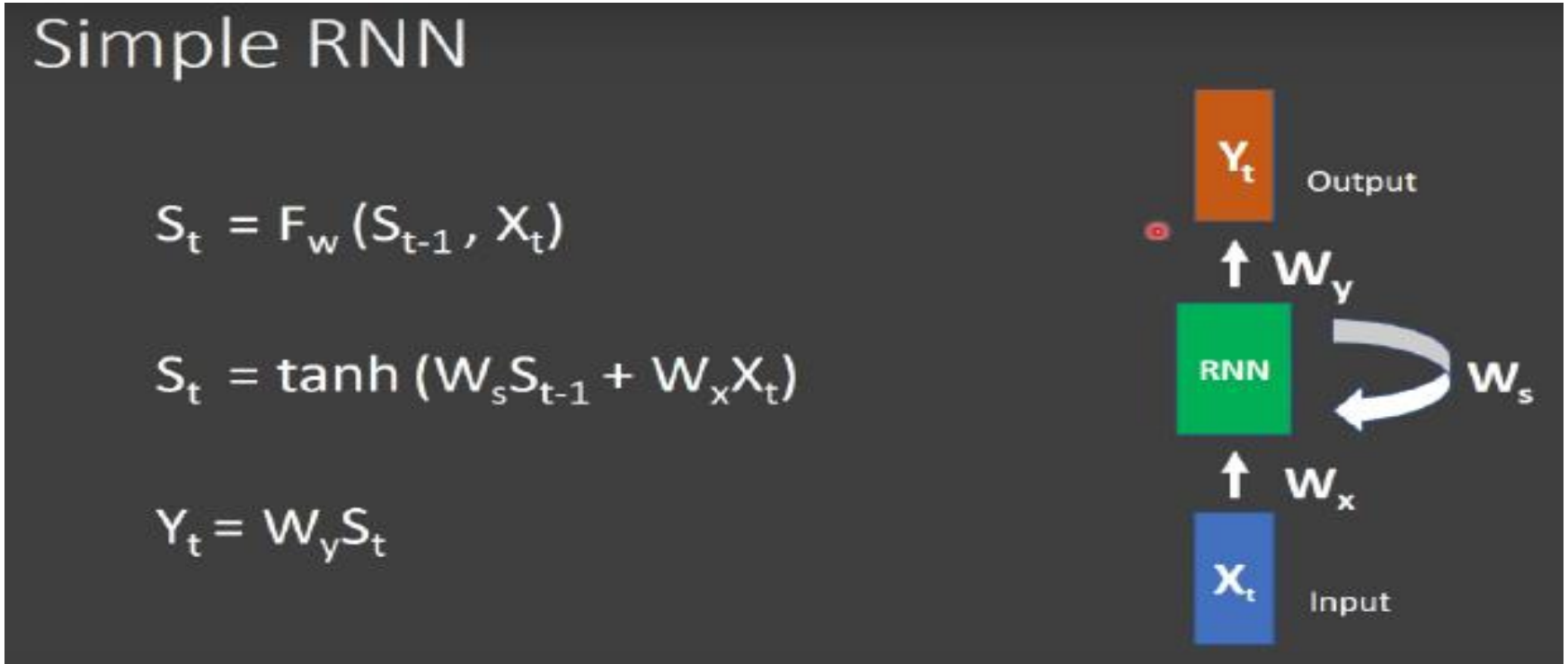
REDES NEURONALES ARTIFICIALES – DEEP LEARNING

MODELOS DE ARQUITECTURA AVANZADA: RNN – GAN – REINFORCEMENT – TRANSFER – AUTOENCODERS - TRANSFORMER Y MECANISMOS DE ATENCIÓN

LAURA DIAZ DÁVILA

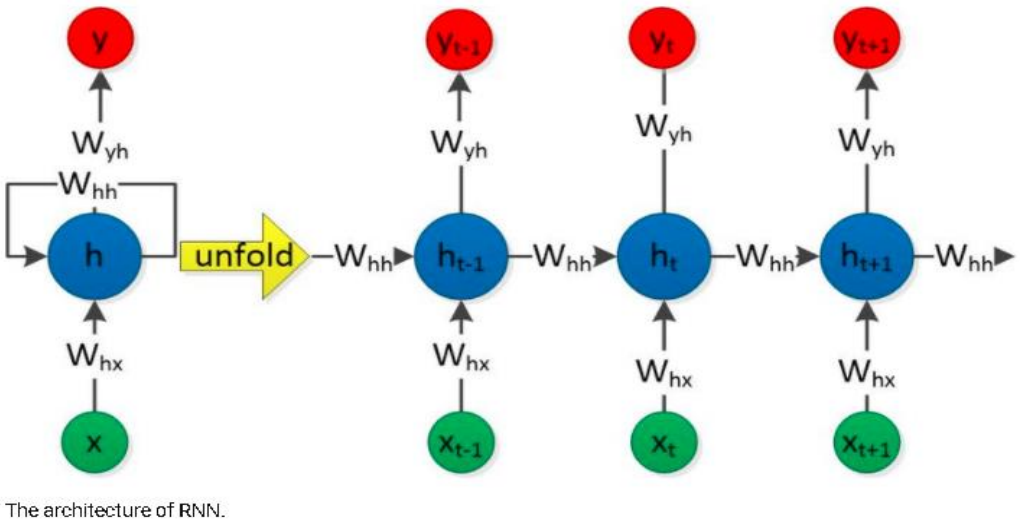
RECURRENT NEURAL NETWORK

Feedforward network produce un valor que en muchos casos es una clase o una predicción. RNN es adecuado para datos de series temporales, donde una salida puede ser el siguiente valor en una secuencia, o los siguientes valores. La decisión depende de lo que requiera la aplicación, regresión de clasificación o previsión



RECURRENT NEURAL NETWORKS

TensorFlow



The architecture of RNN.

https://arxiv.org/abs/1409.3215

Fuente: https://www.tensorflow.org/guide/keras/rnn

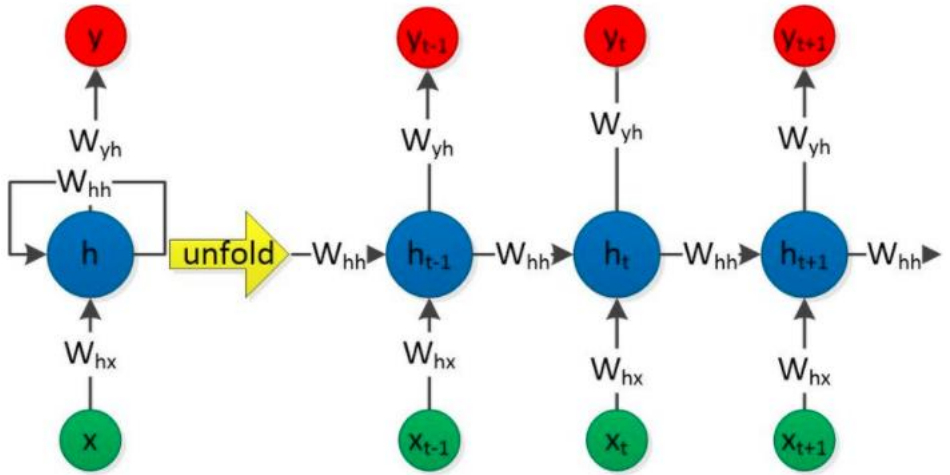
RECURRENT NEURAL NETWORKS

TensorFlow

Capas RNN integradas: un ejemplo sencillo

Hay tres capas RNN integradas en Keras:

1. `keras.layers.SimpleRNN` , un RNN completamente conectado donde la salida del paso de tiempo anterior se alimenta al siguiente paso de tiempo.
2. `keras.layers.GRU` , propuesto por primera vez en Cho et al., 2014 .
3. `keras.layers.LSTM` , propuesto por primera vez en Hochreiter & Schmidhuber, 1997 .



The architecture of RNN.

Fuente: https://www.tensorflow.org/guide/keras/rnn

RECURRENT NEURAL NETWORKS

TensorFlow

Redes neuronales recurrentes (RNN) con Keras

Configuración

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

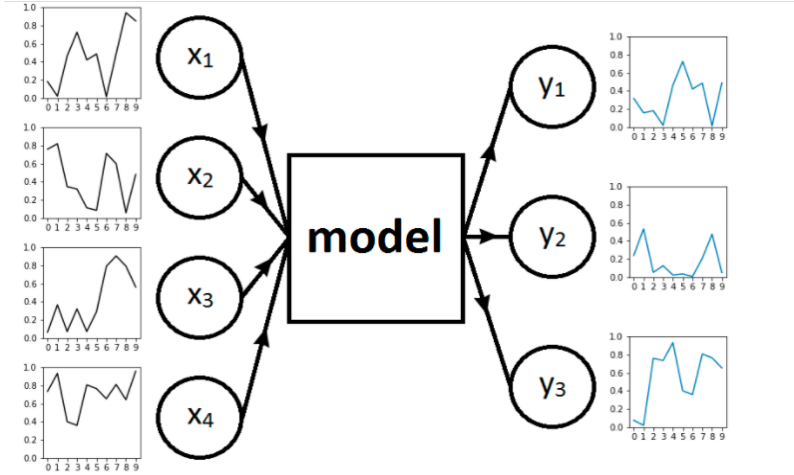


Fig. 1. Framework with input time series on the left, RNN model in the middle, and output time series on the right

Fuente: https://www.tensorflow.org/guide/keras/rnn

Deep Learning – RNN - CNN

- TENSOR FLOW:
  - MLP (mlp.py)
  - RNN (rnn.py)
  - CNN (cnn.py)
- API Keras sobre Tensorflow

Tipos de Modelos en Tensor Flow para Procesamiento de Lenguaje Natural

Detección de texto tóxico

Codificador universal de oraciones

Segmentación semántica

Uso de lenguaje natural para responder preguntas

Tipos de Modelos en Tensor Flow en Imágenes

- Clasificación de imágenes
- Detección de objetos
- Segmentación del cuerpo
- Estimación de pose

TENSOR FLOW y ABSTRACCIÓN

- Facilita el proceso de adquisición de datos, modelos de capacitación, predicciones y refinamiento de resultados futuros.
- Creado por el equipo de Google Brain, es una biblioteca de código abierto para el cálculo numérico y el aprendizaje automático a gran escala.
- Reúne una gran cantidad de algoritmos y modelos de aprendizaje automático y de aprendizaje profundo.
- Utiliza Python para proporcionar una conveniente API de front-end para crear aplicaciones con el framework, mientras ejecuta esas aplicaciones en C ++ de alto rendimiento

Tensorflow Codes (MLP, RNN, CNN) en MNIST dataset

mlp.py

```
Epoch 1 completed out of 10 loss: 1580448.2423095703
Epoch 2 completed out of 10 loss: 413067.3898963928
Epoch 3 completed out of 10 loss: 219827.5313282013
Epoch 4 completed out of 10 loss: 134923.41420912743
Epoch 5 completed out of 10 loss: 81669.84851998091
Epoch 6 completed out of 10 loss: 52424.493058502674
Epoch 7 completed out of 10 loss: 36300.166654587985
Epoch 8 completed out of 10 loss: 29144.069483664625
Epoch 9 completed out of 10 loss: 19910.462027701346
Epoch 10 completed out of 10 loss: 17675.114113055613
Accuracy: 0.9516
```

rnn.py

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
from tensorflow.contrib import rnn
mnist = input_data.read_data_sets("/tmp/data/", one_hot=True)

hm_epochs = 5
n_classes = 10
batch_size = 256
chunk_size = 28
n_chunks = 28
rnn_size = 256
```

```
Epoch 1 completed out of 5 loss: 110.2273875772953
Epoch 2 completed out of 5 loss: 30.447120390832424
Epoch 3 completed out of 5 loss: 20.566452082246542
Epoch 4 completed out of 5 loss: 15.800598226487637
Epoch 5 completed out of 5 loss: 12.482633021660149
Accuracy: 0.9801
```

cnn.py

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("/tmp/data/", one_hot=True)

n_classes = 10
batch_size = 128
```

```
train_neural_network(x)
```

```
Epoch 0 completed out of 10 loss: 2921454.18756
Epoch 1 completed out of 10 loss: 574050.522736
Epoch 2 completed out of 10 loss: 305611.393166
Epoch 3 completed out of 10 loss: 197086.48822
Epoch 4 completed out of 10 loss: 137453.991611
Epoch 5 completed out of 10 loss: 101296.725072
Epoch 6 completed out of 10 loss: 78174.2471703
Epoch 7 completed out of 10 loss: 60469.5233727
Epoch 8 completed out of 10 loss: 50019.7263604
Epoch 9 completed out of 10 loss: 40857.0956538
Accuracy: 0.9681
```

Keras code (CNN) imágenes clasificador Cats and Dogs

```
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D
from keras.layers import Dense, Activation, Dropout, Flatten
from keras import optimizers
from keras.models import Sequential
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from IPython.display import display
from PIL import Image

img_width = 150
img_height = 150
train_data_dir = 'data/train'
valid_data_dir = 'data/validation'
test_dir = 'test'
```

```
Epoch 45/50
  1/128 [..............................] - ETA: 35s - loss: 8.4870e-07 - acc: 1.
  2/128 [..............................] - ETA: 35s - loss: 0.0186 - acc: 1.0000
128/128 [==============================] - 40s 315ms/step - loss: 0.2217 - acc:
0.9375 - val_loss: 1.4616 - val_acc: 0.7308
Epoch 46/50
128/128 [==============================] - 40s 316ms/step - loss: 0.2559 - acc:
0.9414 - val_loss: 1.3608 - val_acc: 0.7380
Epoch 47/50
128/128 [==============================] - 41s 319ms/step - loss: 0.1908 - acc:
0.9473 - val_loss: 1.3346 - val_acc: 0.7500
Epoch 48/50
128/128 [==============================] - 40s 312ms/step - loss: 0.2821 - acc:
0.9160 - val_loss: 1.0499 - val_acc: 0.7380
Epoch 49/50
128/128 [==============================] - 41s 320ms/step - loss: 0.1973 - acc:
0.9443 - val_loss: 1.7975 - val_acc: 0.7284
Epoch 50/50
  1/128 [..............................] - ETA: 47s - loss: 6.1511e-05 - acc: 1.
  2/128 [..............................] - ETA: 45s - loss: 6.2872e-04 - acc: 1.
  3/128 [..............................] - ETA: 46s - loss: 0.0266 - acc: 1.0000
128/128 [==============================] - 42s 326ms/step - loss: 0.2482 - acc:
0.9248 - val_loss: 1.2311 - val_acc: 0.7236
```

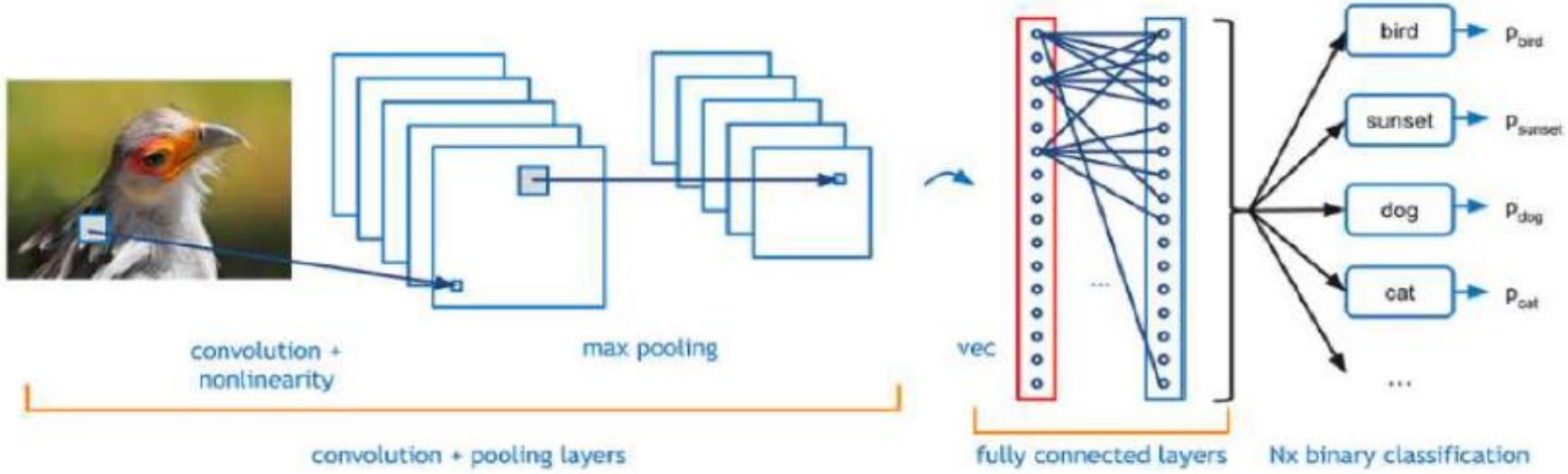
Keras

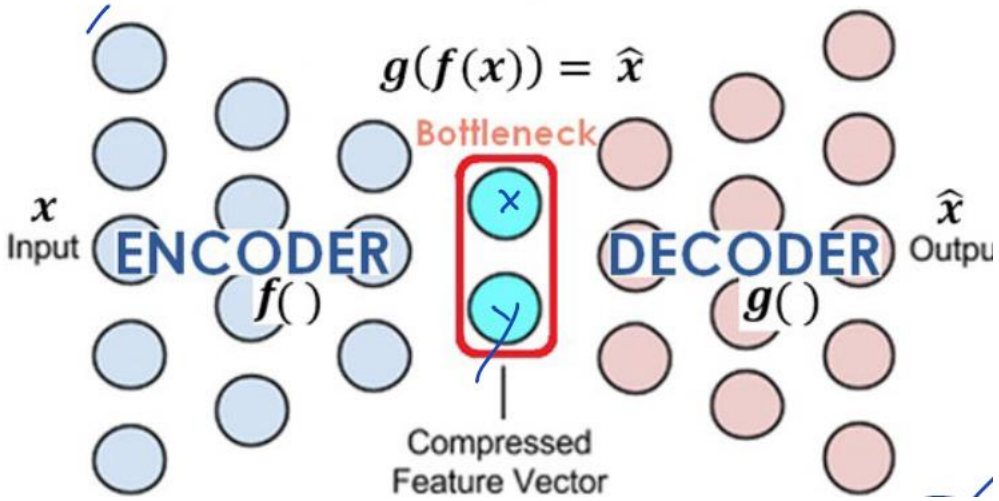
Google colab

TensorFlow

kaggle

PyTorch

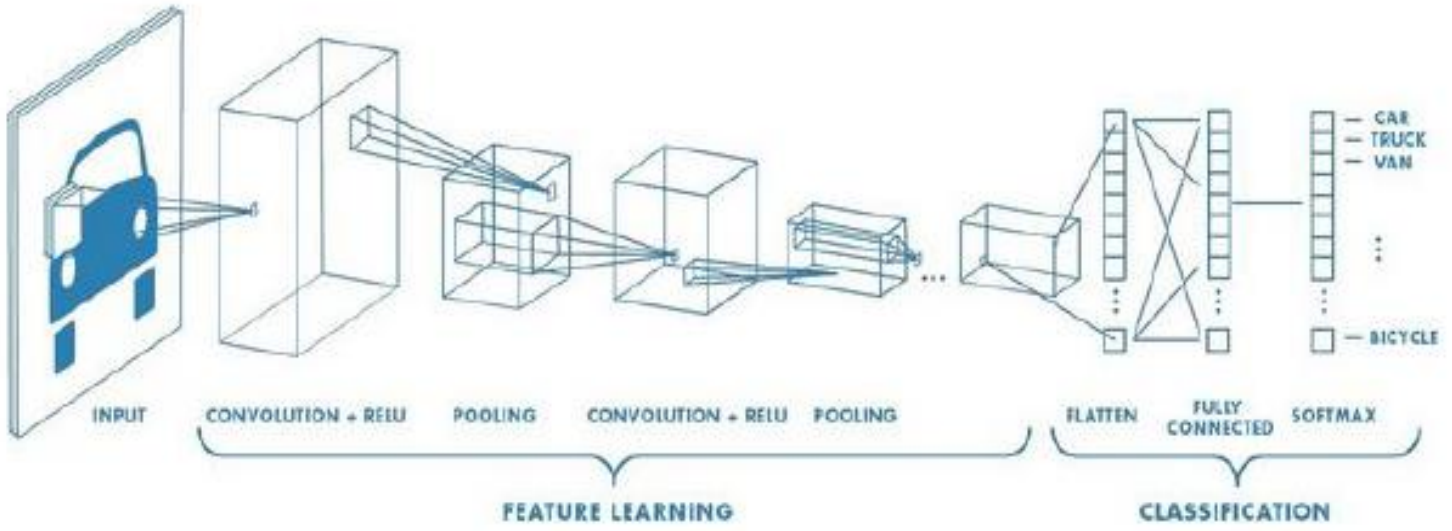




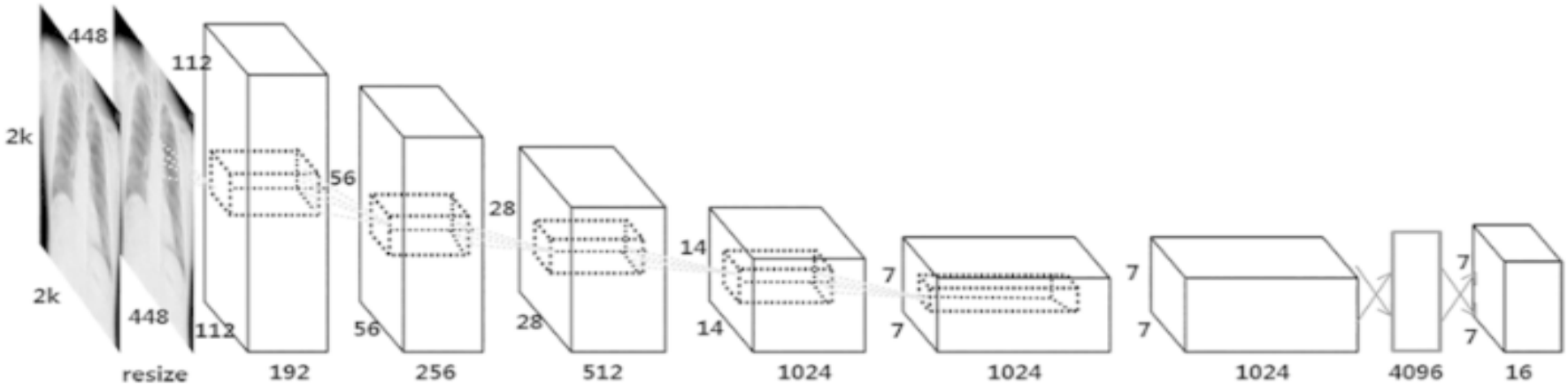
AUTOENCODER CONVOLUCIONAL

REPRESENTACIÓN LATENTE

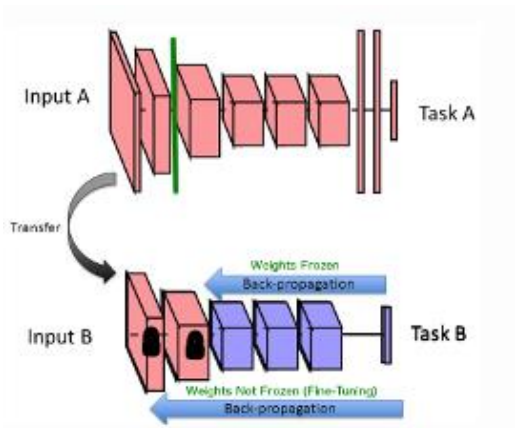
TRANSFER LEARNING

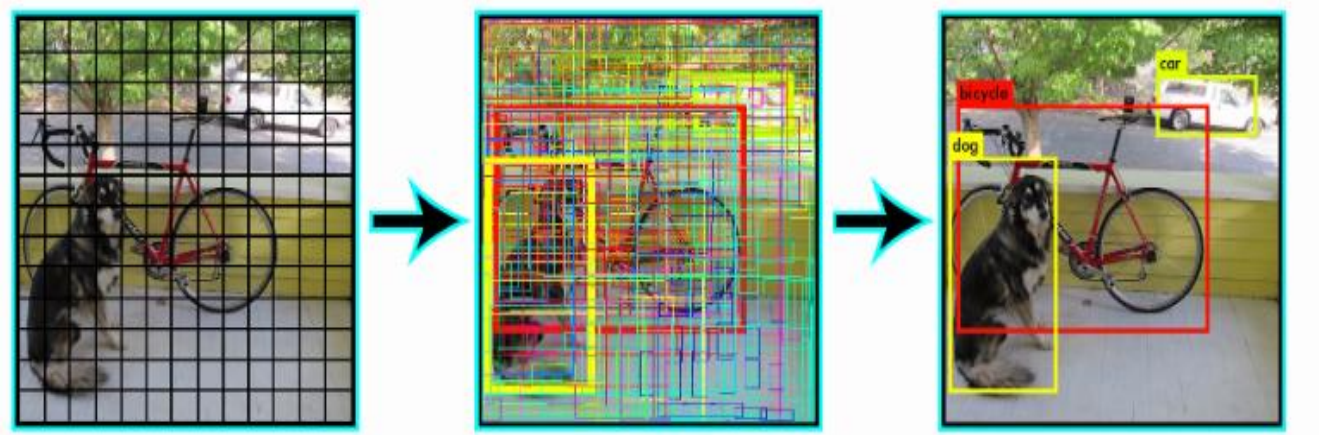


You Only Look Once (YOLO)



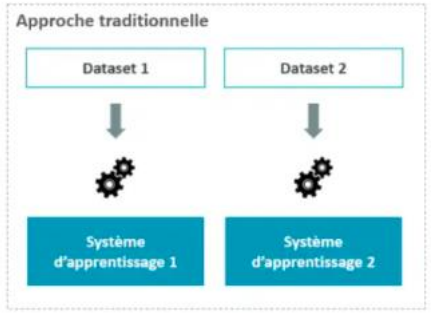
YOLO v2 network with DenseNet201 as a backend network for transfer learning

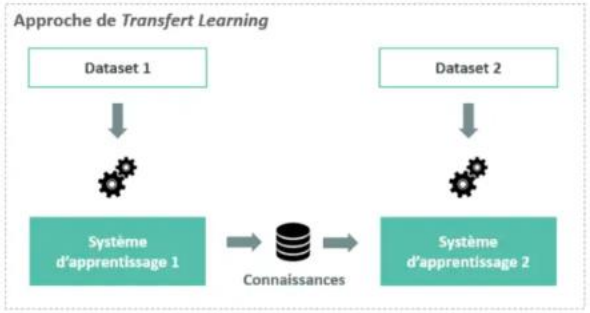




Fuente: https://hypraptive.github.io/2017/01/29/find-the-bears-yolo.html

TRANSFER LEARNING – DEEP LEARNING





Tipos de TL

Inductive

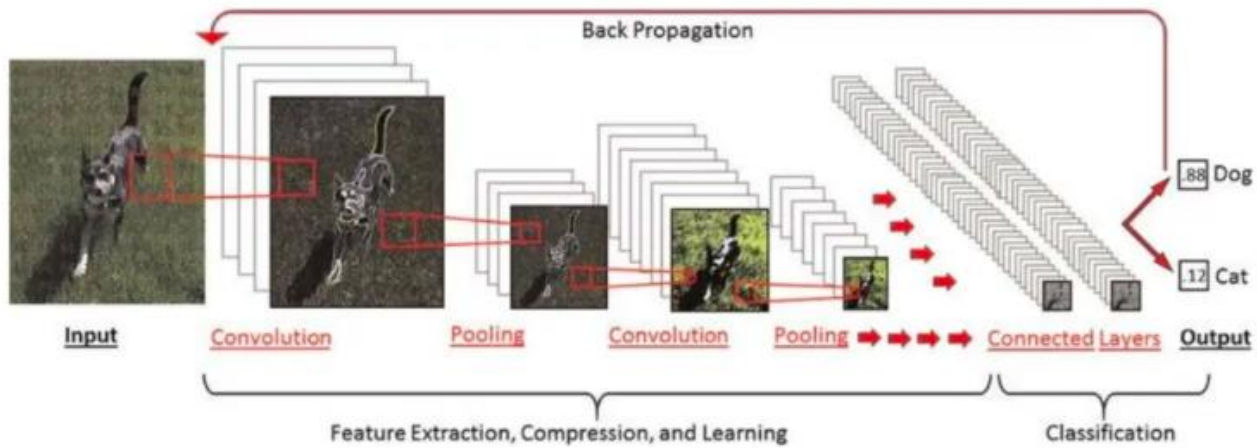
Unsupervised

Transductive

Computer Vision

Natural Language Processing

https://datascientest.com/es/que-es-el-transfer-learning



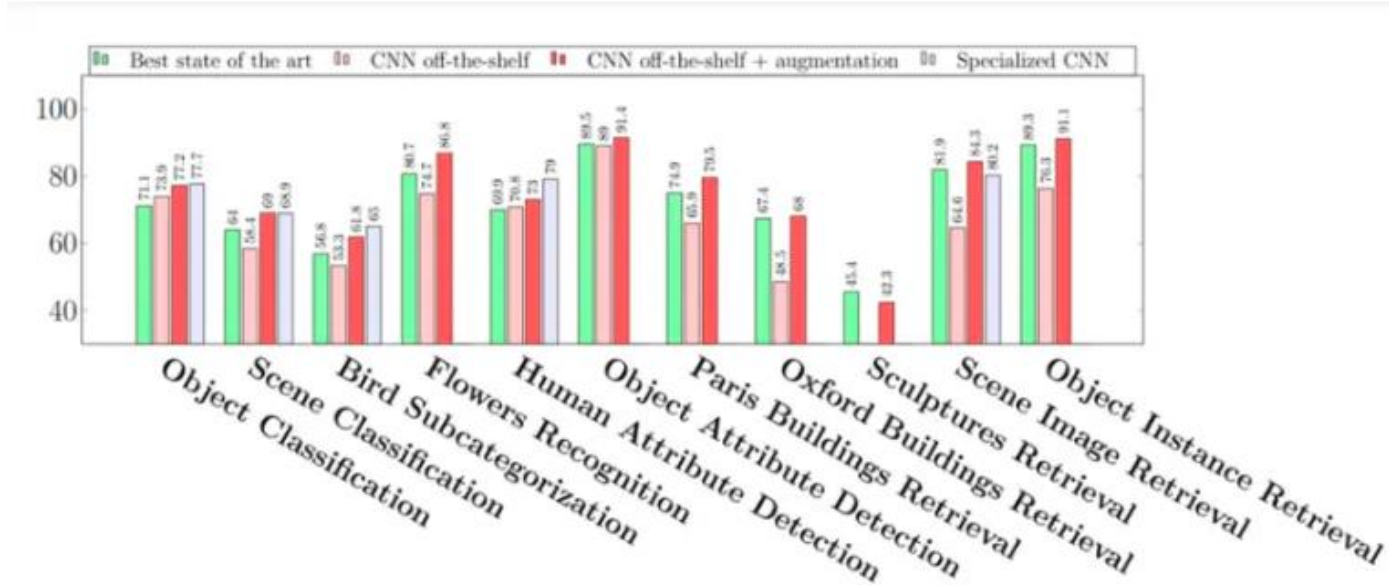


Figura 4: Desempeño de modelos Deep Learning específicamente entrenados sobre la tarea (verde) en comparación con modelos pre-entrenados (rojo, rosa).

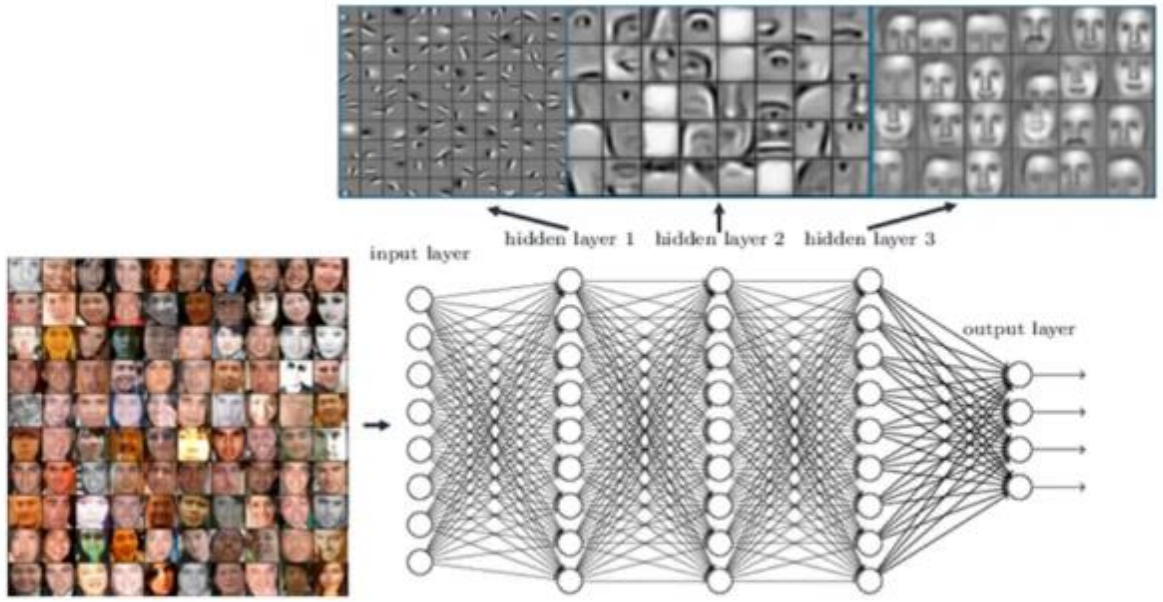


Figura 5 - Ejemplo de modelo de red de neuronas utilizado para reconocimiento facial.

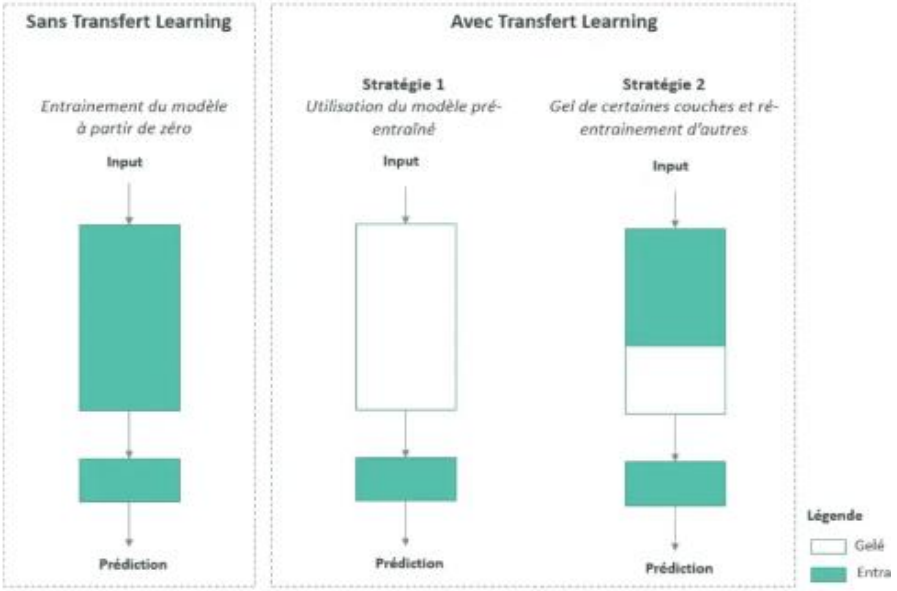


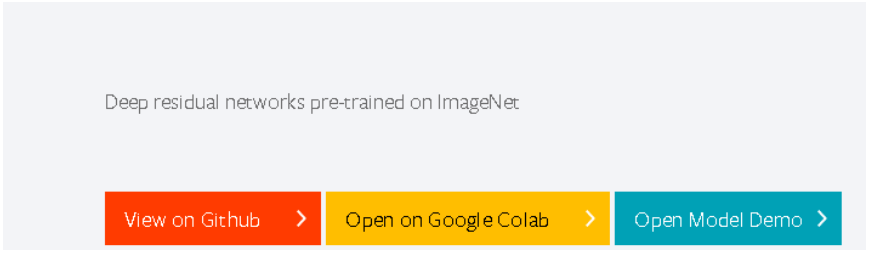
Figura 6: estrategia de Transfer Learning en Deep Learning

Entre los modelos más utilizados están:

- Computer Vision : VGG-16, VGG-19, ResNet-50
- NLP : Word2Vec, GloVe

PyTorch

https://pytorch.org/hub/pytorch_vision_resnet/

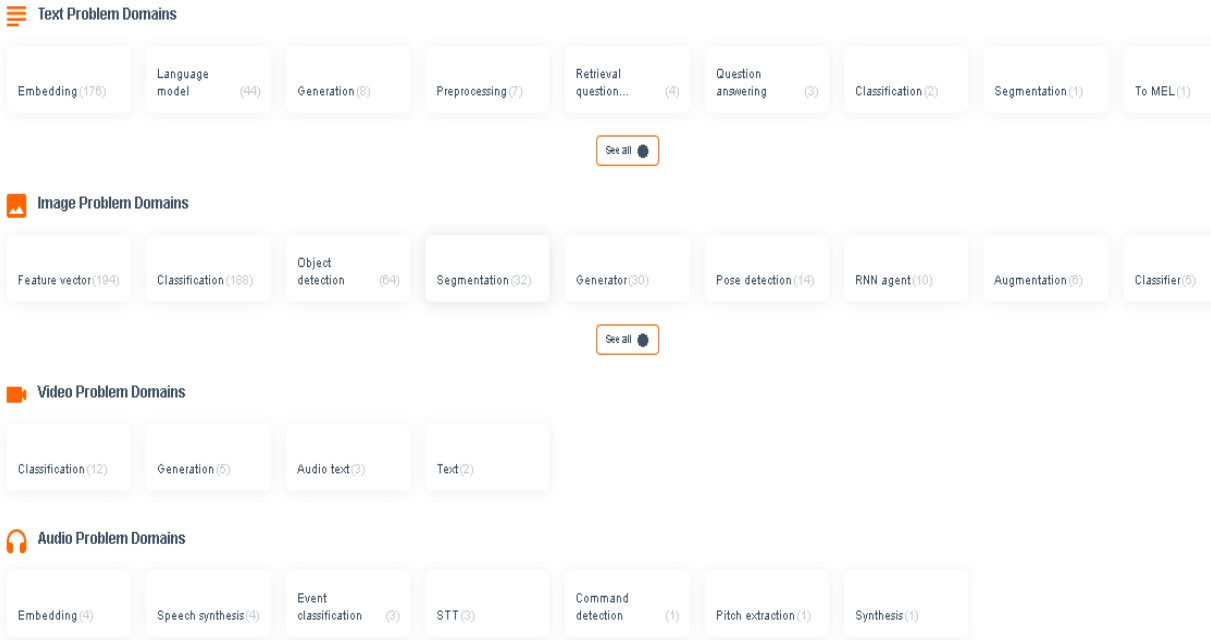


TensorFlow

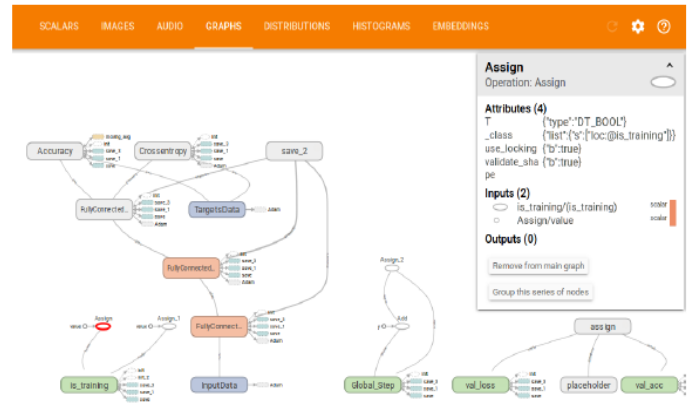
https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet/ResNet101https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet/ResNet101

https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet50/ResNet50

https://www.kaggle.com/datasets/keras/vgg16

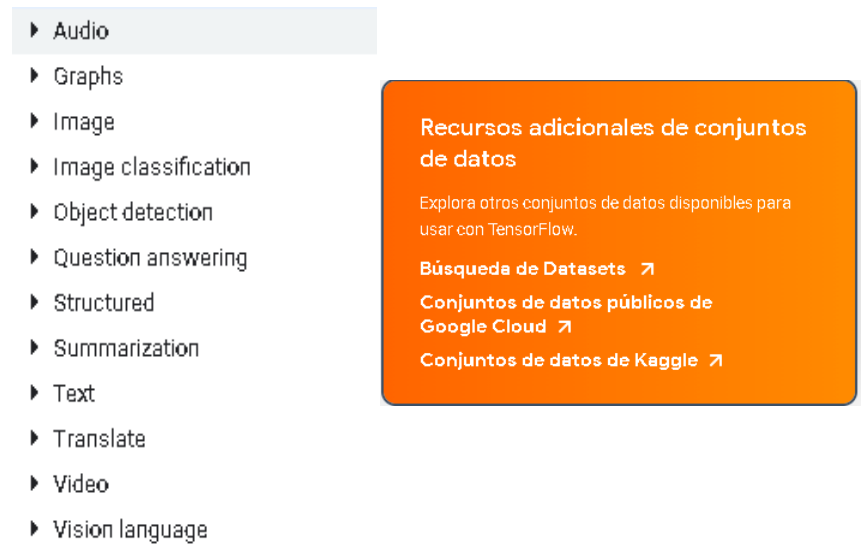


Modelos previamente entrenados en Tensor Flow



Visualización: tensorboard

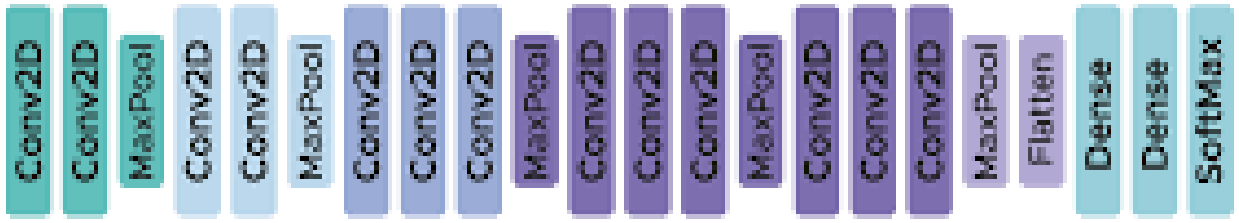
Data sets en Tensor Flow



Fuente: https://www.tensorflow.org/resources/models-datasets

2014 por Simonyan y Zisserman

TensorFlow

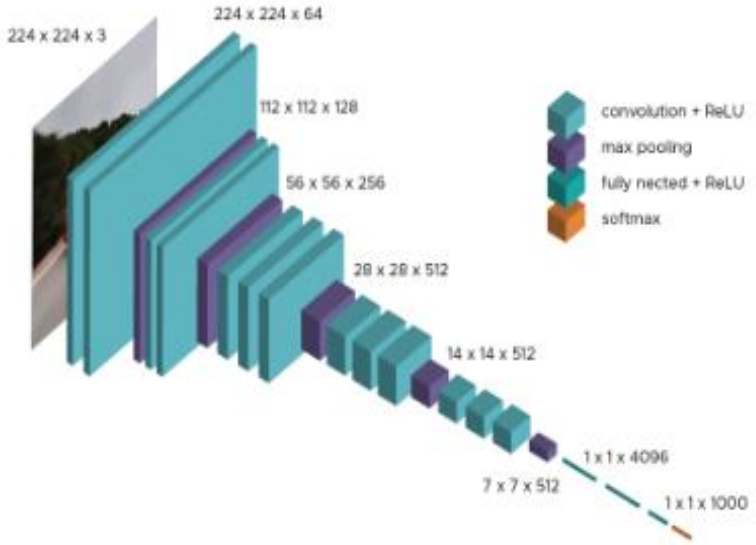


VGG16 Architecture

Como Usuarios: recuperamos los pesos de las capas de convolución y simplemente entrenamos las 3 últimas capas

https://datascientest.com/es/vgg-

Arquitectura del Algoritmo VGG16



Estructura Algoritmo VGG16

ENTRENAMIENTO

Pre-procesamiento: restar a cada píxel el valor RGB medio, calculado en el conjunto de entrenamiento

Input a 1ª capa de convolución: RGB de 224 x 224

El KERNEL, para todas las capas: 3×3

Max-Pooling, cada una: 2×2

Dos capas full connected de 4096 neuronas

Última capa: 1000 neuronas, softmax

PyTorch Get Started

RESNET

```python
import torch
model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet18', pretrained=True)
# or any of these variants
# model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet34', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet50', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet101', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet152', pretrained=True)
model.eval()
```

All pre-trained models expect input images normalized in the same way, i.e. mini-batches of 3-channel RGB images of shape `(3 x H x W)`, where `H` and `W` are expected to be at least `224`. The images have to be loaded in to a range of `[0, 1]` and then normalized using `mean = [0.485, 0.456, 0.406]` and `std = [0.229, 0.224, 0.225]`.

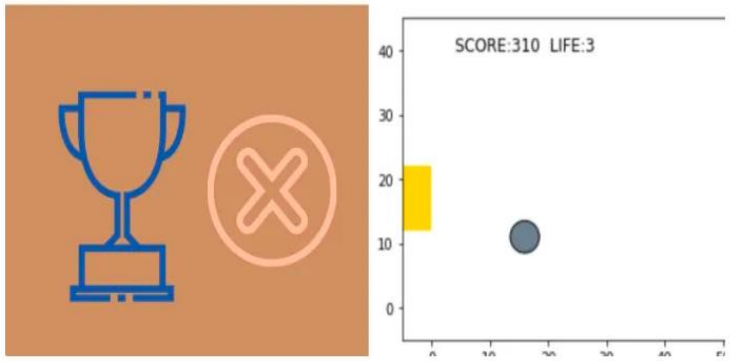
Model Description

Resnet models were proposed in "Deep Residual Learning for Image Recognition". Here we have the 5 versions of resnet models, which contains 18, 34, 50, 101, 152 layers respectively. Detailed model architectures can be found in Table 1. Their 1-crop error rates on imagenet dataset with pretrained models are listed below.

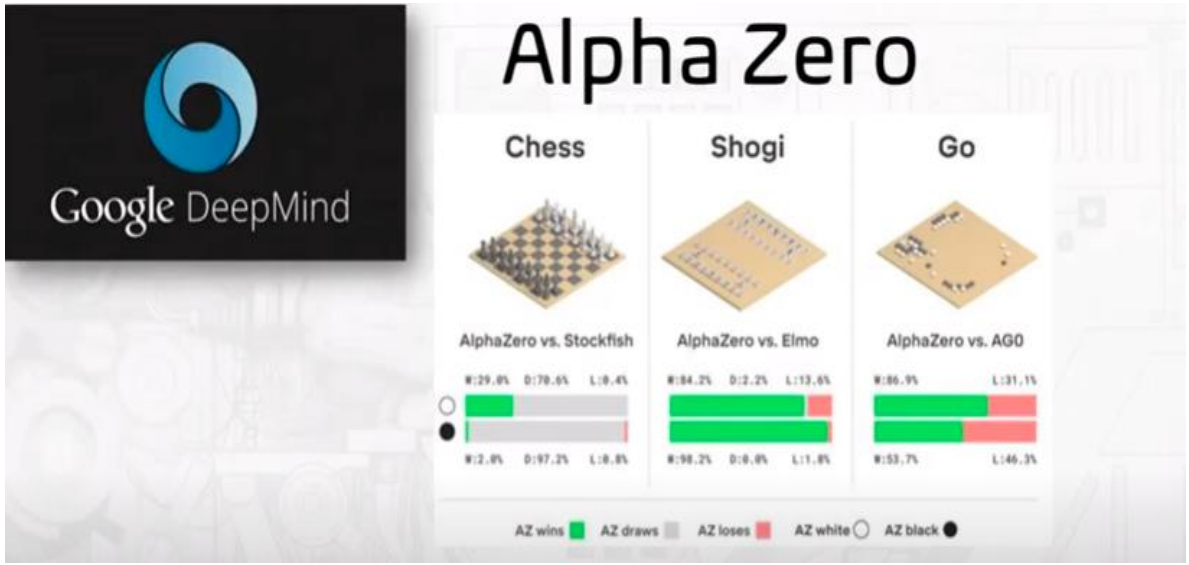
Model structure	Top-1 error	Top-5 error
resnet18	30.24	10.92
resnet34	26.70	8.58
resnet50	23.85	7.13
resnet101	22.63	6.44
resnet152	21.69	5.94

https://pytorch.org/hub/pytorch_vision_resnet/

NEURAL NETWORKS
REINFORCEMENT LEARNING



NEURAL NETWORKS REINFORCEMENT LEARNING

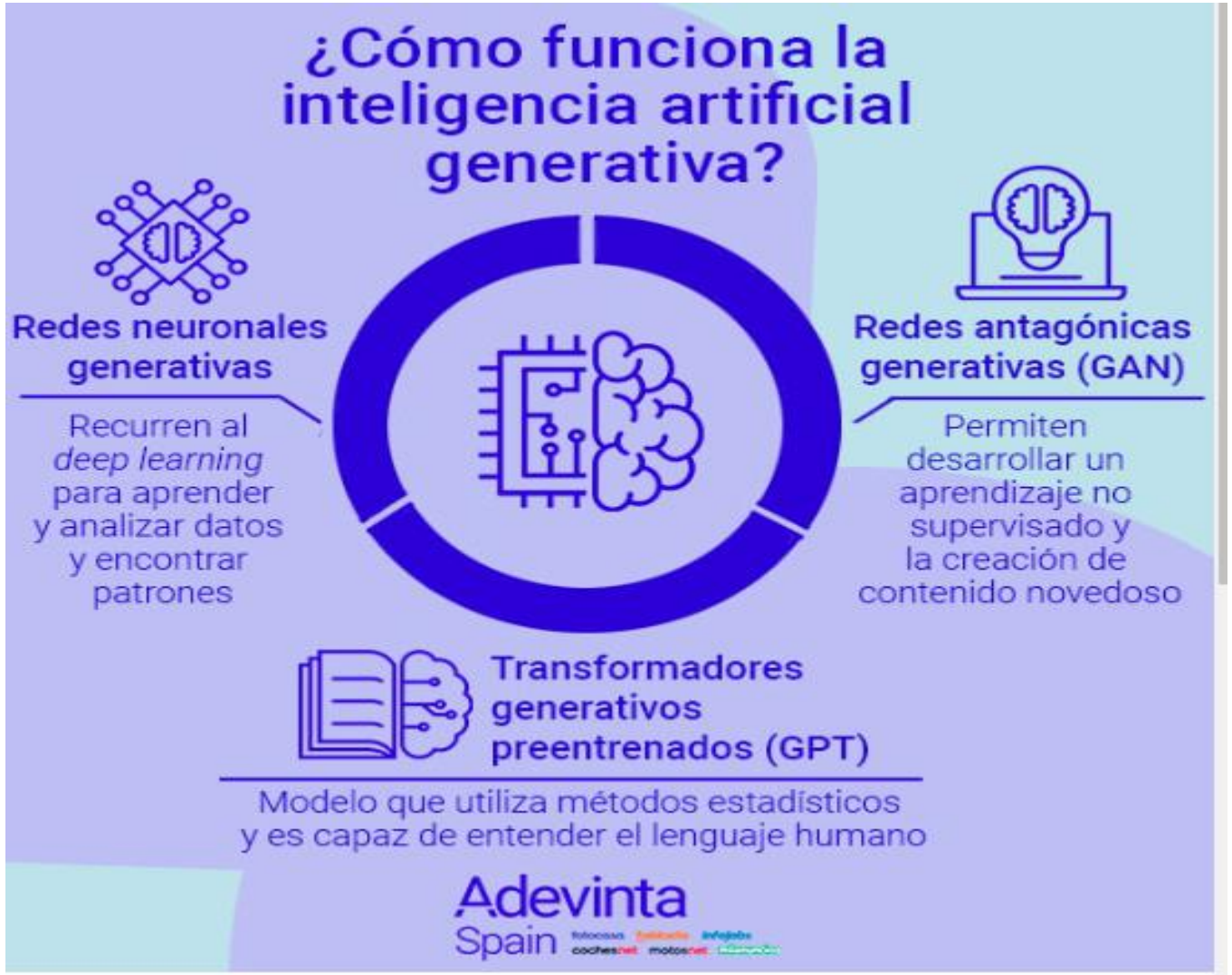


Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... & Lillicrap, T. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

REINFORCEMENT LEARNING

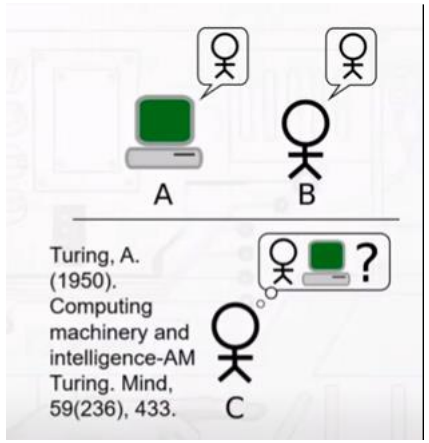
Google DeepMind

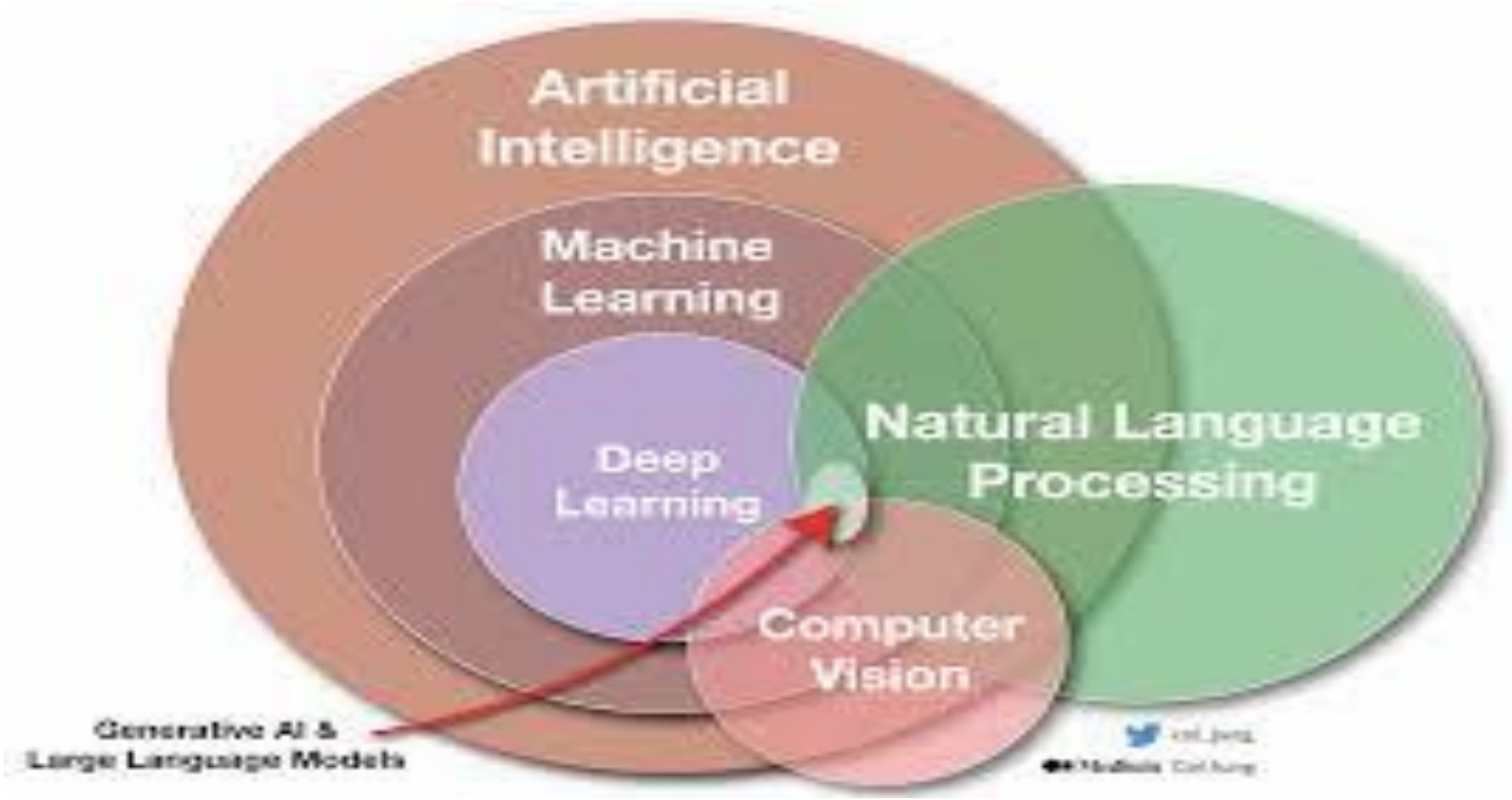
MODELOS DE IA GENERATIVA



Aprenden los patrones y la estructura de los datos de entrada para crear un contenido nuevo similar .

Framework de alto nivel para trabajar con modelos diversos:





ChatGPT & GPT-4 — How OpenAI Won the Natural Language Understanding War | by Col Jung | Apr, 2023

https://generativeai.pub/the-road-to-chatgpt-gpt-4-how-deep-learning-revolutionised-natural-language-processing-835d89560577

GENERATIVE ADVERSARIAL NETWORKS

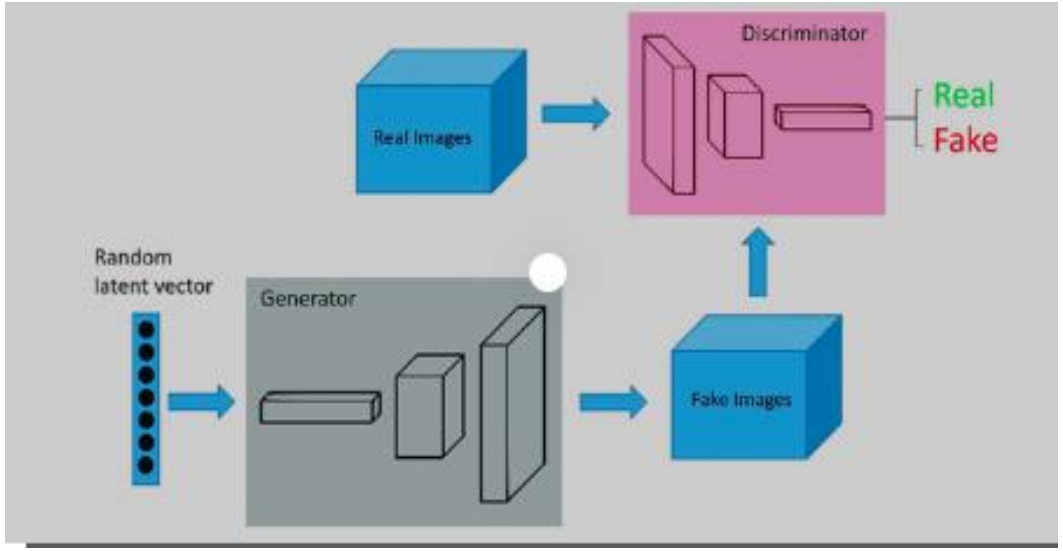
OPONEN DOS REDES CONVOLUCIONALES

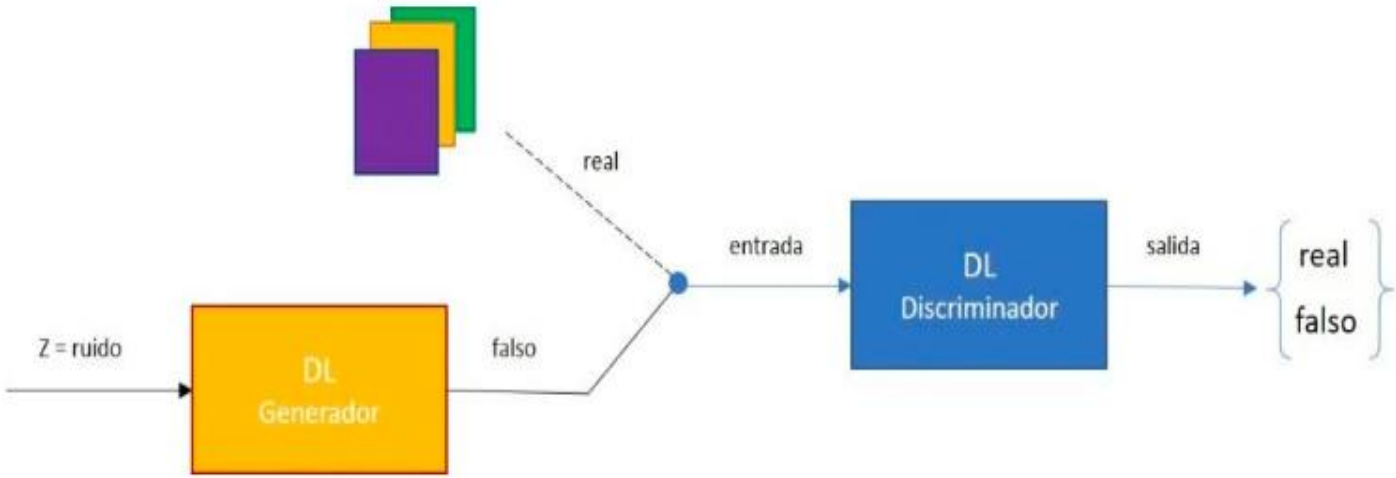
COMO SI JUGARAN AL GATO Y AL RATÓN

UNA DE ELLAS TIENE LA MISIÓN DE GENERAR “UN ROSTRO” LO MÁS REALISTA POSIBLE

MIENTRAS, LA OTRA SE ESPECIALIZA EN DETECTAR “FOTOS” QUE NO SON REALES

JUEGAN ENFRENTADAS, EVOLUCIONANDO AMBAS: “TRATAMIENTO DE IMÁGENES”





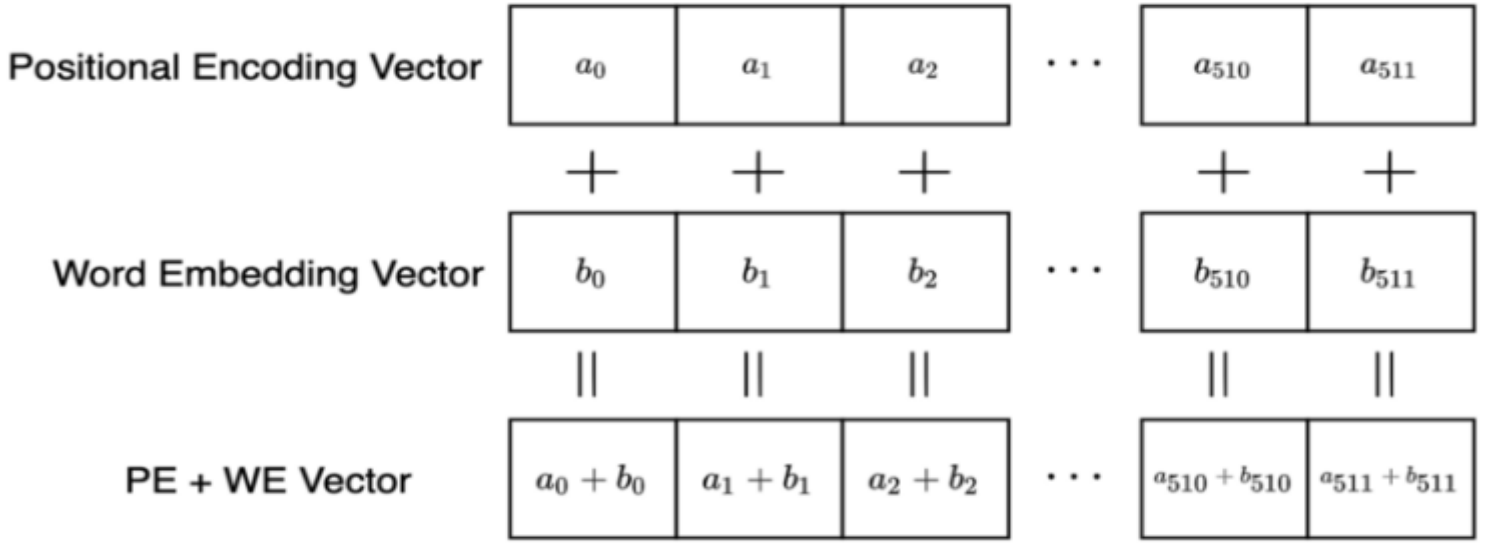
https://www.iartificial.net/redes-neuronales-generativas-adversarias-gans/

TRANSFORMERS

Aprendizaje profundo que prescinde de la Recurrencia y la Convolución

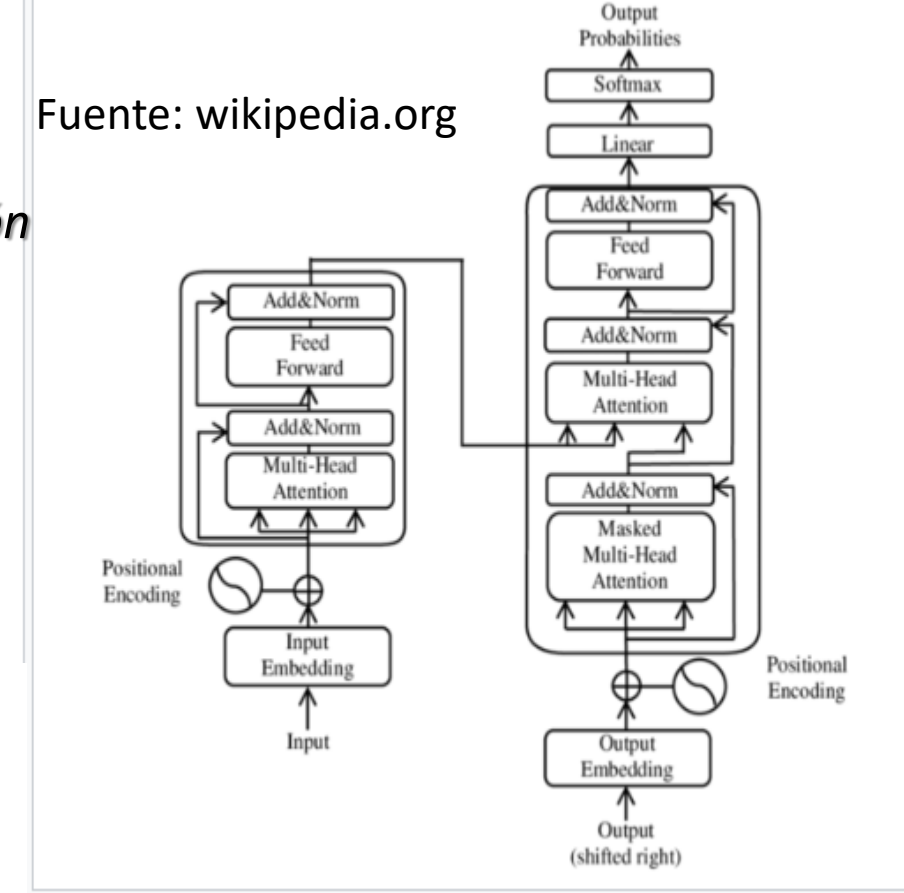
Basado en Mecanismos de atención

Codificación e Incrustación: Palabras a Tokens; Tokens a Vectores (word emdedding) más ¡positional enconding! (PE).
Posición relativa de los tokens en la secuencia.
Dimensión del original: 512!. La codificación posicional es una función sinusoidal



Fuente: /transformers-positional-encoding/

Fuente: wikipedia.org



Arquitectura del modelo transformador

$$PE_{(pos,\,2i)} = \sin(pos/10000^{(2i/L)}$$

[Tutorial de Pytorch]

Computer Science > Computation and Language

[Submitted on 12 Jun 2017 (v1), last revised 6 Dec 2017 (this version, v5

Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Usz … Kaiser, Illia Polosukhin

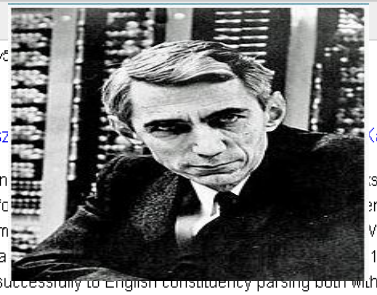
The dominant sequence transduction models are based on … s in an encoder-decoder configuration. The best performing models also connect the … We propose a new simple network architecture, the Transfo … ensing with recurrence and convolutions entirely. Experiments on two machine translat … being more parallelizable and requiring significantly less tim … VMT 2014 English-to-German translation task, improving over the existing best results … English-to-French translation task, our model establishes a … 1.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of t … Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Comments: 15 pages, 5 figures

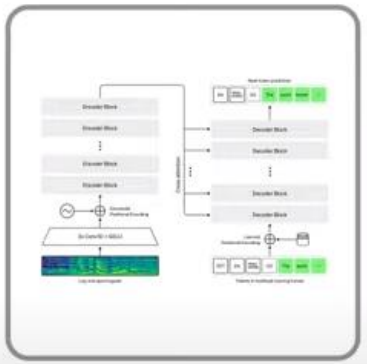
Subjects: **Computation and Language (cs.CL)**; Machine Learning (cs.LG)

Cite as: arXiv:1706.03762 **[cs.CL]**

sion)



TRANSFORMER (2017)



TRANSFORMER:
RED NEURONAL NATIVA PARA PROCESAMIENTO DE LENGUAJE NATURAL
APRENDE CONTEXTO BASADO EN DATOS QUE CONTIENEN SECUENCIAS

MECANISMOS DE ATENCIÓN

POR CASO, CHAT GPT (Generative Pre-training Transformer)

ES UN "GRAN MODELO DE LENGUAJE"

NO ES UN MODELO DE RAZONAMIENTO

SU DESEMPEÑO PARA RAZONAR:

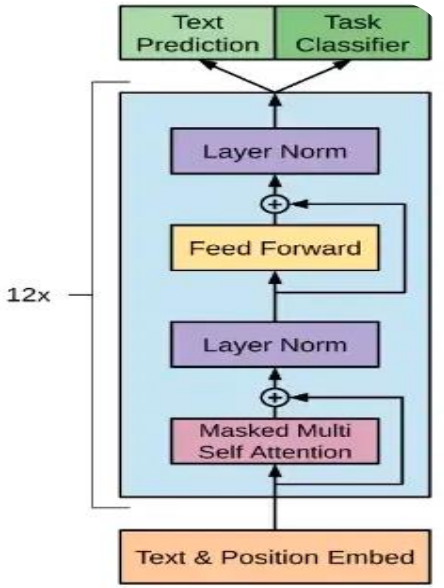
DEDUCIR

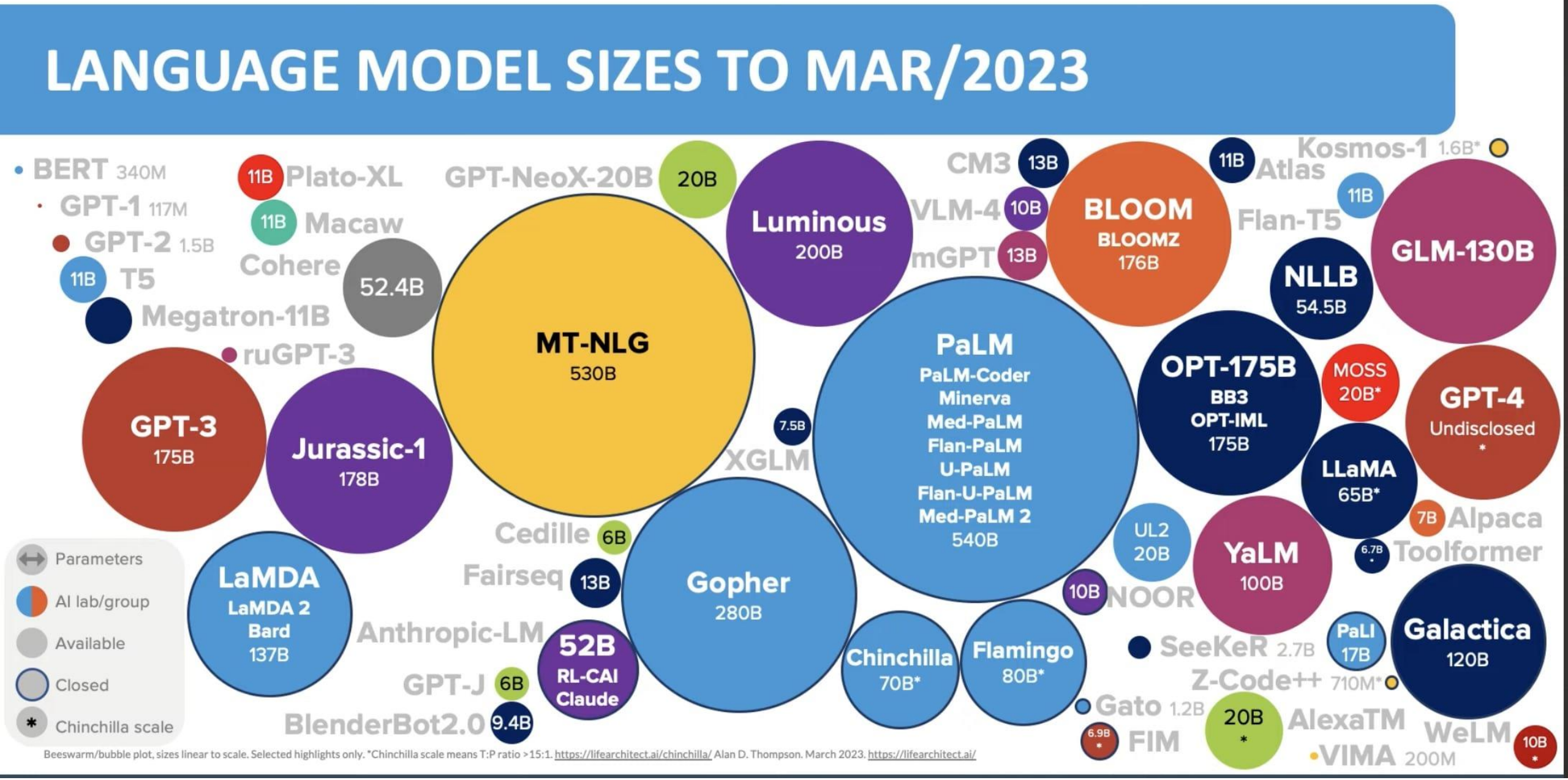
ABDUCIR

~~INDUCIR~~

SU VENTAJA COMPARATIVA: UX

ESTÁ BASADO EN TEXTO





https://lifearchitect.ai/models/

ChatGPT

ChatGPT es un prototipo de chatbot de inteligencia artificial desarrollado en 2022 por OpenAI que se especializa en el diálogo. El chatbot es un gran modelo de lenguaje ajustado con técnicas de aprendizaje tanto supervisadas como de refuerzo. Se basa en el modelo GPT-3.5 de OpenAI, una versión mejorada de GPT-3.

ChatGPT se lanzó el 30 de noviembre de 2022[1] y ha llamado la atención por sus respuestas detalladas y articuladas, aunque se ha criticado su precisión fáctica. El servicio se lanzó inicialmente como gratuito para el público, con planes de monetizarlo más adelante. El 4 de diciembre, OpenAI calculaba que ChatGPT ya tenía más de un millón de usuarios.[2] [3]

https://chat.openai.com/chat

PROMPT ENGINEERING

https://openai.com/research/instruction-following

RETROALIMENTACIÓN HUMANA: REINFORCEMENT LEARNING FROM HUMAN FEEDBACK (RLHF)

IN CONTEXT LEARNING (ICL)

FINE TUNING

ALUCINACIONES: CONTENIDO FALSO

POLÍTICA DE VALORES Y PREFERENCIAS

CONTENIDO NO DESEADO

CONTENIDO PELIGROSO

¡GRACIAS!