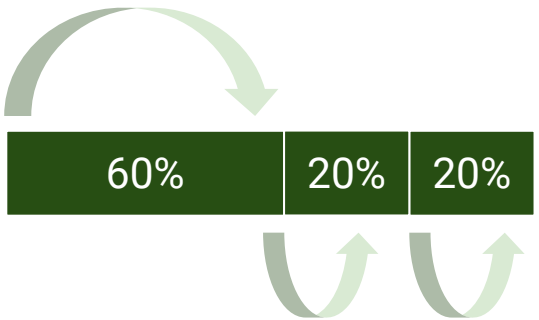
CONFIGURACIÓN PROMEDIO

Entrenamiento

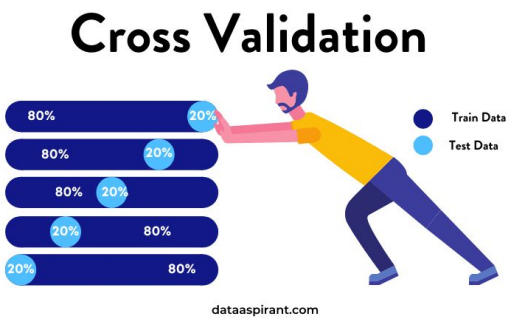


ANDREW NG

Entrenamiento

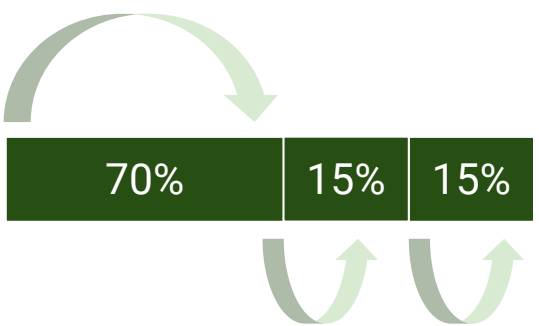


POCOS DATOS



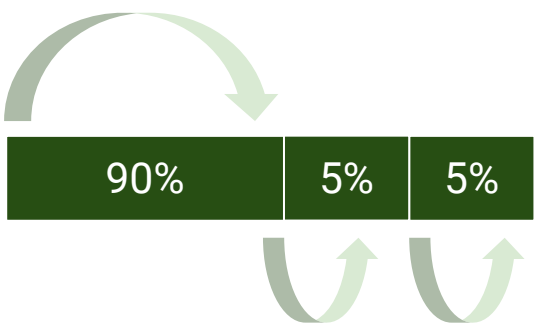
CONFIGURACIÓN RECOMEDADA

Entrenamiento



MUCHOS DATOS

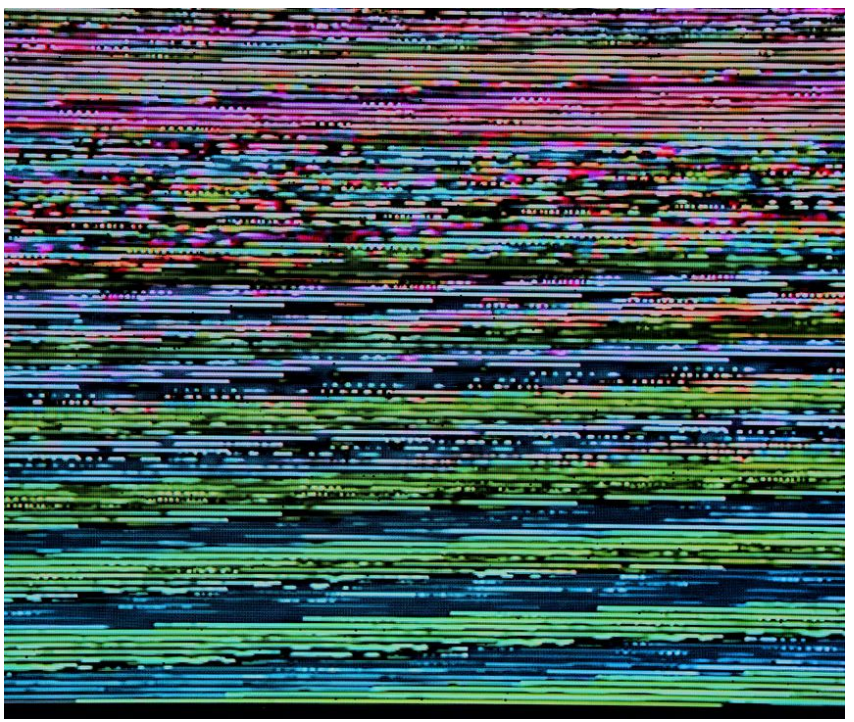
Entrenamiento



Cuidado con confundir a tu modelo

Errores más comunes

- Agregar datos de entrenamiento a testeo.
- Bases de datos no balanceadas en clases.
- Muy pocos datos.



Llevemos todos nuestros códigos a la acción - Crear modelo

Optimización de precisión de modelos

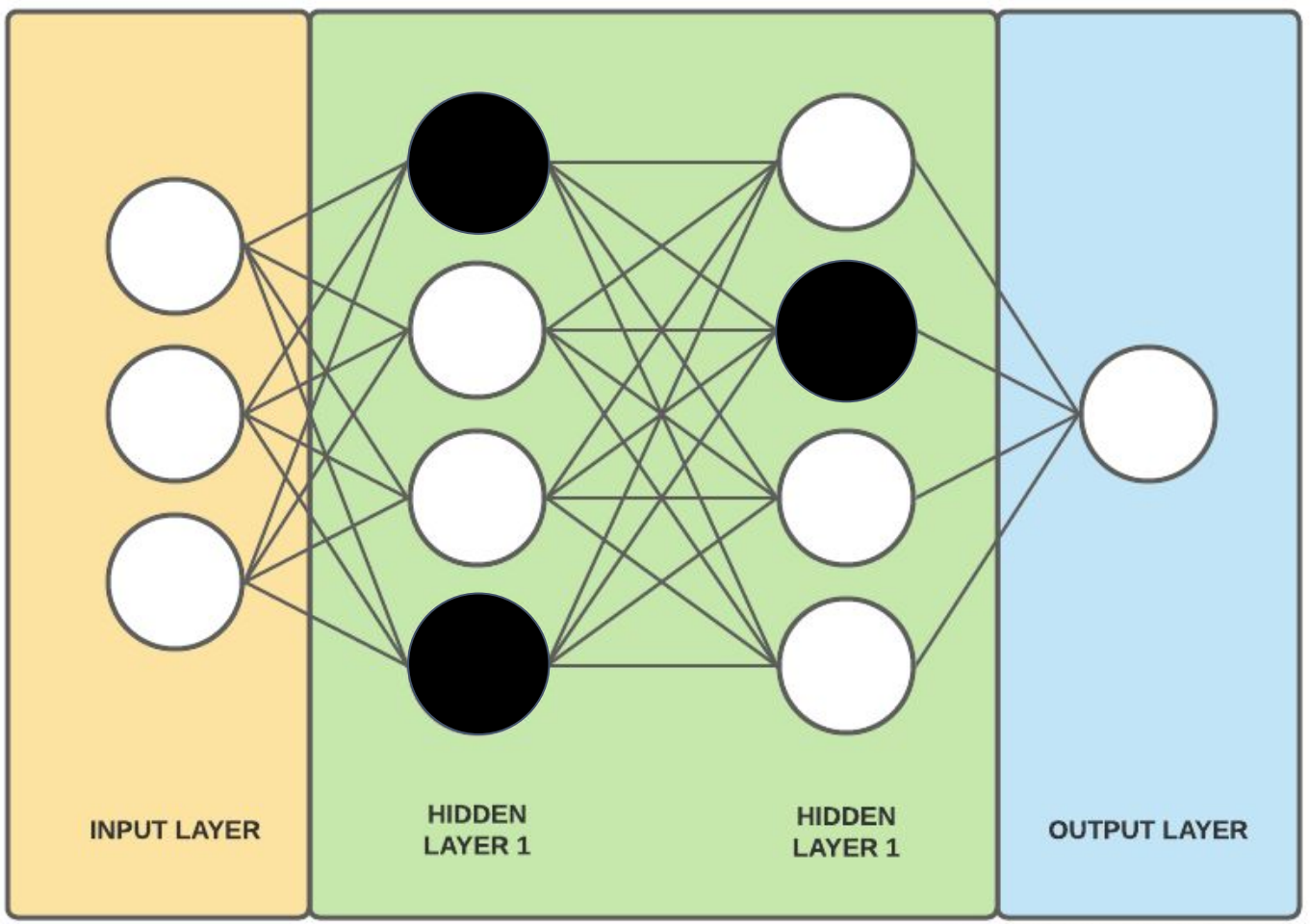
En esta sección veremos

- Underfitting y overfitting.
- Recomendaciones prácticas para ajustar mi modelo.
- Métricas para medir eficacia: callbacks.
- Monitoriza el entrenamiento de modelos con early stopping.
- Autotunnig con Keras.

Métodos de regularización

Overfitting y underfitting

Dropout


INPUT LAYER
HIDDEN LAYER 1
HIDDEN LAYER 1
OUTPUT LAYER
Designed by: Correlation one

¿Cuál regularizador utilizar?

Lasso L1

Datos de entrada irrelevantes.

Ridge L2

Datos de entrada correlacionados entre ellos.

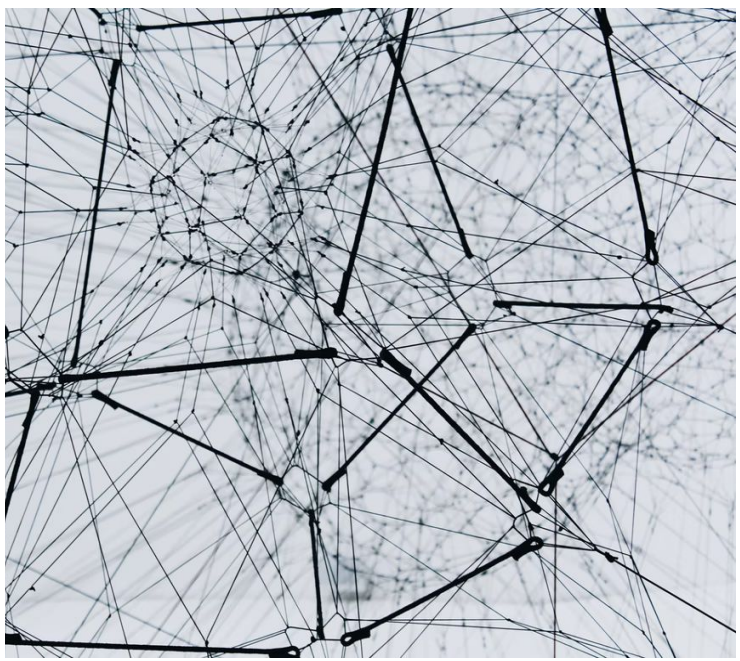
ElasticNet L1 + L2

Gran número de atributos.

Recomendaciones prácticas para ajustar mi modelo

Recomendaciones para preprocesamiento

- Busca datos null.
- Archivos corruptos.
- Balancea tu base de datos.
- Aplica normalización.
- Visualiza la base de datos.
- Entiende los datos.

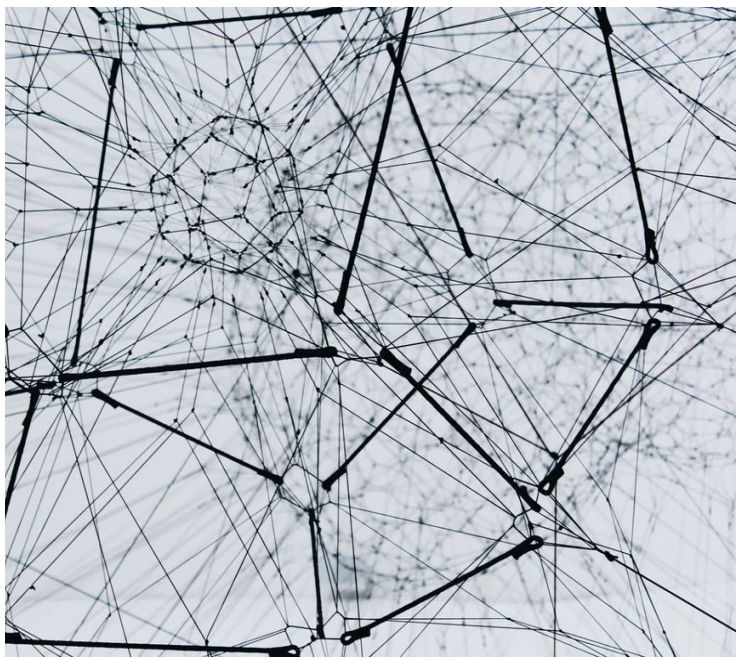


Recomendaciones de valores

- Convoluciones (3x3)
- Pooling (2x2)
- Flatten (imágenes)
- Neuronas (64, 128, 256, 512)
- Learning Rate (0.001) ADAM
- L1_2 (1e-5)
- Dropout (0.2)

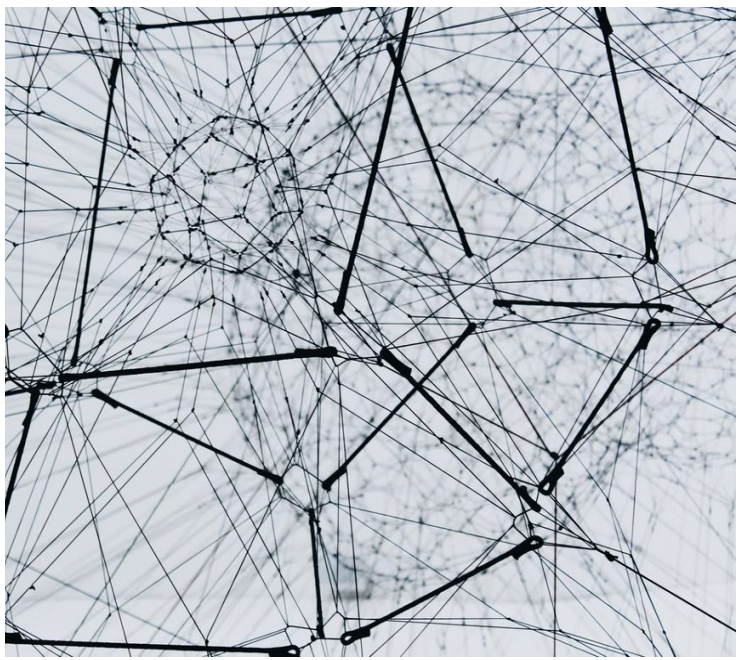
Recomendaciones para regularizadores

- L1, L2, L1+L2
- Agrega más datos.
- Data augmentation.
- Dropout.
- Early stopping.
- Callbacks.



Recomendaciones funciones de activación

- Multi Clases: **Softmax**
- Binarios: **Sigmoidal**
- Regresiones: **Función lineal**.
- Clase predicciones mayor que 0: **RelU.**

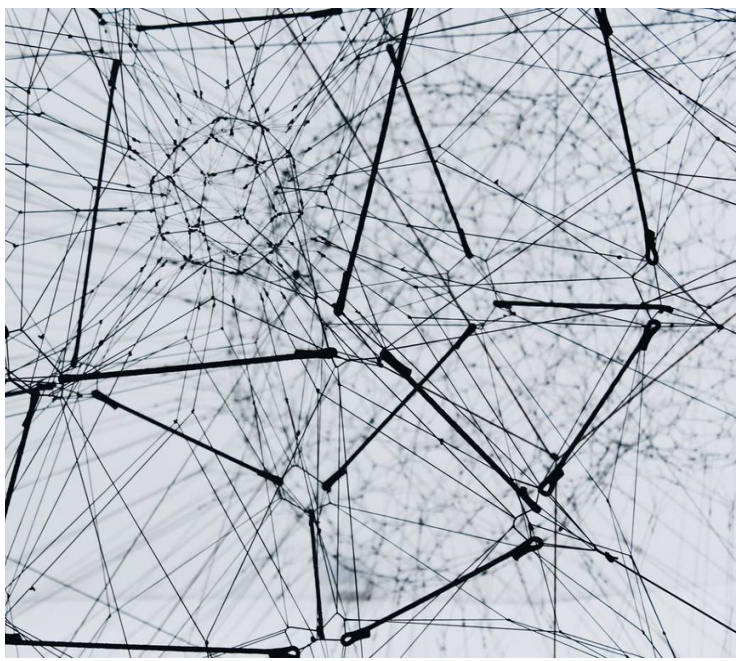


Recomendaciones configuración de red

- Aplica capas convoluciones y poolings (1 layers).
- Tómate el tiempo de encontrar el learning rate óptimo.
- Ve almacenando en cada epoch de entrenamiento.

Recomendaciones de la red

- Entre más capas y neuronas, no significa que es mejor la red.
- La solución no es siempre redes neuronales.
- Aliado aprendizaje por transferencia.



Métricas para medir la eficiencia de tu modelo

Callback

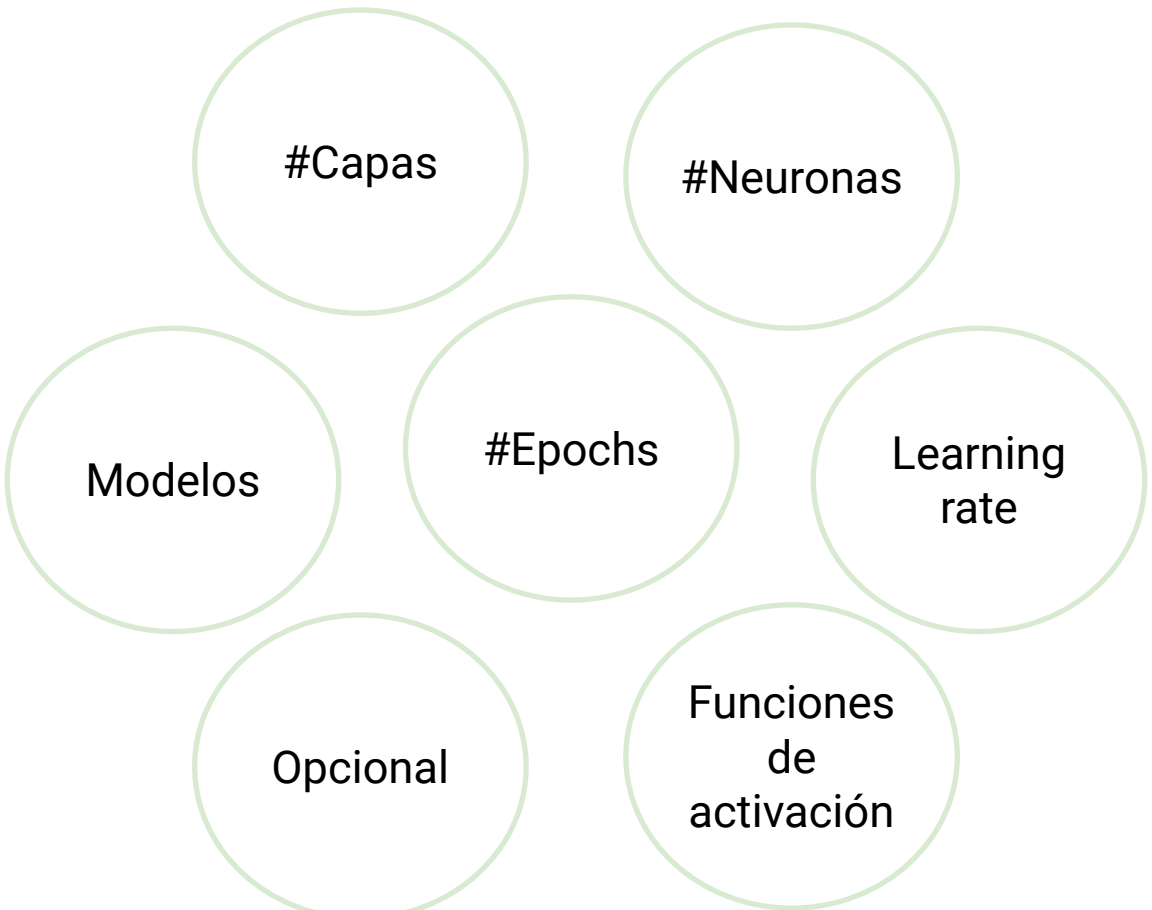
Monitoreo del entrenamiento en tiempo real

Early stopping y patience

Keras Tuner

Auto tuning

Historia del Tunner de Keras



Cómo se hacía antes

Results_entrenamiento .XLSX

Archivo Editar Ver Insertar Formato Datos Herramientas Ayuda

	REDES NEURONALES		
	Arquitectura	**Loss**	**Accuracy**
4	hp_units = hp.Int('units', min_value=32, max_value=512, step=32) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid', kernel_regularizer=regularizers.l2(0.001))) hp_units2 = hp.Int('units', min_value=32, max_value=512, step=32) model.add(keras.layers.Dense(units=hp_units2, activation='tanh')) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid', kernel_regularizer=regularizers.l2(0.001))) model.add(keras.layers.Flatten()) model.add(keras.layers.Dense(4))	1,636657119	0,8537979126
5	model.add(keras.layers.Dense(units=hp_units, activation='sigmoid', kernel_regularizer=regularizers.l2(0.001))) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid')) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid', kernel_regularizer=regularizers.l2(0.001))) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid')) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid')) model.add(keras.layers.Flatten()) model.add(keras.layers.Dense(4))	2,440923691	0,8797893524
6	hp_units = hp.Int('units', min_value=16, max_value=512, step=16) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid', kernel_regularizer=regularizers.l2(0.001))) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid')) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid', kernel_regularizer=regularizers.l2(0.001))) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid')) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid')) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid')) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid', kernel_regularizer=regularizers.l2(0.001))) model.add(keras.layers.Flatten()) model.add(keras.layers.Dense(4))	8,43824832	0,5502405
7	hp_units = hp.Int('units', min_value=16, max_value=512, step=16) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid', kernel_regularizer=regularizers.l2(0.001))) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid')) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid')) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid')) model.add(keras.layers.Dense(units=hp_units, activation='sigmoid')) model.add(keras.layers.Flatten()) model.add(keras.layers.Dense(4))	8,579723358	0,5503473282

Keras Tuner

Buscando la mejor configuración
para tu modelo

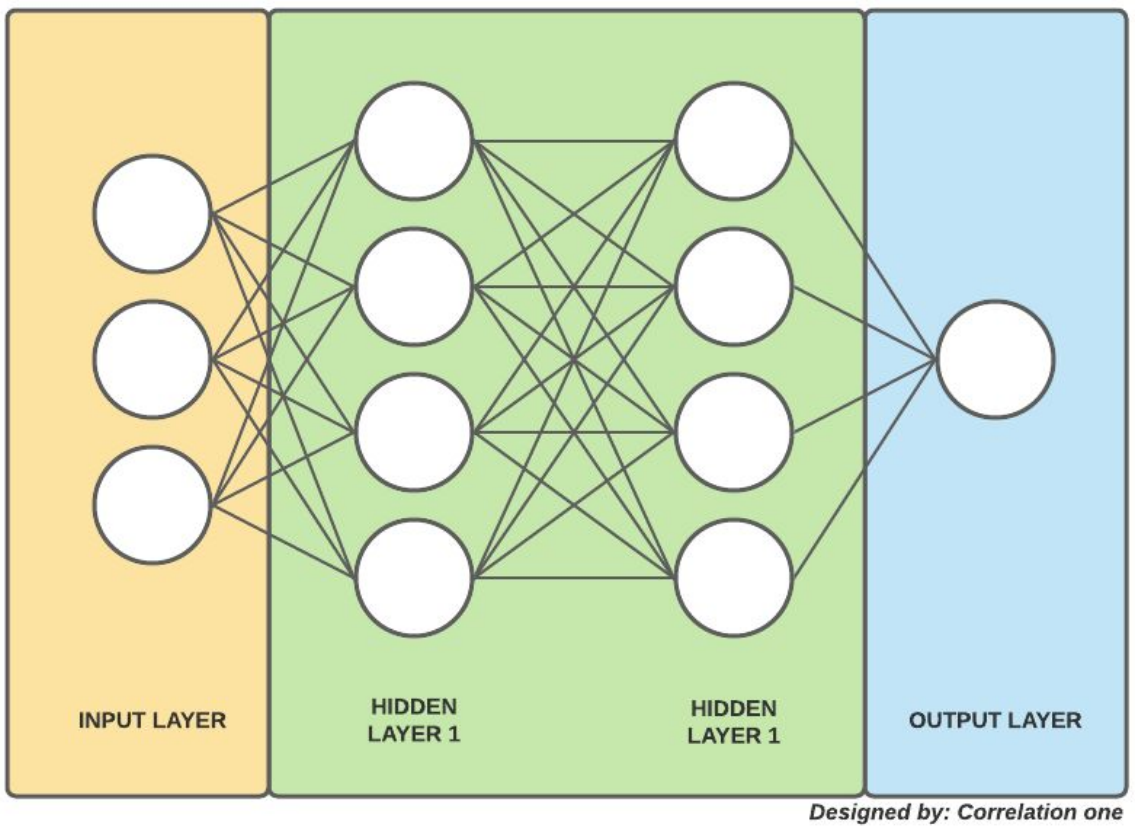
Almacenamiento y carga de modelos

Pesos y arquitectura

En esta sección aprenderemos

- Almacenamiento y carga de modelos: pesos y arquitectura.
- Criterios para almacenar los modelos.

Componentes relevantes de un modelo



Arquitectura

Pesos

Etiquetas (Labels)

Aprendizaje por transferencia.

Criterios para almacenar los modelos

Introducción al aprendizaje por transferencia

En este módulo aprenderemos

- Introducción al aprendizaje por transferencia.
- Cómo cargar sistemas pre-entrenados en Keras.
- Cómo utilizar sistemas pre-entrenados desde los repositorios de TensorFlow Hub.
- Aplicar modelo pre-entrenado a nuestro proyecto.

Transferir conocimiento

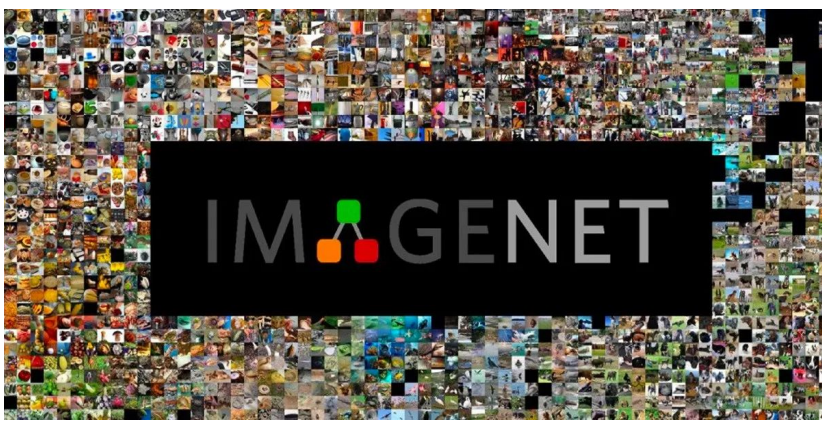
Transferir características principales

PEDRO TONELO

Modelo Matemático

Modelo Matemático

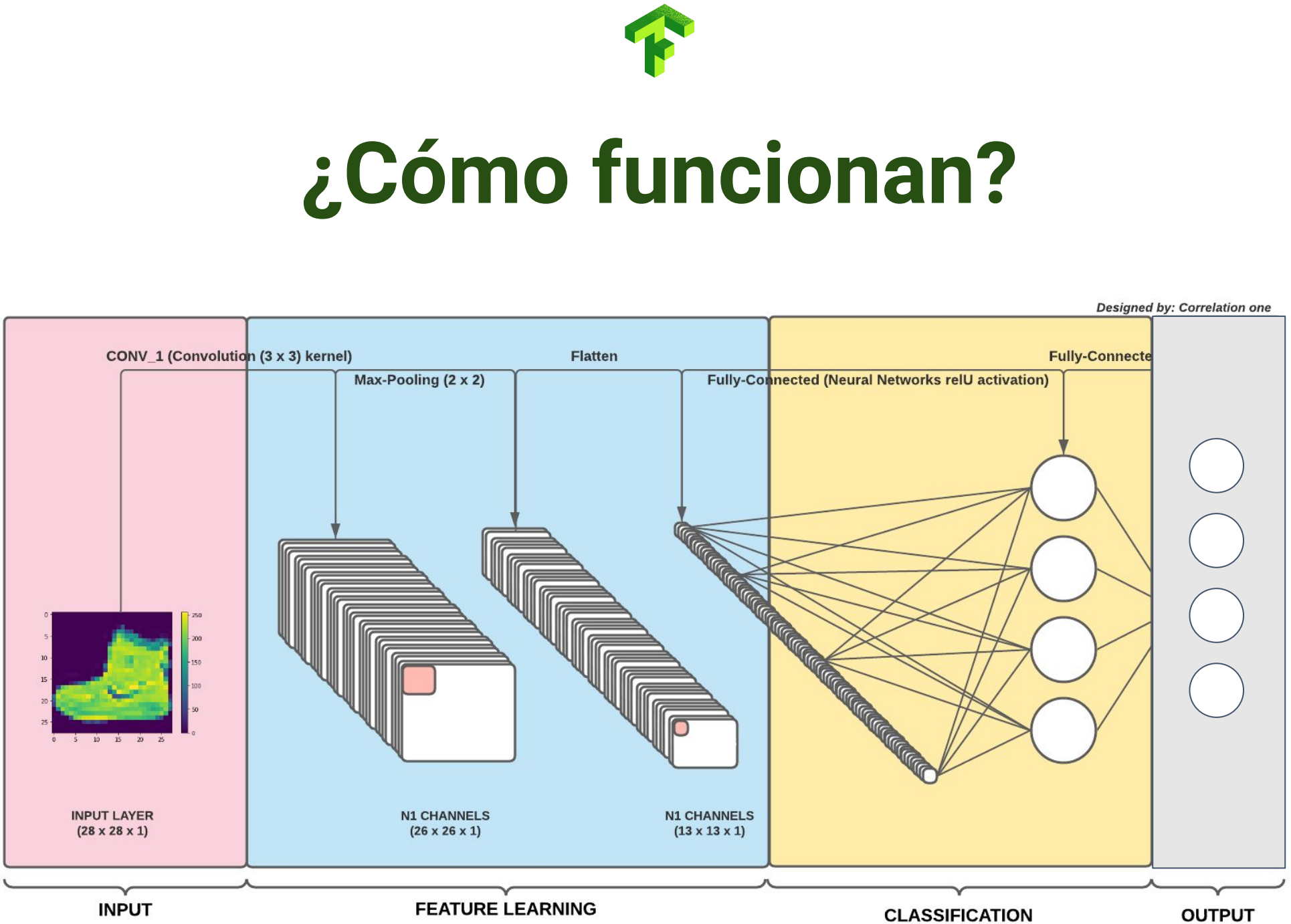
Bases de datos de entrenamiento



OpenAI

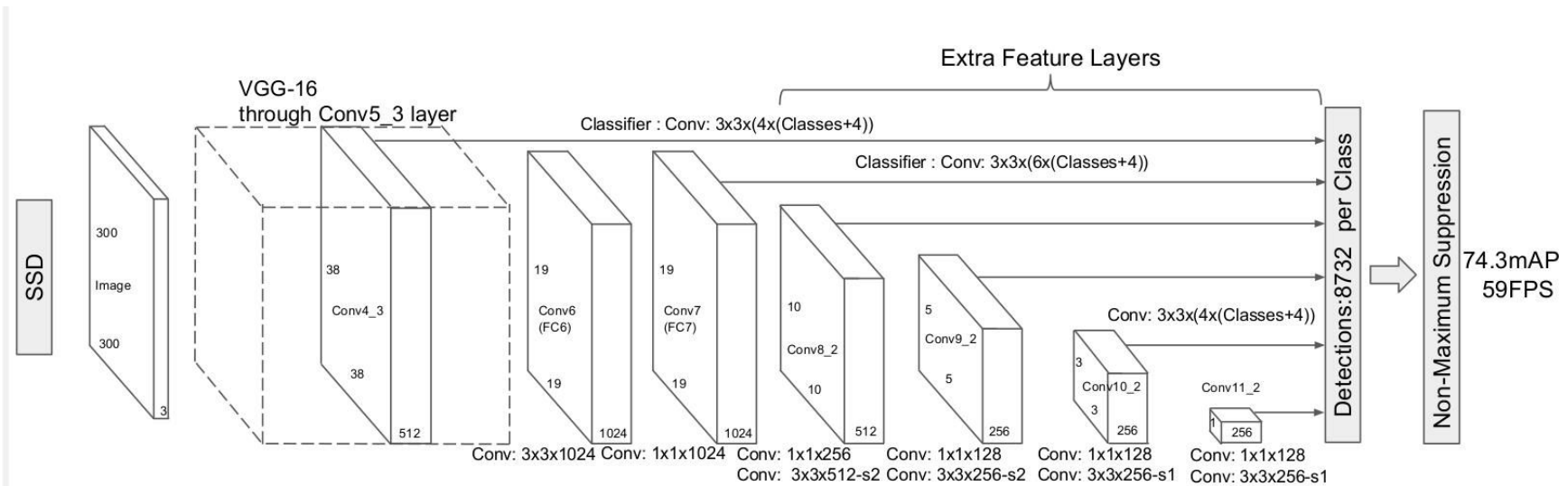
GPT-3, an autoregressive language model with 175 billion parameters

COCO
Common Objects in Context



¿Cómo funcionan?

Ejemplo de infraestructuras de una red convolucional



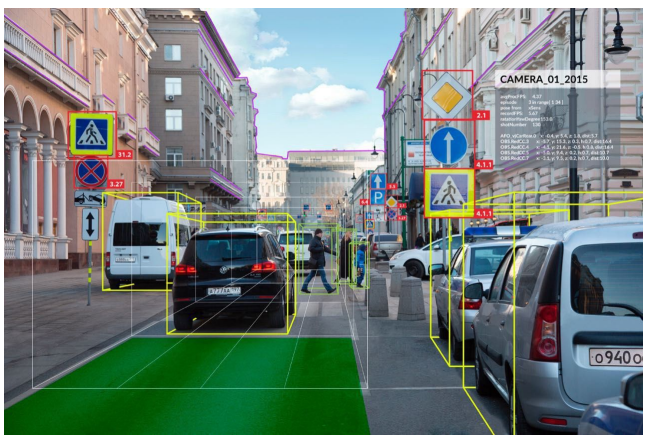
Fuente: https://arxiv.org/abs/1512.02325

```
from tensorflow.keras.layers import Dense
from tensorflow.keras.Model import Sequential
from tensorflow.keras.applications.inception_v3 import InceptionV3

## Url inception model.
weights_file = '/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'

new_model = Sequential()
new_model.add(InceptionV3(include_top = False,
                          weights = weights_file))

new_model.add(Dense(num_classes, activation = 'softmax'))
new_model.layers[0].trainable = False
```

Cuándo utilizar aprendizaje por transferencia

Cuándo utilizar modelos pre-entrenados

Procesamiento de lenguaje natural

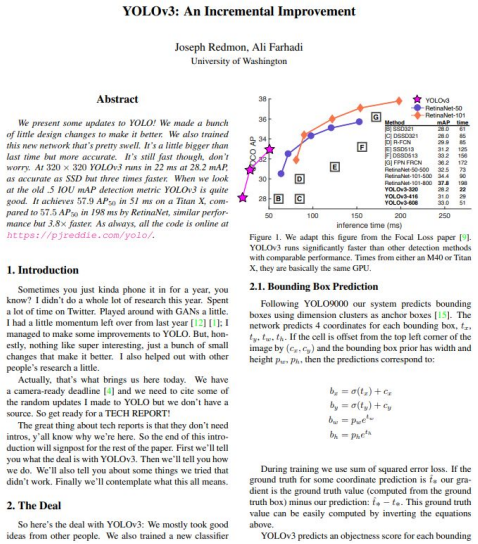


Visión computarizada

¿Por qué utilizarlos?

- Se necesita poca data.
- Te permite generar iteraciones muy rápidas.
- Generaliza muy bien los modelos.
- Transferir extracción de características principales.

Recomendaciones de artículos pre-entrenados



YOLOv3: An Incremental Improvement

Joseph Redmon, Ali Farhadi
University of Washington

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP_{50} in 51 ms on a Titan X, compared to 57.5 AP_{50} in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.

1. Introduction

Sometimes you just kinda phone it in for a year, you know? I didn't do a whole lot of research this year. Spent a lot of time on Twitter. Played around with GANs a little. I had a little momentum left over from last year [12] [1]; I managed to make some improvements to YOLO. But, honestly, nothing like super interesting, just a bunch of small changes that make it better. I also helped out with other people's research a little.

Actually, that's what brings us here today. We have a camera-ready deadline [4] and we need to cite some of the random updates I made to YOLO but we don't have a source. So get ready for a TECH REPORT!

The great thing about tech reports is that they don't need intros, y'all know why we're here. So the end of this introduction will signpost for the rest of the paper. First we'll tell you what the deal is with YOLOv3. Then we'll tell you how we do. We'll also tell you about some things we tried that didn't work. Finally we'll contemplate what this all means.

2. The Deal

So here's the deal with YOLOv3: We mostly took good ideas from other people. We also trained a new classifier

Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

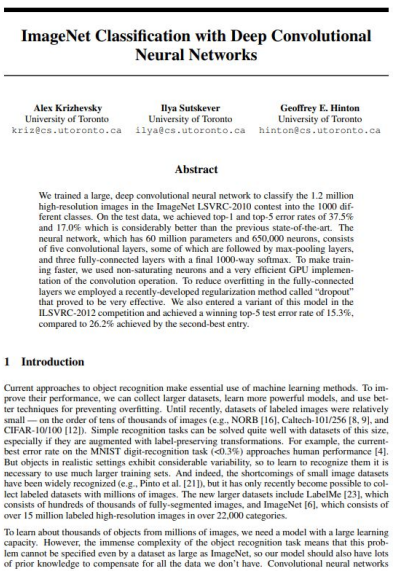
2.1. Bounding Box Prediction

Following YOLO9000 our system predicts bounding boxes using dimension clusters as anchor boxes [15]. The network predicts 4 coordinates for each bounding box, t_x, t_y, t_w, t_h. If the cell is offset from the top left corner of the image by (c_x, c_y) and the bounding box prior has width and height p_w, p_h, then the predictions correspond to:

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

During training we use sum of squared error loss. If the ground truth for some coordinate prediction is $\hat{t}_*$ our gradient is the ground truth value (computed from the ground truth box) minus our prediction: $\hat{t}_* - t_*$. This ground truth value can be easily computed by inverting the equations above.

YOLOv3 predicts an objectness score for each bounding



ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

1 Introduction

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small — on the order of tens of thousands of images (e.g., NORB [16], Caltech-101/256 [8, 9], and CIFAR-10/100 [12]). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current-best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance [4]. But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized (e.g., Pinto et al. [21]), but it has only recently become possible to collect labeled datasets with millions of images. The new larger datasets include LabelMe [23], which consists of hundreds of thousands of fully-segmented images, and ImageNet [6], which consists of over 15 million labeled high-resolution images in over 22,000 categories.

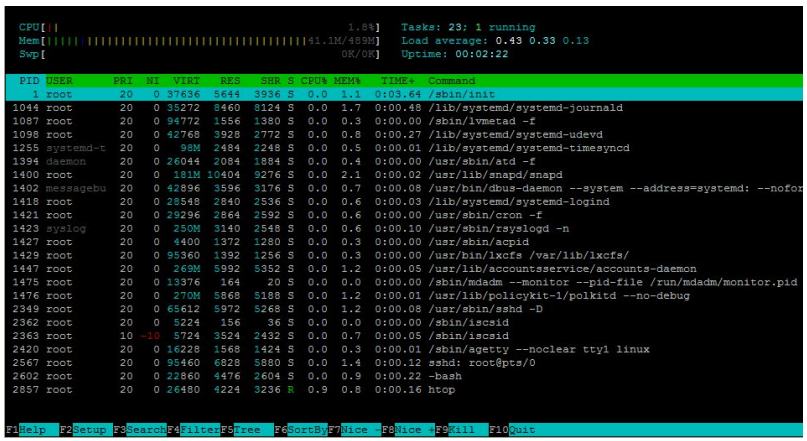
To learn about thousands of objects from millions of images, we need a model with a large learning capacity. However, the immense complexity of the object recognition task means that this problem cannot be specified even by a dataset as large as ImageNet, so our model should also have lots of prior knowledge to compensate for all the data we don't have. Convolutional neural networks

YoloV3

AlexNet

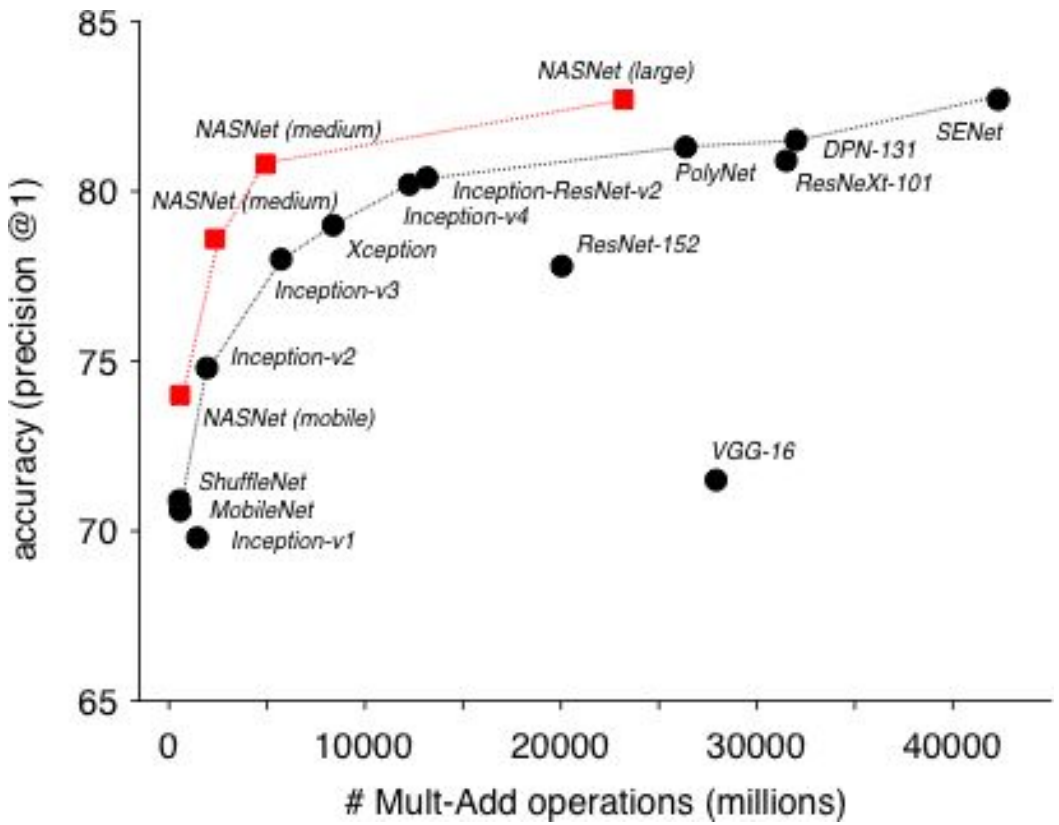
¿Cómo saber cuál modelo seleccionar?

Accuracy (precisión)



Coste computacional

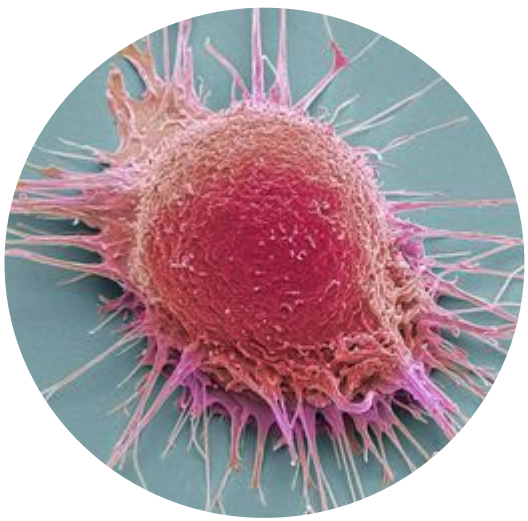
¿Cómo seleccionar cual modelo pre-entrenado usar?



Fuente: https://ai.googleblog.com/2017/11/automl-for-large-scale-image.html

Reto

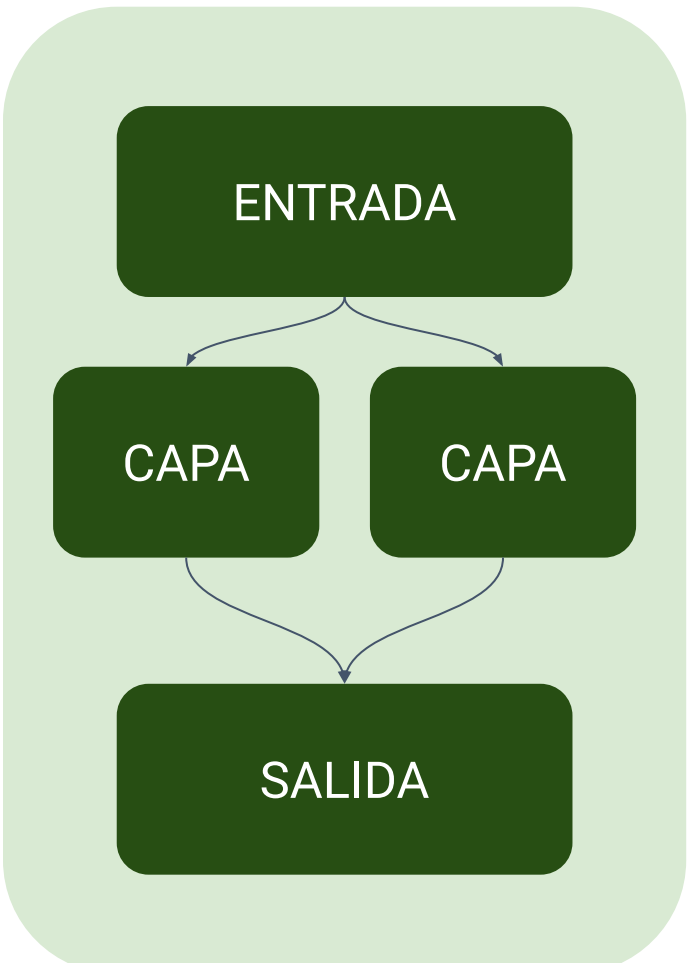
Dron detectando y persiguiendo aves.

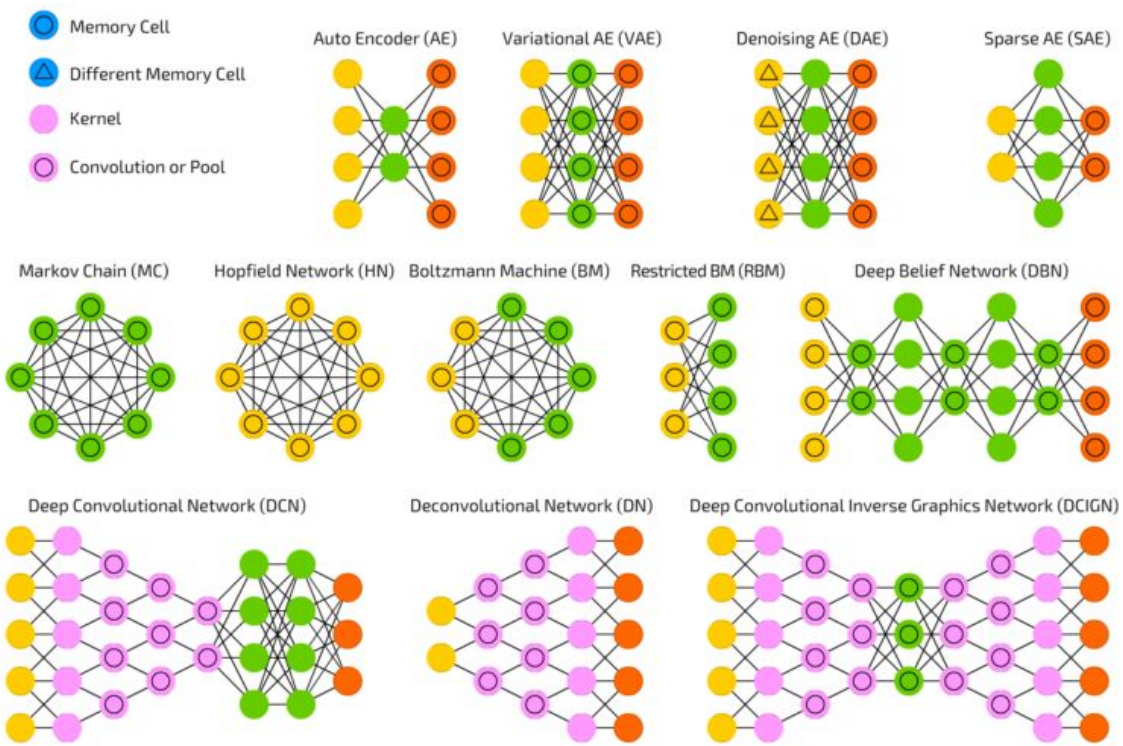


Detección de cáncer.

Carga de sistemas pre-entrenados en Keras

API Funcional Keras





API funcional de Keras

Uso sistemas pre-entrenados de TensorFlow Hub

Introducción a variables relevantes del TensorBoard

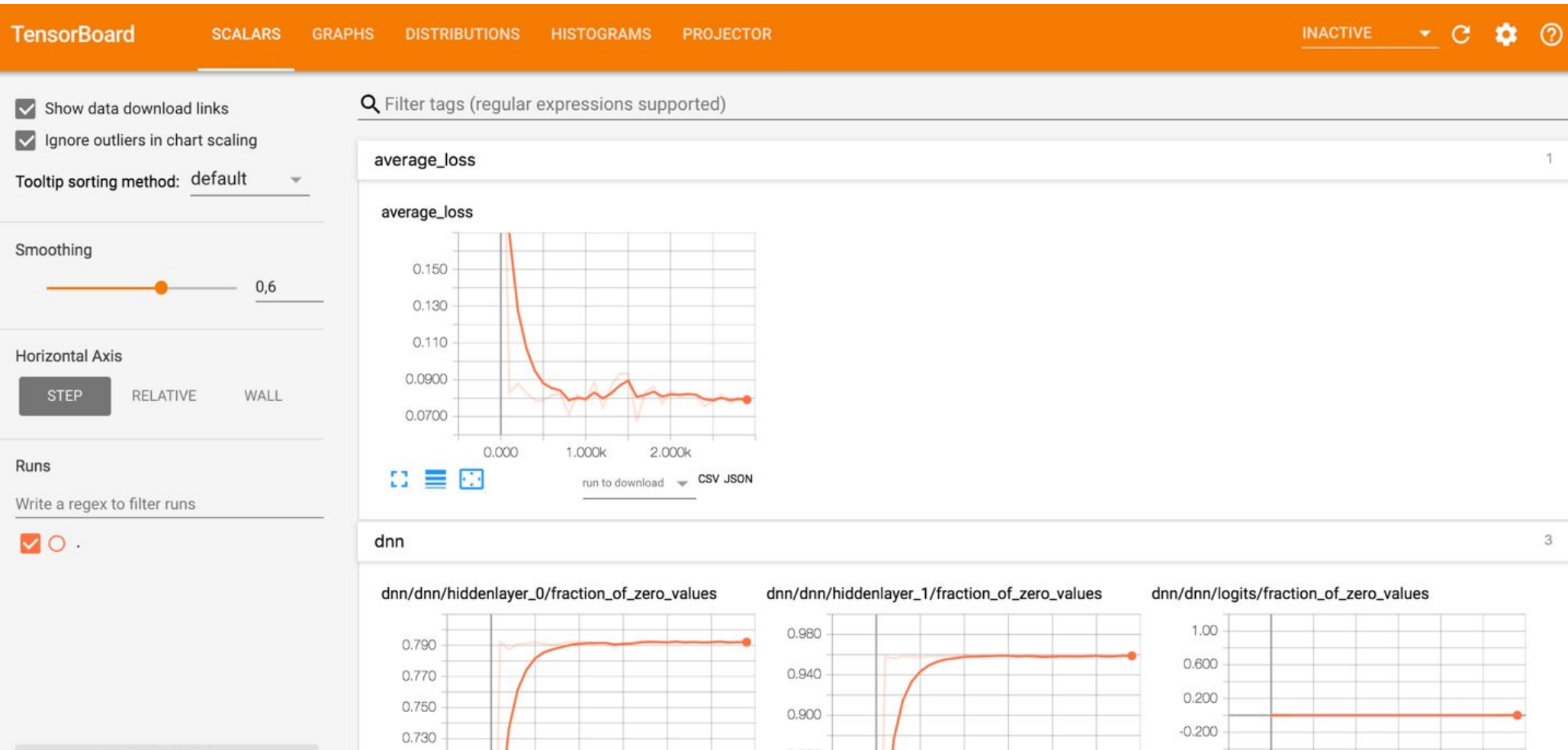
En esta sección aprenderemos

- Introducción a variables relevantes del TensorBoard.
- Análisis y publicación de resultados del entrenamiento.
- Introducción para poner modelos en producción.
- Cómo poner en producción.

Introducción a TensorBoard

- Herramienta para visualizar resultados.
- Entender el flujo y los tensores.
- Debuggear tu modelo.

Ejemplo práctico de TensorBoard



Análisis y publicación de resultados del entrenamiento

Introducción al despliegue de modelos en producción

Production


Configuration
Data Collection
Data Verification
Machine Resource Management
Serving Infrastructure
Monitoring
ML Code
Analysis Tools
Feature Extraction
Process Management Tools

Ejemplos de producción

Nube

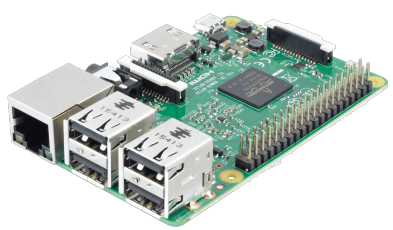
Sistemas embebidos

Google Cloud

Azure

aws

ORACLE CLOUD



Ejemplos de producción

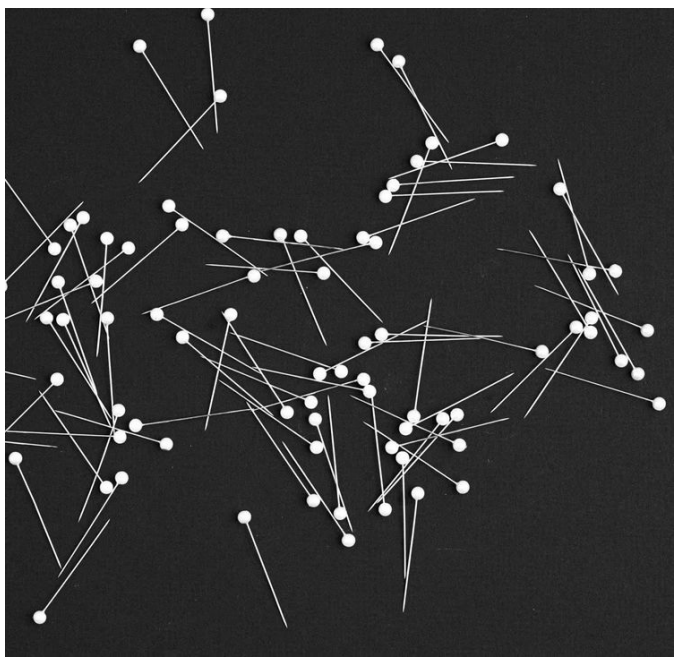
Local

USB Accelerator

Siguientes pasos con deep learning

Qué más aprender

- Redes neuronales convolucionales.
- Data Augmentation.
- Formato TF Records para entrenar modelos.
- Despliegue de modelos a producción.
- Visión computarizada.



www.switchai.co