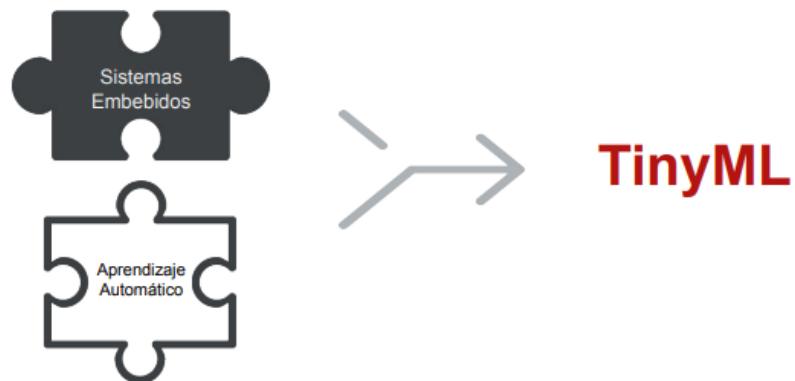
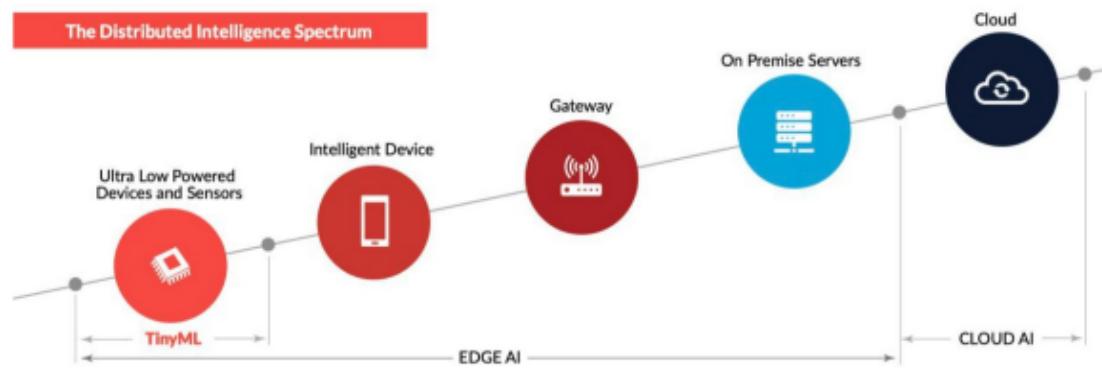


¿Qué es el Aprendizaje Automático Diminuto (Tiny ML)? El aprendizaje automático diminuto (TinyML) es un campo de rápido crecimiento de tecnologías y aplicaciones de aprendizaje automático incluyendo algoritmos, hardware y software capaz de realizar análisis de datos de sensores en el dispositivo con un consumo de energía extremadamente bajo. **Ej.** Utilidades, venta al por menor, transporte y logística, ciudades inteligentes, industria y manufactura, cuidado de la salud, consumidor, bancos y finanzas y agricultura.



Edge AI (o Edge ML): es el procesamiento de algoritmos de Inteligencia Artificial en el borde, es decir, en los dispositivos de los usuarios. El concepto se deriva de Edge Computing, que parte de la misma premisa: los datos se almacenan, procesan y administran directamente en los puntos finales de Internet de las cosas (IoT). **Ej.** Control automático de carros.

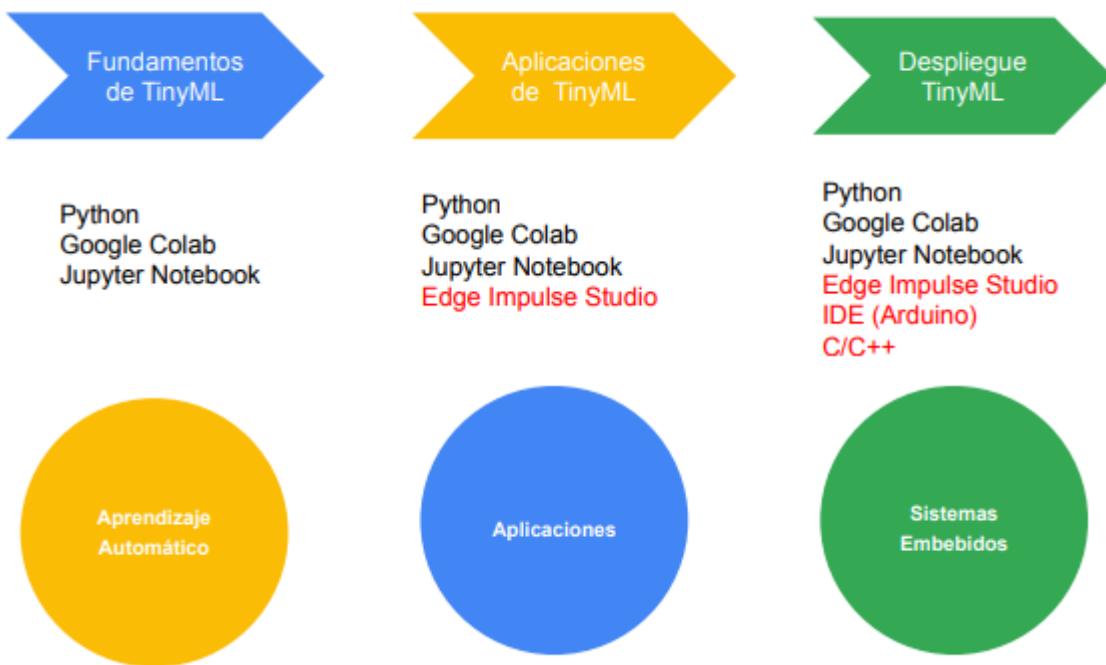


TinyML: es un subconjunto de EdgeML, donde los sensores generan datos con un consumo de energía ultrabajo (baterías), para que finalmente podamos implementar el aprendizaje automático de forma continua ("siempre en los dispositivos"). **Ej.** Motion & Biometric de un reloj inteligente.



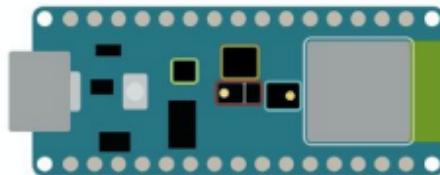
Aplicaciones del TinyML: en hogares, oficinas e industrias. Ej. Mantenimiento Predictivo, Seguimiento de activos y monitoreo, Sensado de humanos y animales.

Áreas de fundamento para construir aplicaciones TinyML



Software: Machine Learning (TensorFlow - Keras), Programming environments (Jupyter and Colab) y Edge Impulse Studio.

Hardware: Arduino 33 BLE Sense, Sensors, Raspberry y ESP 32
TinyML Kit

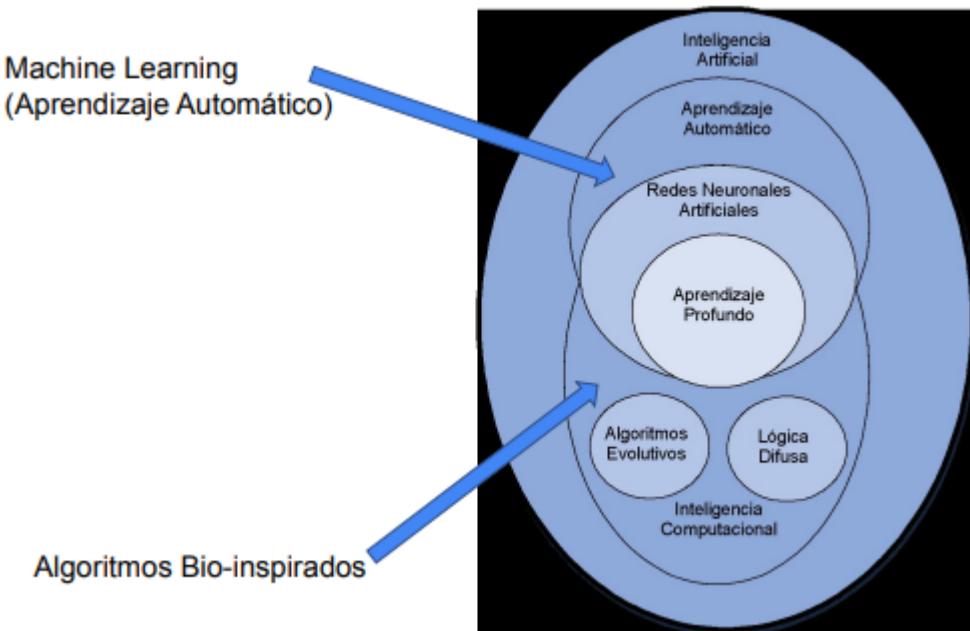


- ◆ Color, brightness, proximity and gesture sensor
- ◆ Digital microphone
- ◆ Motion, vibration and orientation sensor
- ◆ Temperature, humidity and pressure sensor
- ◆ Arm Cortex-M4 microcontroller and BLE module

Ej. Visión computacional, movimiento y KWS

Inteligencia artificial: El término se le atribuye a John McCarthy (1955) : "Es la ciencia y la ingeniería de hacer máquinas inteligentes, especialmente programas informáticos. Está relacionado con la tarea de usar computadoras para comprender la inteligencia humana, pero la IA no tiene que limitarse a métodos que sean biológicamente observables.". Ej. La Inteligencia Artificial de Nivel Humano o Inteligencia Artificial General.

Inteligencias específicas: Biomimetic locomotion, Sound recognition, Image recognition y Natural Language Processing.



Inteligencia Biológica: Enjambre de Partículas, Algoritmos de Hormigas, Algoritmos de Bacterias y Algoritmos Evolutivos.

Machine Learning: el aprendizaje automático es un subconjunto de la inteligencia artificial que tiene la capacidad de "aprender" (es decir, mejorar progresivamente el rendimiento en una tarea específica) con datos, sin ser programado explícitamente.

Aplicaciones en Procesamiento de Imágenes: Enfoque Top-Down, Estimación de Postura, Enfoque Bottom-Up.

- **Un objeto:** clasificación y localización.
- **Varios objetos:** detección y segmentación semántica.

Aplicaciones en Señales Secuenciales: habla, sonido, sensores iniciales y series temporales.

Aplicaciones del Procesamiento Natural de Lenguaje: El procesamiento de lenguaje natural, abreviado PLN –en inglés, Natural Language Processing, NLP– es un campo de las ciencias de la computación, de la inteligencia artificial y de la lingüística que estudia las interacciones entre las computadoras y el lenguaje humano.

Recolección de Datos y entrenamiento del modelo

- **Modelo entrenado:** modelo inicial + datos de entrada + datos de salida
- Modelo entrenado + dato de entrada = salida

Selección del Modelo

https://cezannec.github.io/Convolutional_Neural_Networks/

<http://jalammar.github.io/illustrated-transformer/>

Utilización del modelo

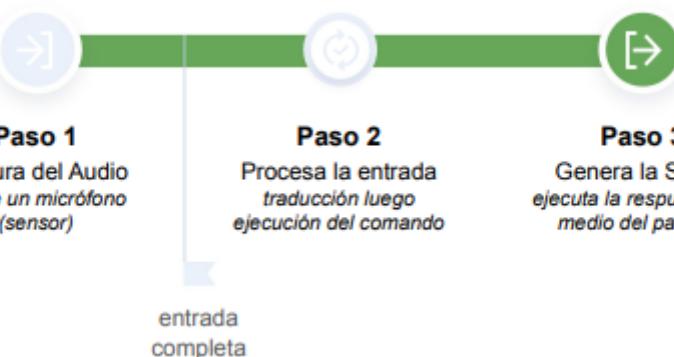
Centro de datos: Todas las capacidades de los ejemplos anteriores, requirió una cantidad notable de energía y capacidades informáticas, entonces lo que las empresas están haciendo, es tomar todas estas computadoras, empaquetándolos en centros de datos, que están siendo dedicados a proporcionar las capacidades cómputo necesarias para el aprendizaje automático.

TPU y GPU: Para poder proporcionar la capacidad de ML, empresas como Google están creando procesadores especiales como las TPU (Unidades de procesamiento de tensores) además GPU

(Unidades de procesamiento de gráficos) que producen empresas como NVIDIA y AMD. Ambos sistemas informáticos son capaces de ejecutar el aprendizaje automático extremadamente rápido.

Centro de Datos	Teléfono Inteligente
Consumo alto de energía	Consumo bajo de energía
Ancho de banda grande	Ancho de banda pequeño
Alta latencia	Baja latencia
Más costosa la trasmisión de información	Menos costosa la trasmisión de información
Se puede compartir información sensible	No se comparten datos sensible

Procesamiento

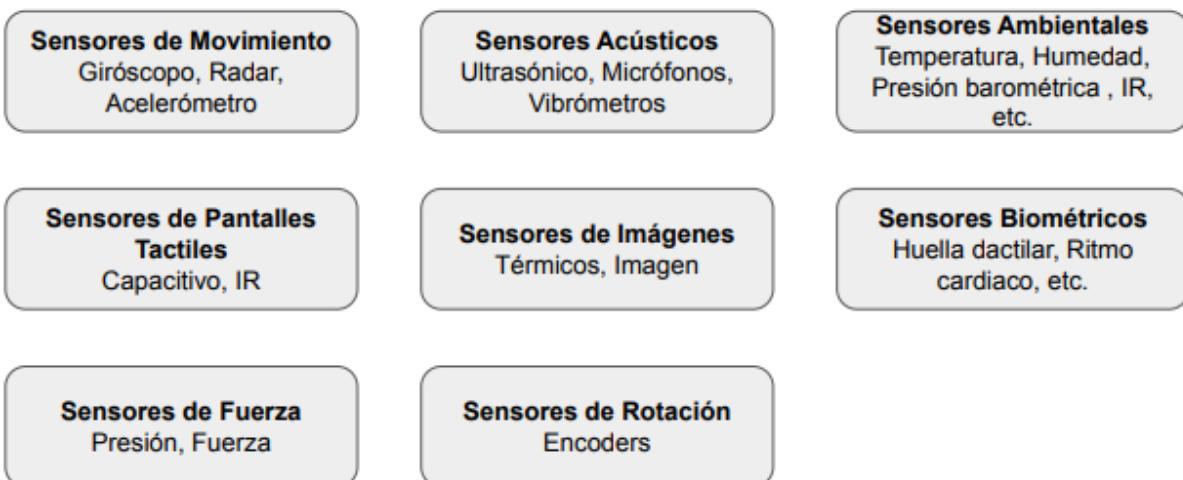


Comparación de Consumo de Energía según el tamaño del procesamiento

Procesamiento grande	Procesamiento pequeño	Procesamiento diminuto
BIG GPU / CPU 561mm ² 300W NVIDIA Tesla K80	SMALL Mobile SoC 83mm ² 3.64W Apple A12	TINY Apple 0778 30mm ² Procesador de Decisión Neuronal Siempre-encendido deep learning reconocimiento de audio/habla Ultra low power, 128KB SRAM, 12-pin, 2.52mm ² 140 µW Syntiant NDP100

La tendencia del uso de MCU (Unidad microcontroladora) ha ido en aumento desde el 2016, alcanzando a 250 mil millones de unidades (2023). Esto ha conllevado a la reducción de costos en los MCU.

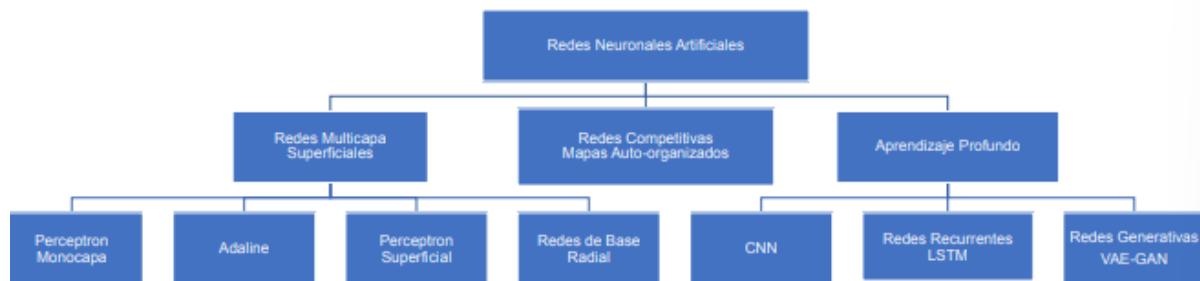
Captura de la Entrada por Medio de Sensores



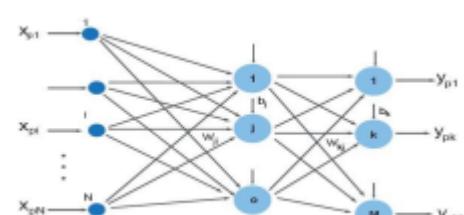
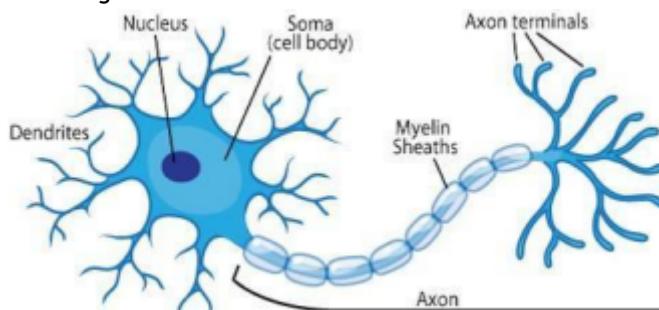
Relación del uso del MCU con el TinyML

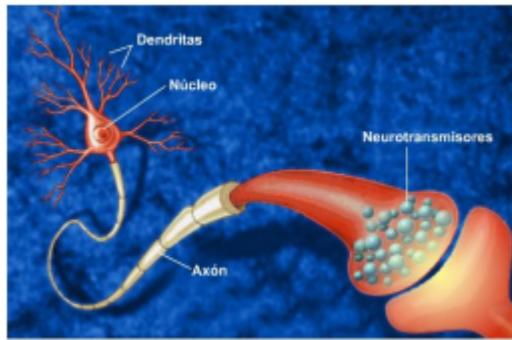


Introducción a las Redes Neuronales Artificiales (Deep Learning)



¿Qué son las redes neuronales artificiales? Conjunto de elementos de procesamiento que emulan algunas características de funcionamiento del cerebro humano.

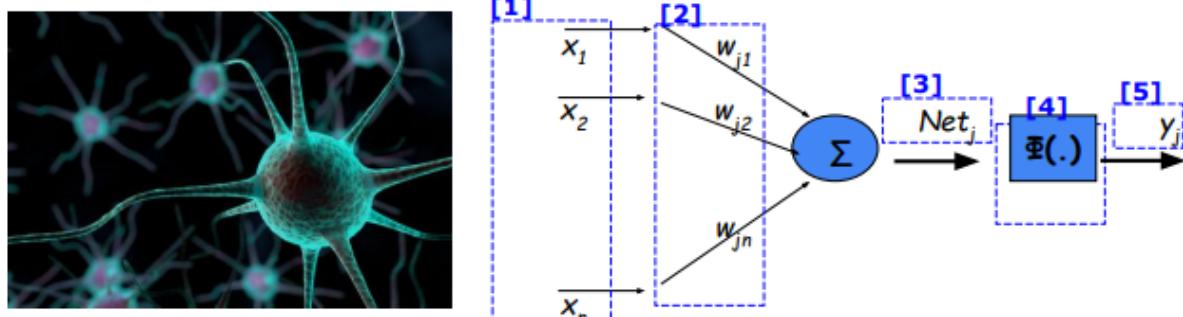




Aspectos funcionales de una neurona biológica:

1. Los elementos de proceso (neuronas) reciben las señales de entrada.
2. Las señales pueden ser modificadas por los pesos sinápticos.
3. Los elementos de proceso "suman" las entradas afectadas por las sinapsis.
4. Las neuronas transmiten una señal de salida.
5. La salida del elemento de proceso puede ir a muchas neuronas A.

La Neurona Artificial: Modelo propuesto a partir de la neurona biológica.

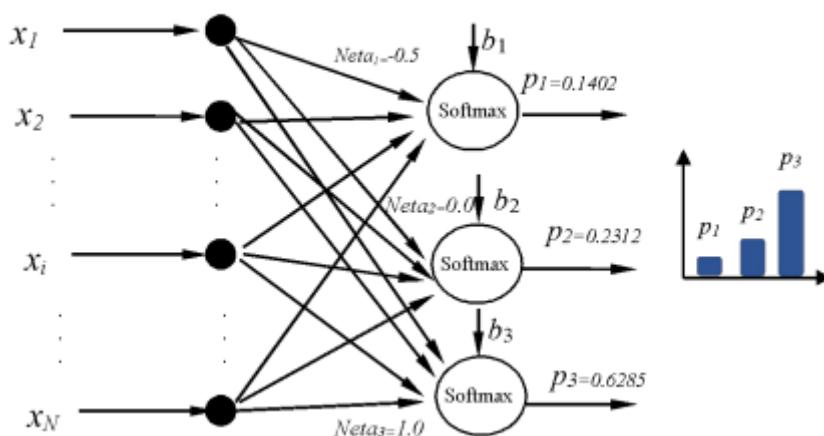


Pesos sinápticos: La matriz de pesos sinápticos es el parámetro que se utiliza para representar la conexión entre las neuronas, de manera similar a lo que hace la sinapsis en las neuronas biológicas (w_{ij}). Este término representa el valor de la conexión que va a la neurona iésima, proveniente de la neurona jésima.

Entrada neta: La magnitud que denominamos Neta representa la entrada total de información o estímulo que recibe una neurona, proveniente de fuentes externas o de otras neuronas a las cuales está conectada. Este valor lo evaluamos en la función de activación para generar la salida total de la neurona, tal como lo vemos en la expresión que sigue:

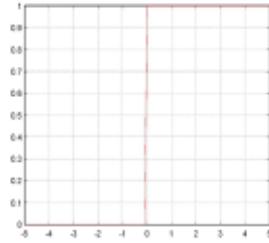
$$\begin{aligned} \text{Neta}_j &= \sum_{i=1}^n w_{ji}x_i + b_j \\ y_j &= \Phi(\text{Net}_j) \end{aligned}$$

Funciones de activación



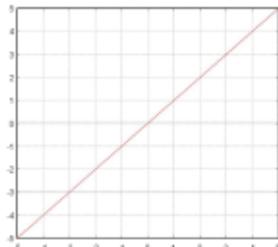
Escalón

$$f(neta) = \begin{cases} 1 & \text{si neta} \geq 0 \\ 0 & \text{si neta} < 0 \end{cases}$$



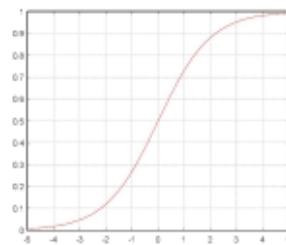
Lineal

$$f(neta) = neta$$



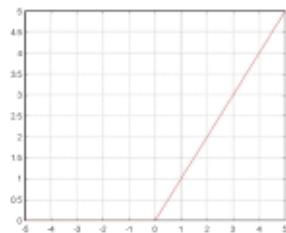
Sigmoidal

$$f(neta) = \frac{1}{1 + e^{-neta}}$$



ReLU

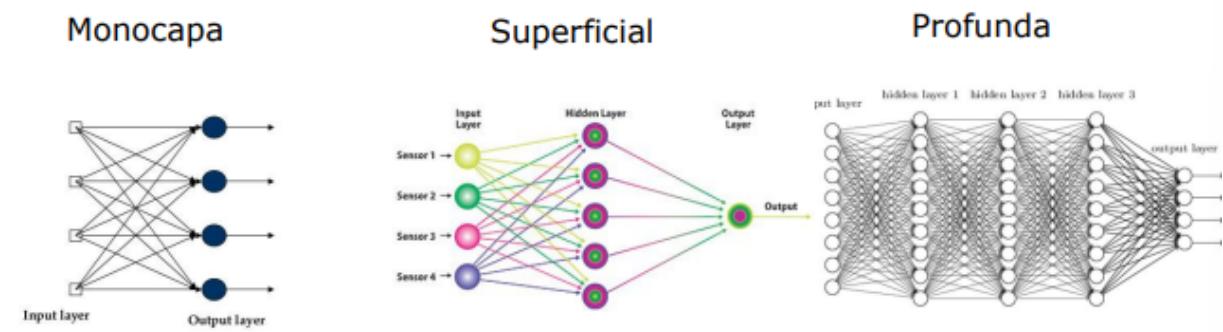
$$f(neta) = \max(0, neta)$$



Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = \max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Copyright © Sebastian Raschka 2016
<http://sebastianraschka.com>

Capas



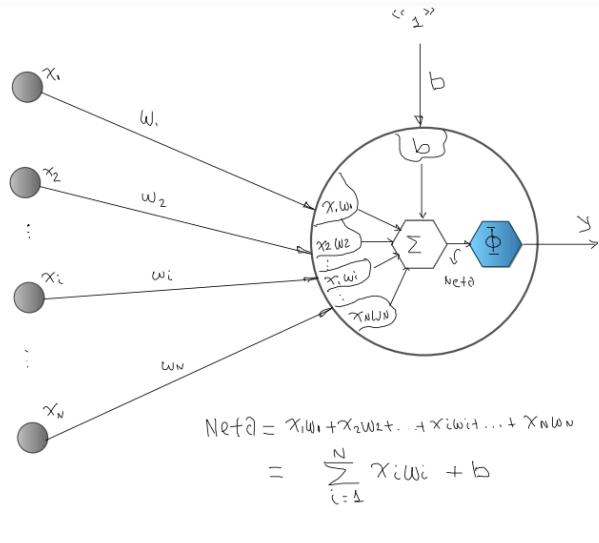
Semejanza con el cerebro: Emula el funcionamiento del cerebro en dos aspectos:

1. El conocimiento es obtenido por la red a través de un proceso de aprendizaje.
2. Las conexiones entre neuronas conocidas como pesos sinápticos son utilizadas para almacenar dicho conocimiento.

Las RNA se desarrollan como generalizaciones de modelos matemáticos del conocimiento humano o de la biología neuronal, con base en los siguientes considerandos:

- Elementos de procesamiento simples (Neuronas)
- Conectividad
- Enlaces de conexión con pesos
- Función de Activación

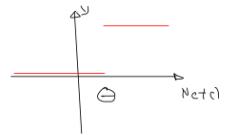
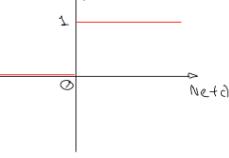
Ejercicios prácticos



Φ = Función de activación

$$y = \Phi(\text{Net}\delta)$$

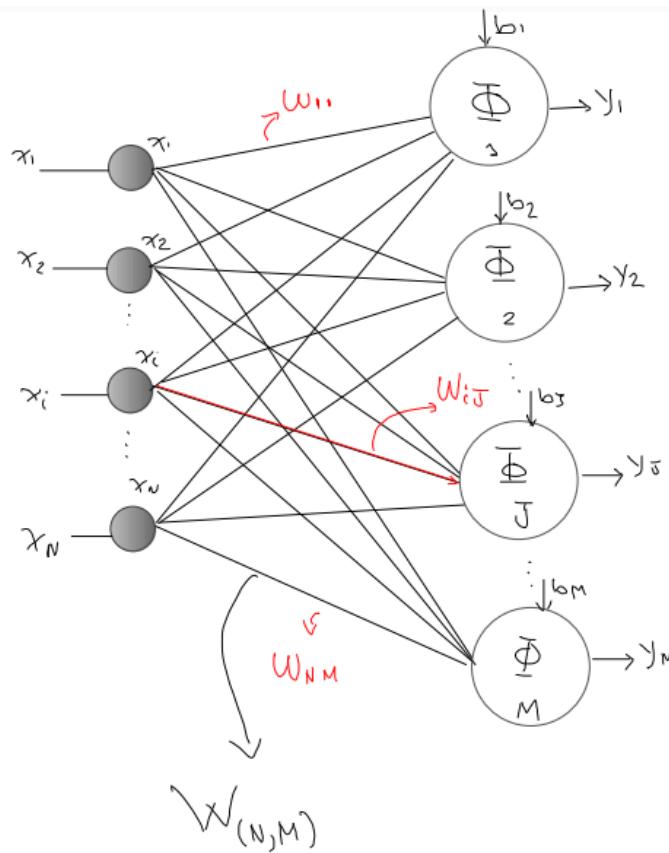
→ Escalón



$$y = \begin{cases} 1 & \text{if } \text{Net}\delta > 0 (\oplus) \\ 0 & \text{if } \text{Net}\delta \leq 0 (\ominus) \end{cases}$$

$$\sum_{i=1}^N x_i w_i \geq \Theta \Rightarrow \sum_{i=1}^N \underbrace{x_i w_i}_{\text{Net}\delta} - \Theta \geq 0$$

$$\sum_{i=1}^N x_i w_i \geq \Theta \Rightarrow \sum_{i=1}^N \underbrace{x_i w_i}_{\text{Net}\delta} - \Theta \geq 0$$



$$W_{N,M} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1M} \\ w_{21} & w_{22} & \dots & w_{2M} \\ \vdots & \vdots & & \vdots \\ w_{N1} & w_{N2} & \dots & w_{NM} \end{bmatrix}$$

$$B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix}$$

$$\text{Net}_j = \mathbf{W}^T \mathbf{x} + \mathbf{B}$$

$(3 \times 1) \quad (3 \times 4)(4 \times 1) \quad (3)$
 $\qquad\qquad\qquad \underbrace{\qquad\qquad\qquad}_{3 \times 1} \qquad\qquad\qquad \underbrace{\qquad\qquad\qquad}_{3 \times 1}$

y_1

y_2

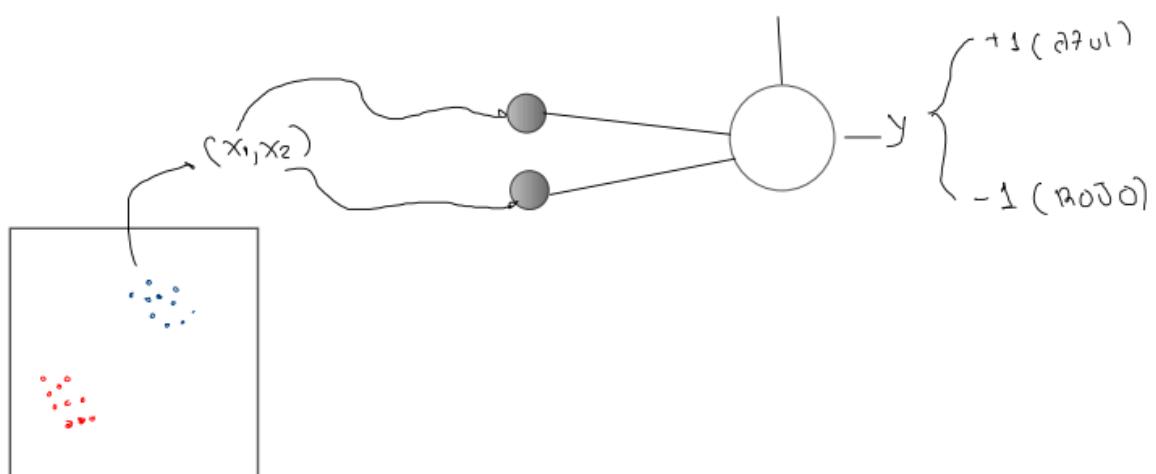
y_3

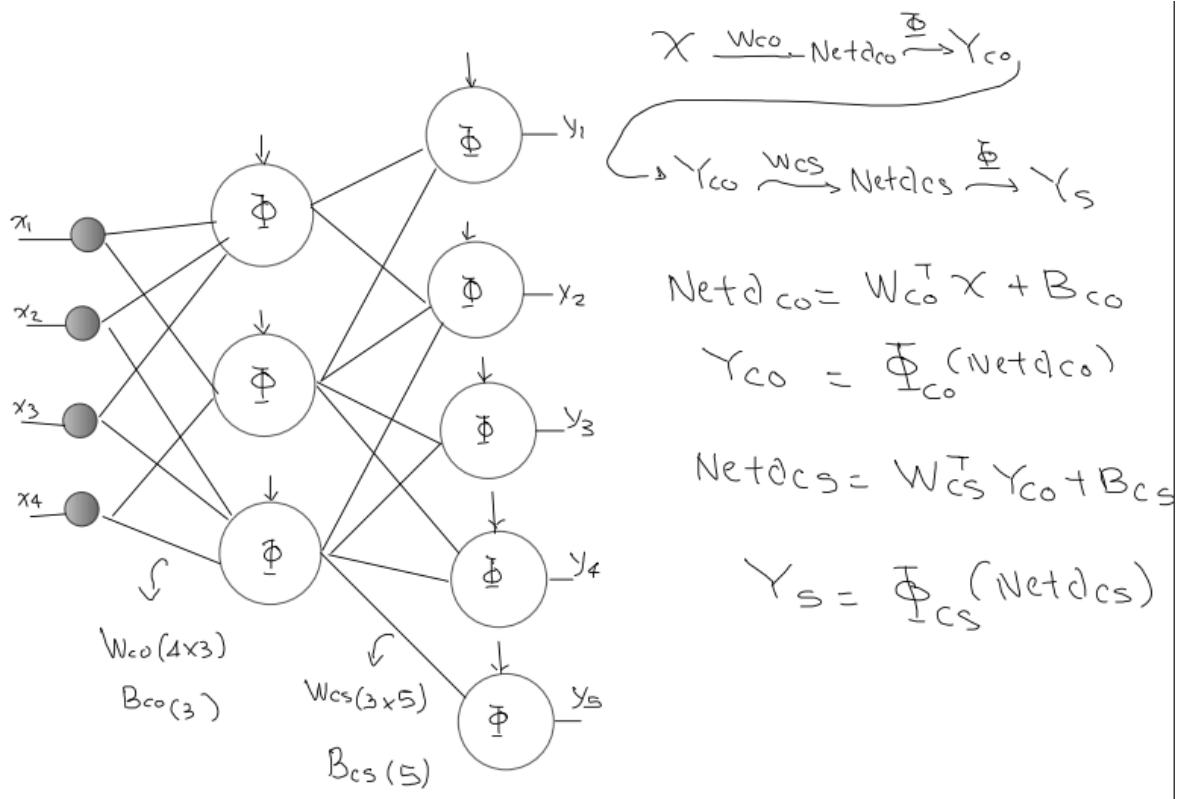
$W(4 \times 3)$

$B(3)$

$$Y = \Phi(\text{Net}_j)$$

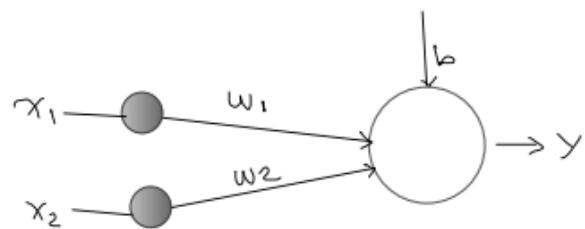
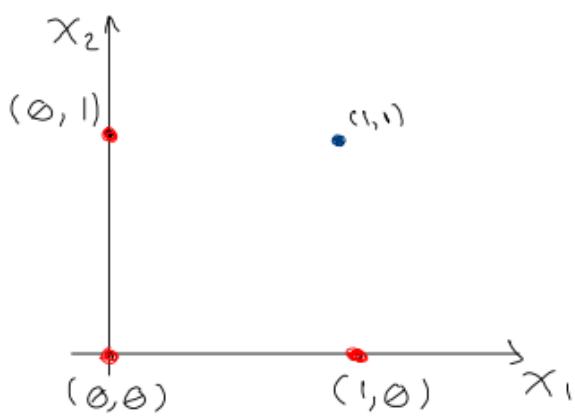
$(3 \times 1) \quad (3 \times 1)$

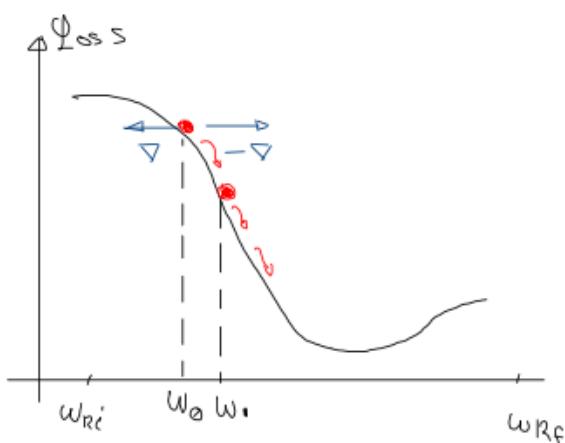




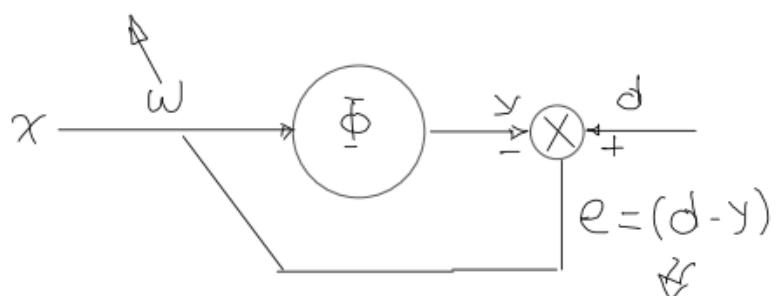
x_1	x_2	x_1 and x_2
0	0	0
0	1	0
1	0	0
1	1	1

$$\begin{aligned} 0 &\rightarrow \text{Rock} \\ 1 &\rightarrow \text{Axel} \end{aligned}$$



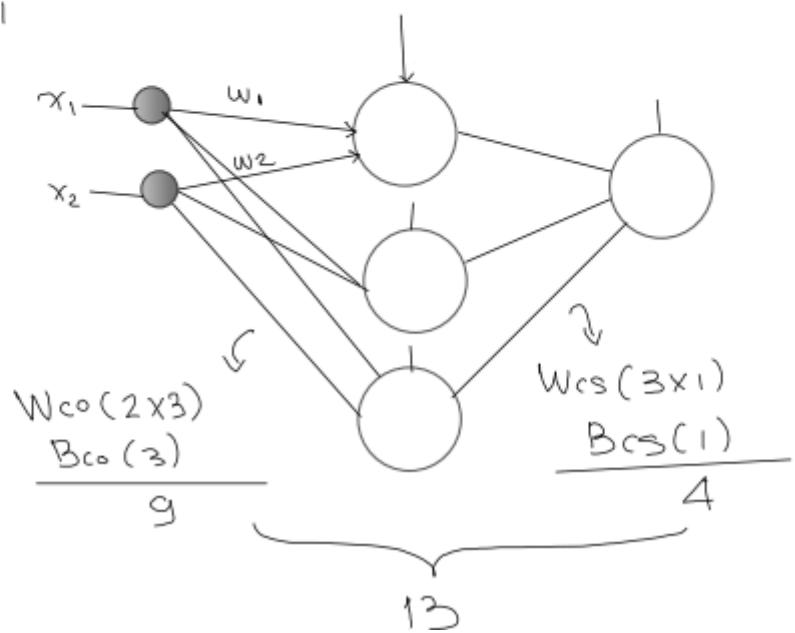
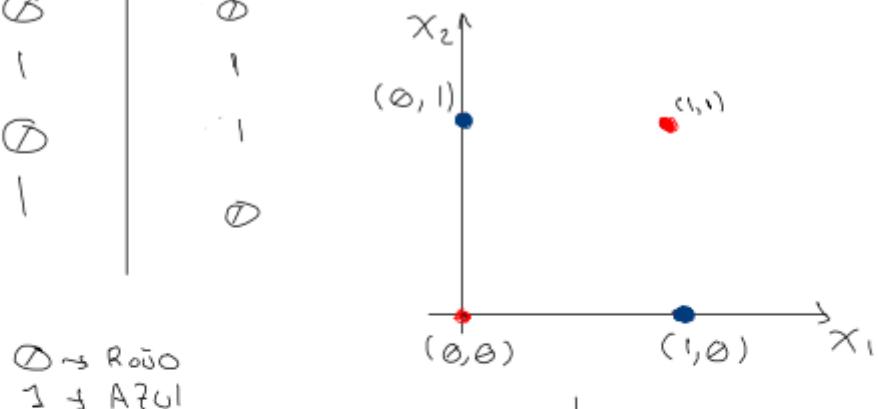


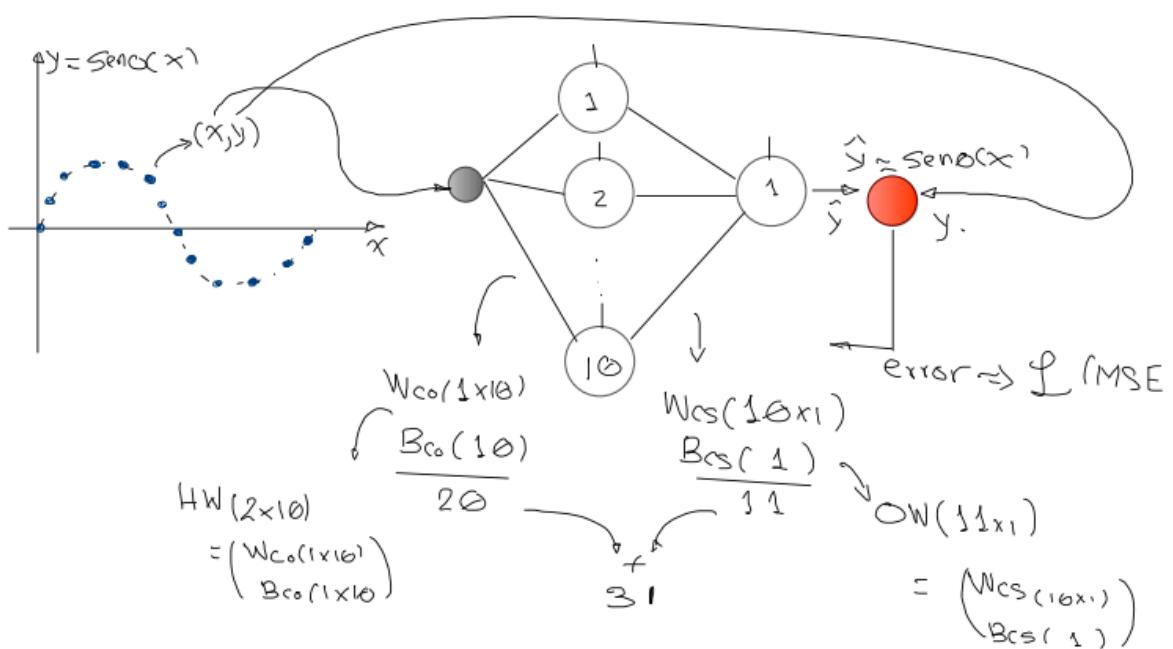
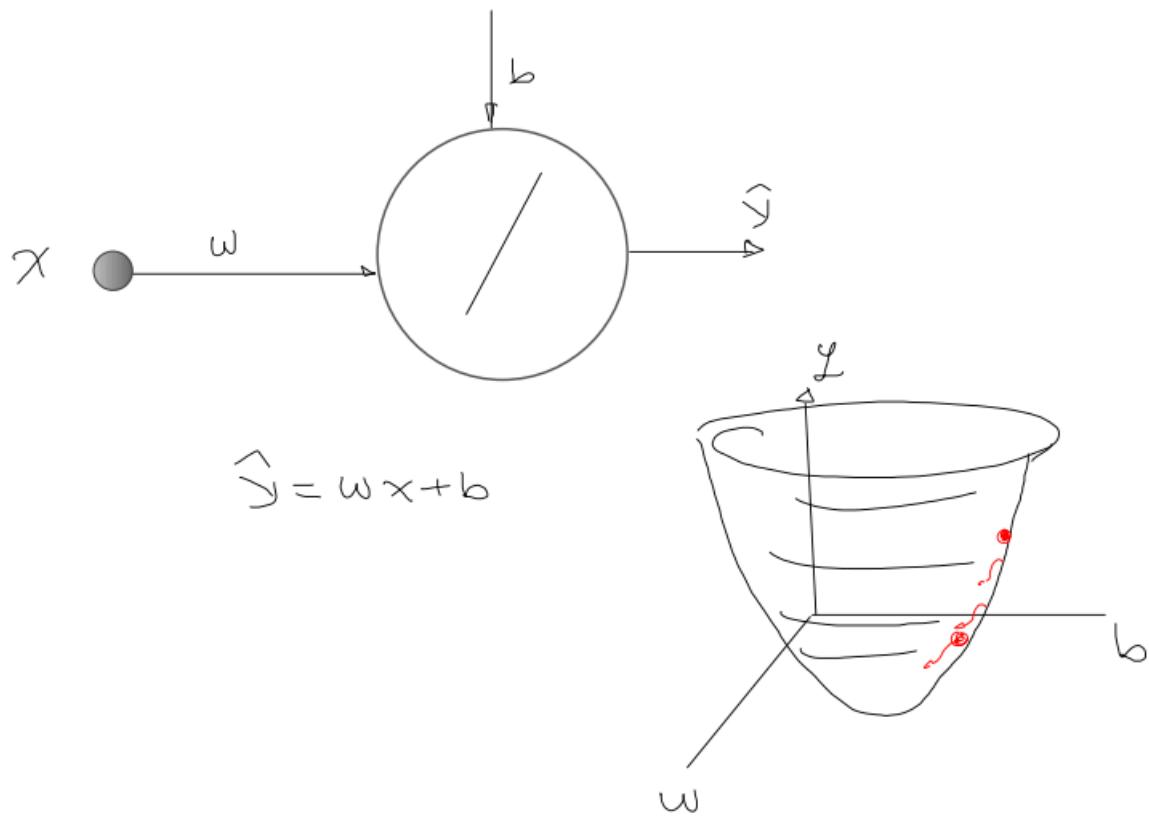
$$\begin{aligned}\omega_{t+1} &= \omega_t + \Delta\omega(t) \\ \omega_{t+1} &= \omega_t + \alpha \left(-\frac{\partial \mathcal{L}}{\partial \omega} \right) \\ &\downarrow \\ \text{LR} & \quad \text{Learning Rate}\end{aligned}$$



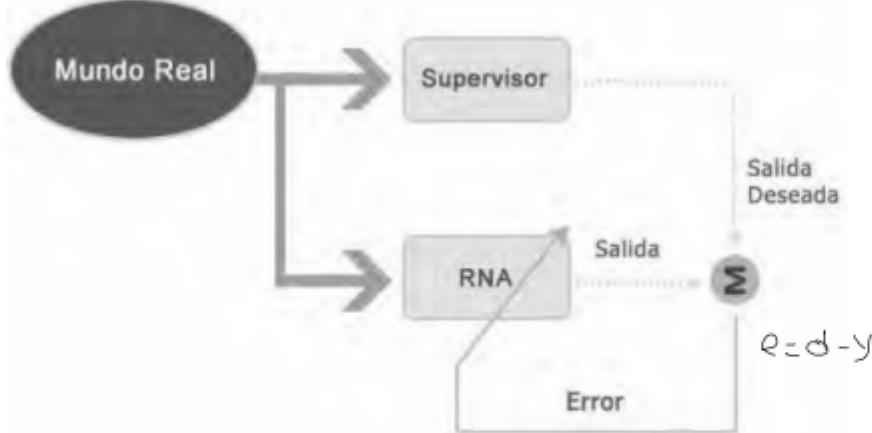
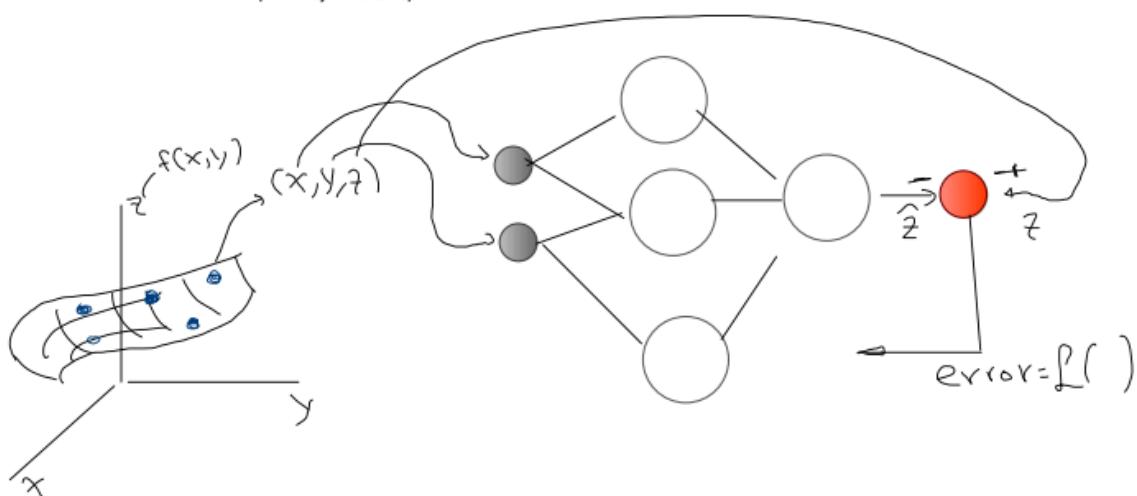
$$\begin{aligned}\text{Loss} &= f(e) \approx \text{MSE} \\ &= \frac{1}{PM} \sum_{p=1}^P \sum_{k=1}^M (d_k - y_k)^2\end{aligned}$$

x_1	x_2	$x_1 \text{ and } x_2$
0	0	0
0	1	1
1	0	1
1	1	0



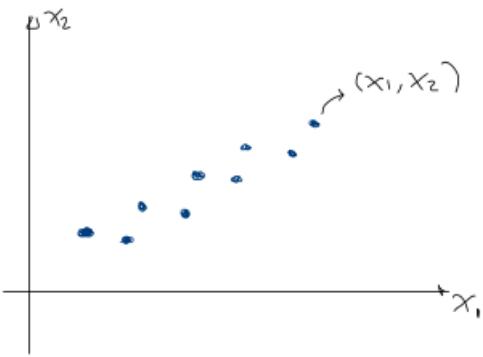


Aprendizaje de una superficie

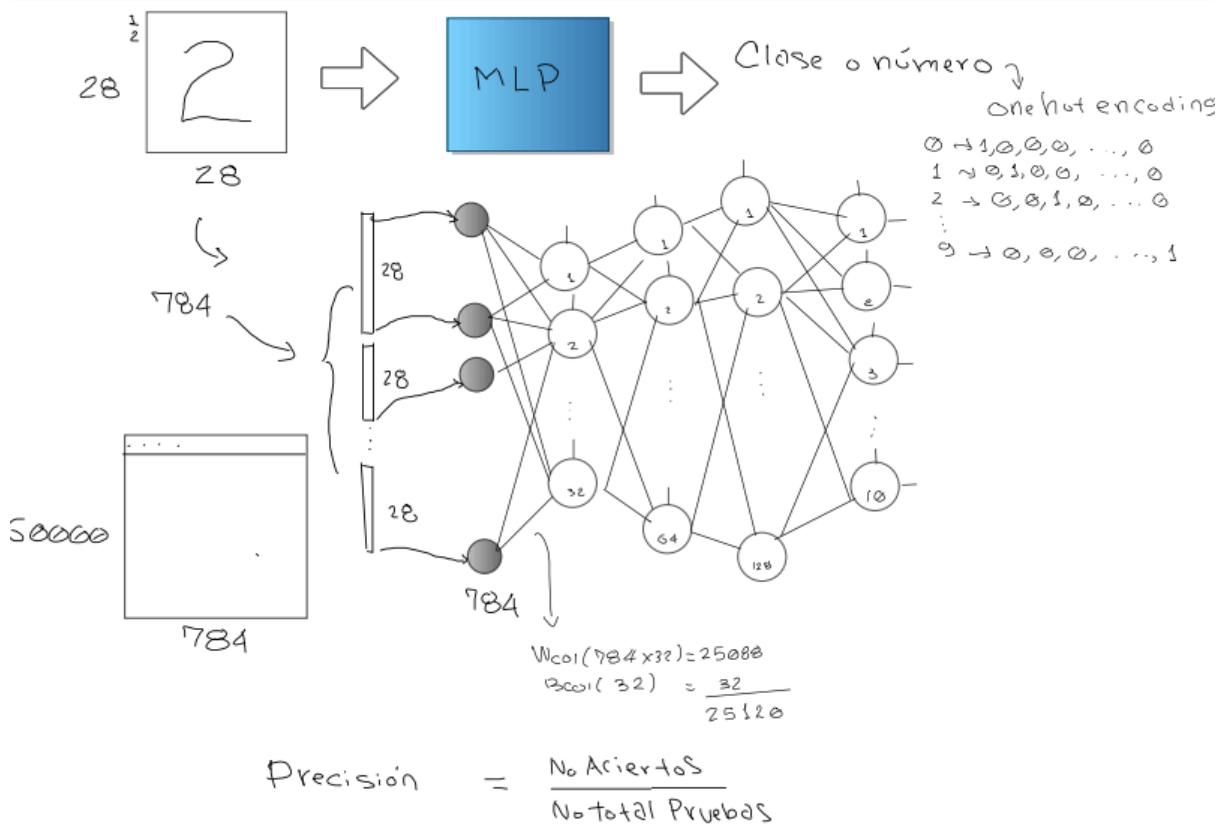
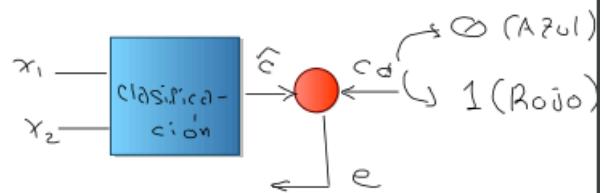
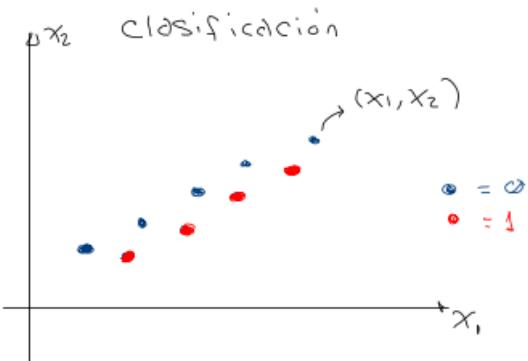
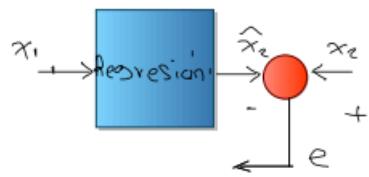


$$\begin{aligned} w(t+1) &= w(t) + \Delta w(t) \Rightarrow w(t+1) = w(t) + f(e) \\ w(t+1) &= w(t) + \alpha \left(-\frac{\partial L}{\partial w} \right) \end{aligned}$$



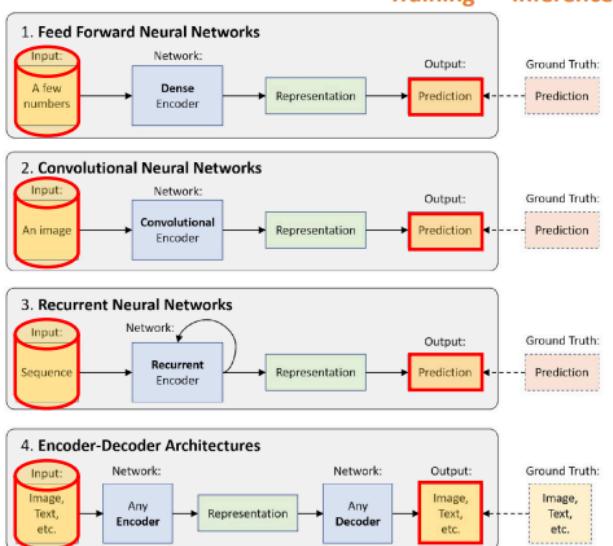


Regresión
 $x_2 = f(x_1)$



Arquitectura del deep learning

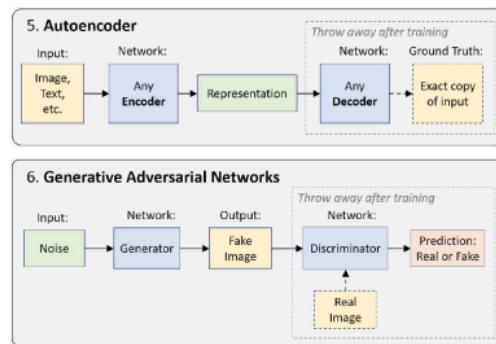
Supervised Learning



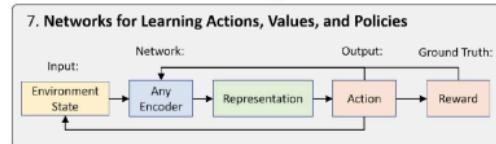
Training

Inference

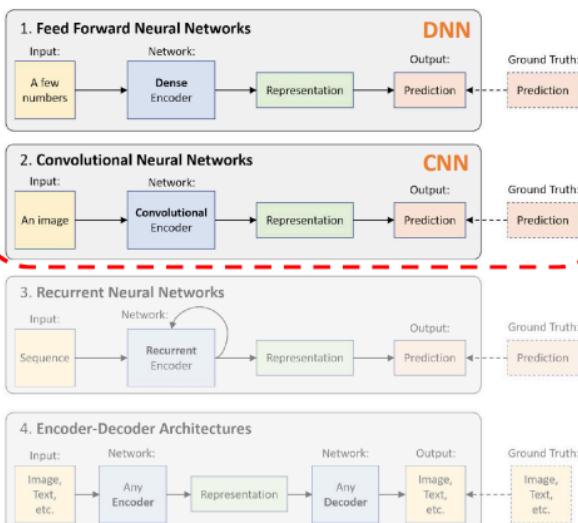
Unsupervised Learning



Reinforcement Learning



Supervised Learning



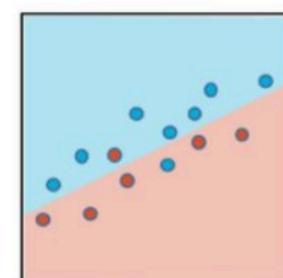
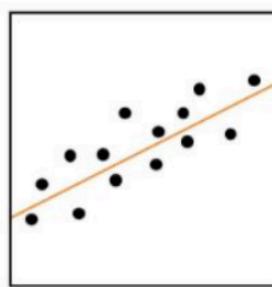
DNN

CNN

Aprendizaje Supervisado

Regresión

Clasificación



Regresión Clasificación

Regresión



Regresión

¿Qué temperatura hará mañana?

PREDICTION

84°



Clasificación

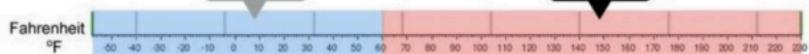


Clasificación

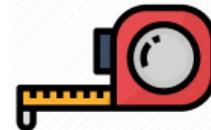
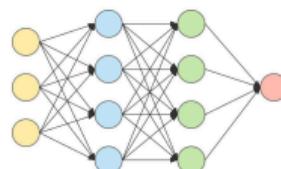
¿Mañana será un día caluroso o frío?

COLD

HOT



Flujo de trabajo



Recolección de Datos

Preprocesamiento de los Datos

Diseño de un Modelo

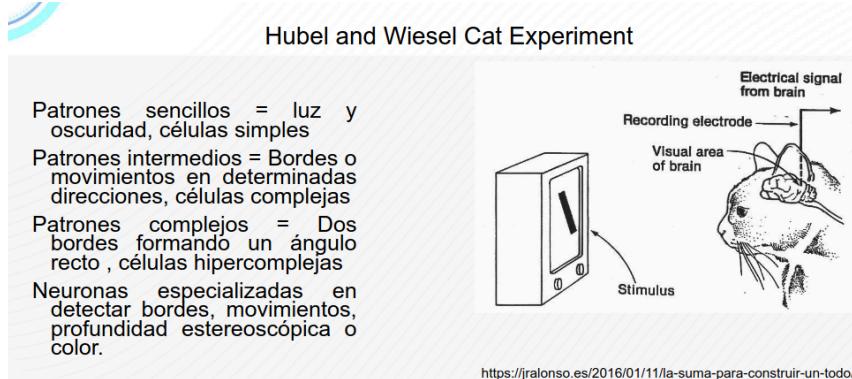
Entrenamiento del Modelo

Evaluación del Modelo

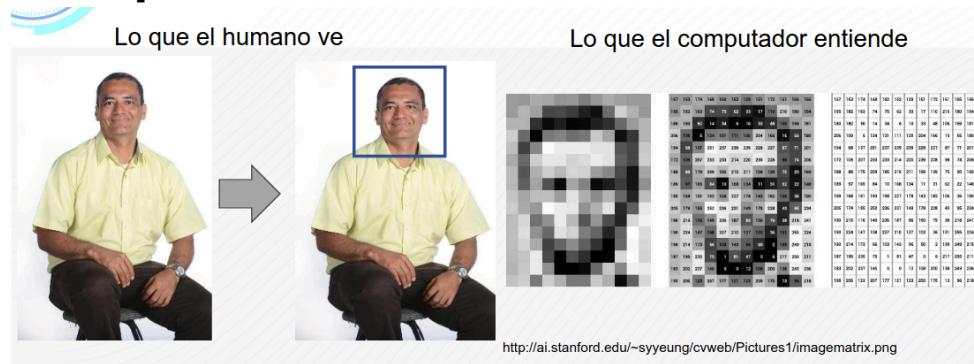
Realizar Inferencias



Inspiración Biológica de una CNN



¿Cómo “ve” un computador?

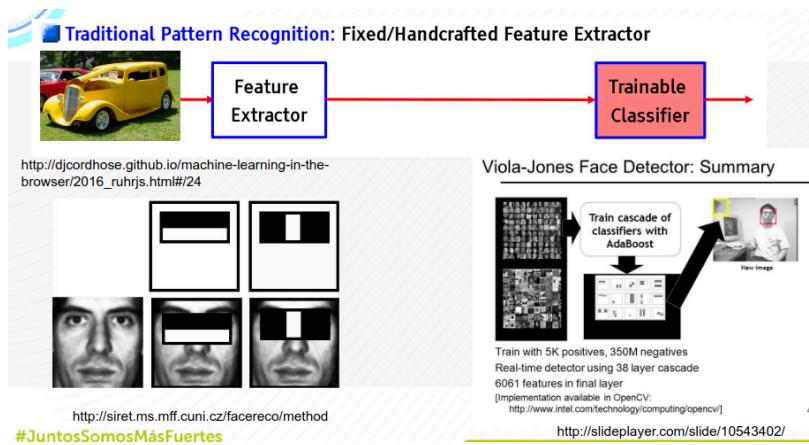


Fuente: Deep Learning. Teoria y Aplicaciones.. Jesus Alfonso Lopez. 2021

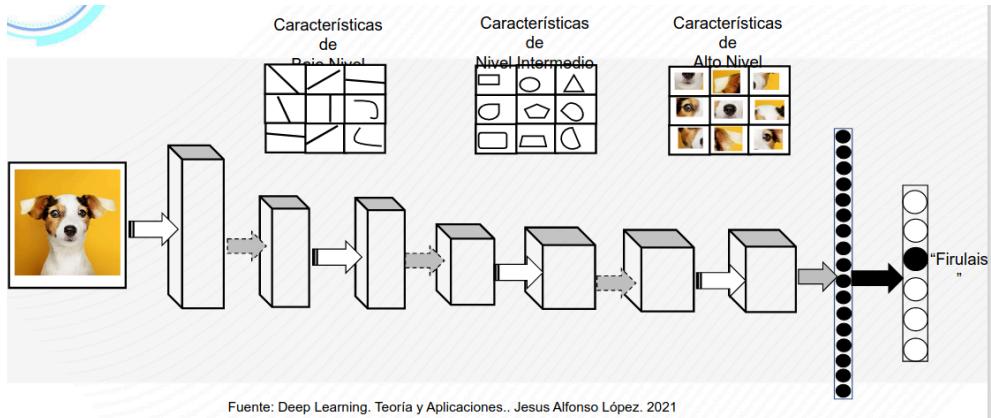
Filtros o máscaras como detectores de características



Enfoque tradicional

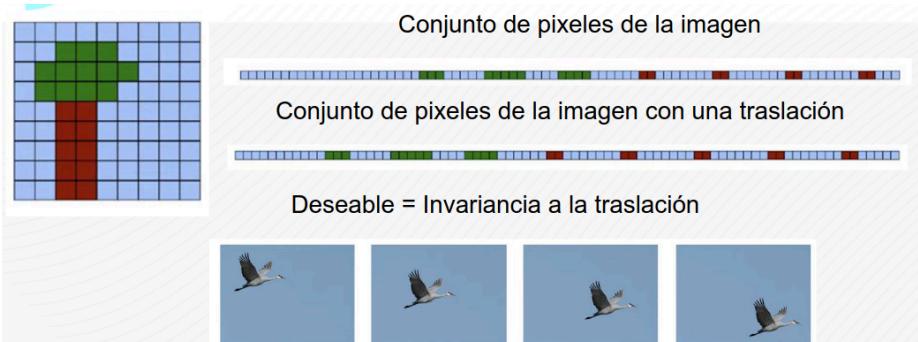


Deep learning



Fuente: Deep Learning. Teoría y Aplicaciones.. Jesus Alfonso López. 2021

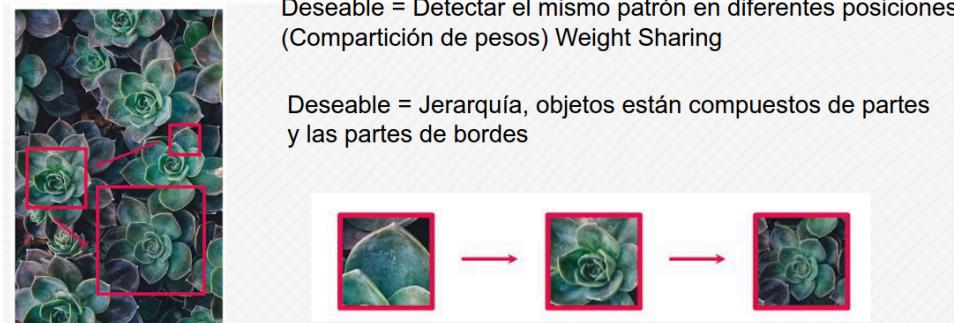
¿Por qué usar CNN?



<https://www.youtube.com/watch?v=shVKhOmT0HE&list=PLqYmG7hTraZCDxZ44o4p3N5Anz3lRVZF&index=3>

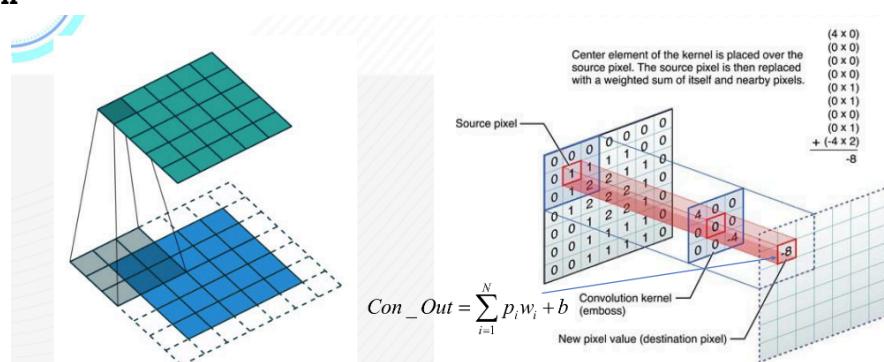
Deseable = Detectar el mismo patrón en diferentes posiciones (Compartición de pesos) Weight Sharing

Deseable = Jerarquía, objetos están compuestos de partes y las partes de bordes



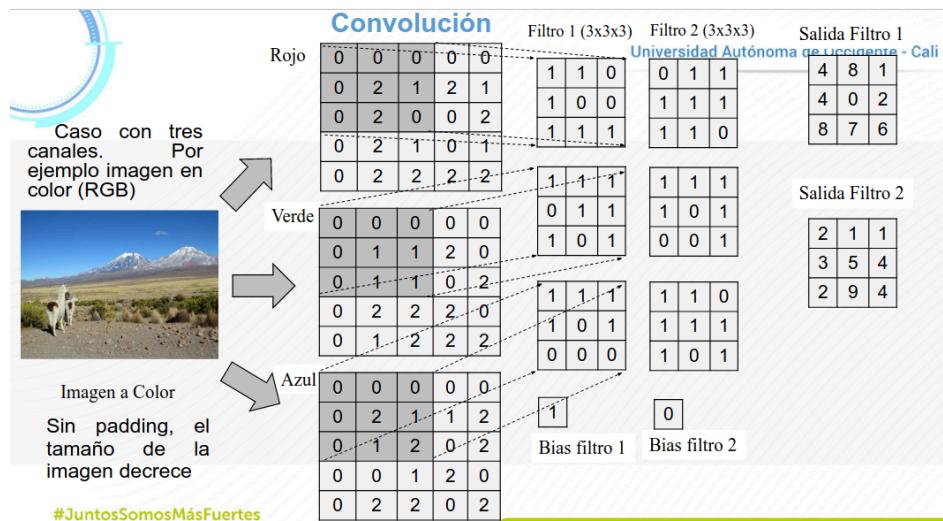
<https://www.youtube.com/watch?v=shVKhOmT0HE&list=PLqYmG7hTraZCDxZ44o4p3N5Anz3lRVZF&index=3>

Convolución

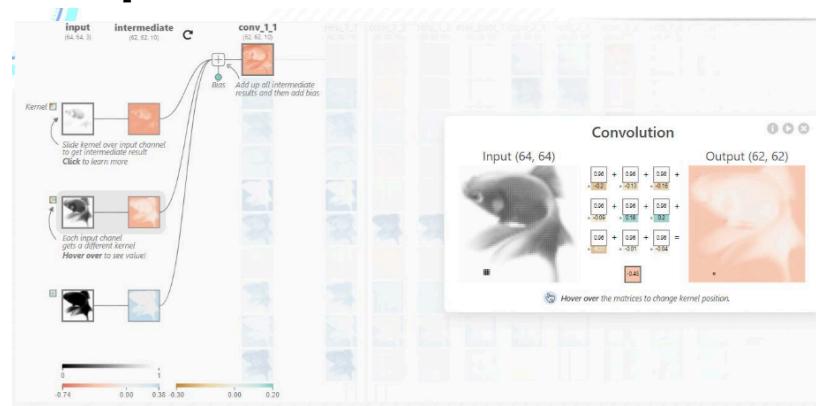


http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

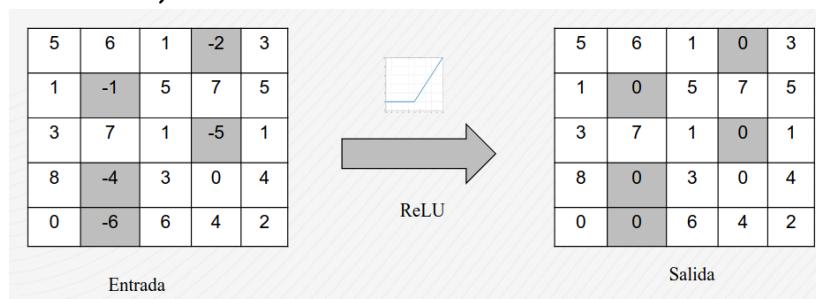
<https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html>



Convolución en CNN-Explainer

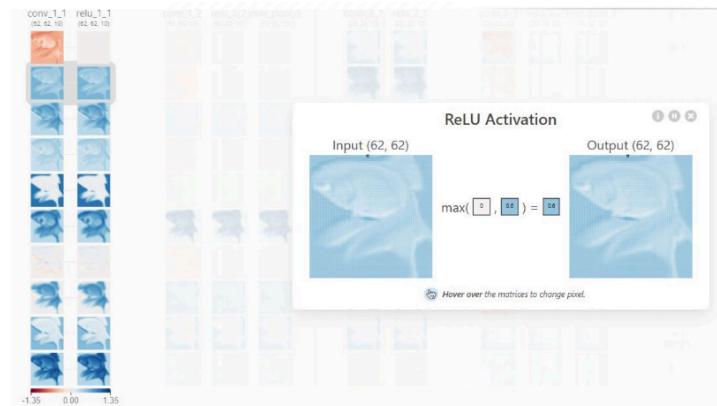


ReLU (Rectified Linear Unit)



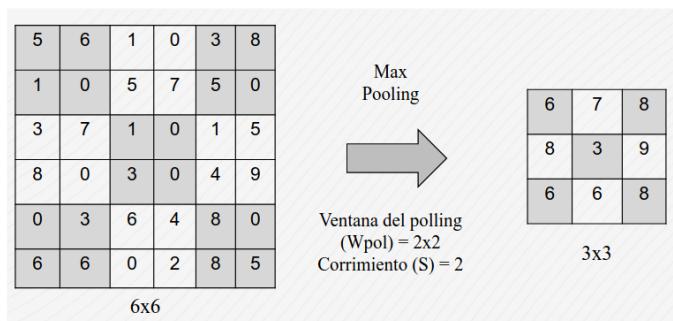
Fuente: Deep Learning. Teoría y Aplicaciones.. Jesus Alfonso López. 2021

ReLU en CNN-Explainer

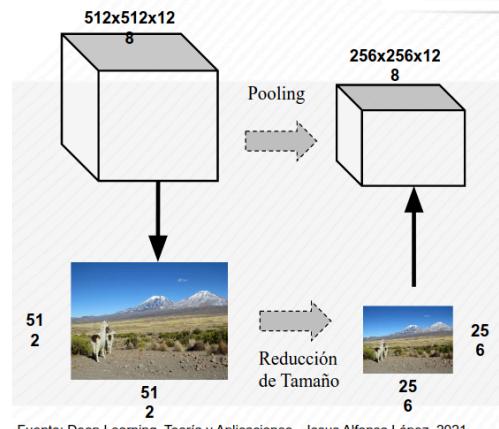


<https://poloclub.github.io/cnn-explainer/>

Pooling

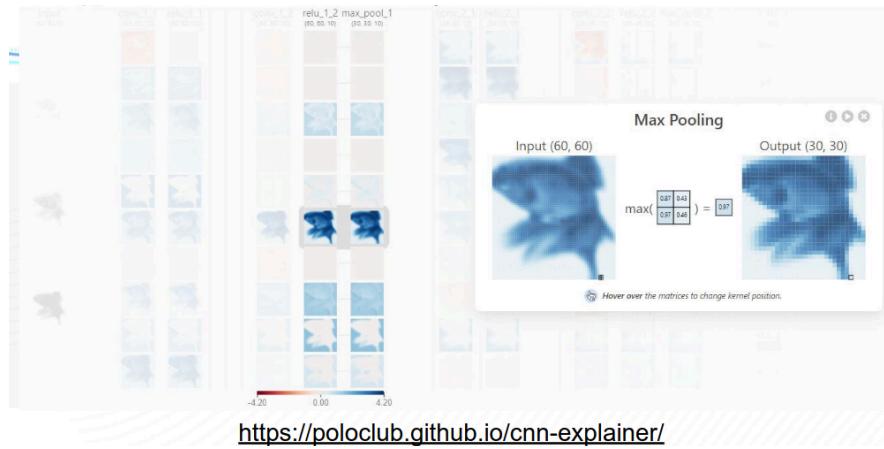


Fuente: Deep Learning. Teoría y Aplicaciones.. Jesus Alfonso López. 2021



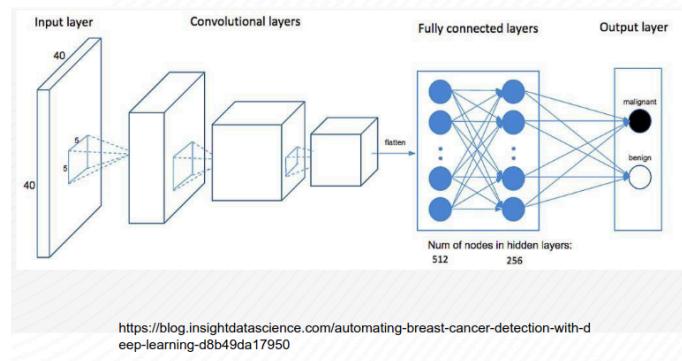
Fuente: Deep Learning. Teoría y Aplicaciones.. Jesus Alfonso López. 2021

Pooling en CNN-Explainer

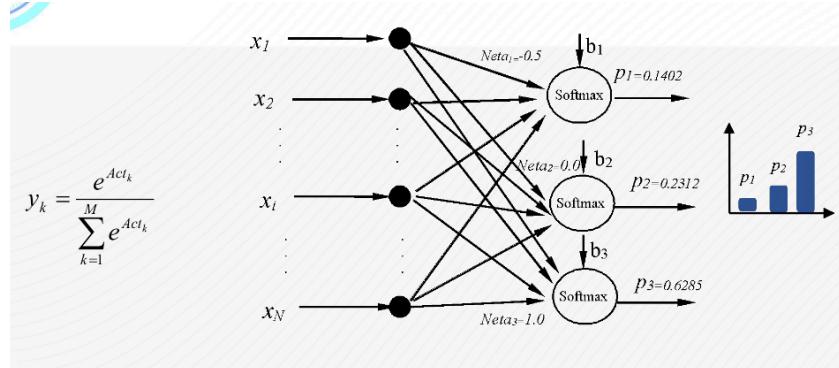


<https://poloclub.github.io/cnn-explainer/>

Capas completamente conectadas

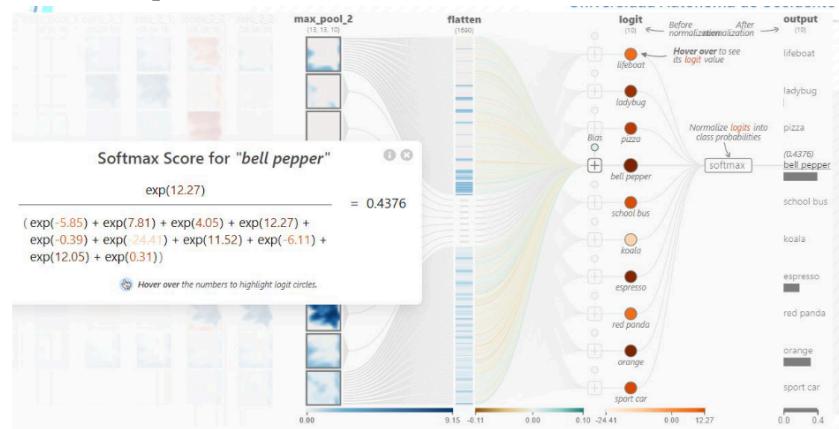


Capa Clasificadora. Softmax



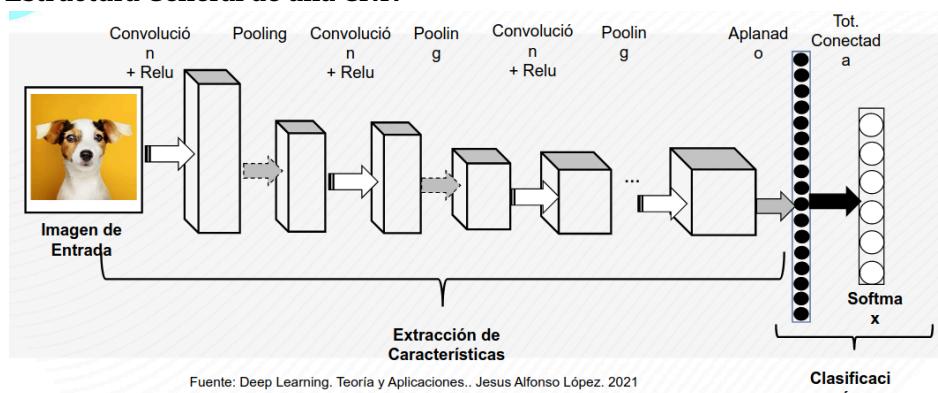
Fuente: Deep Learning. Teoría y Aplicaciones.. Jesus Alfonso López. 2021

Capa softmax en CNN-Explainer

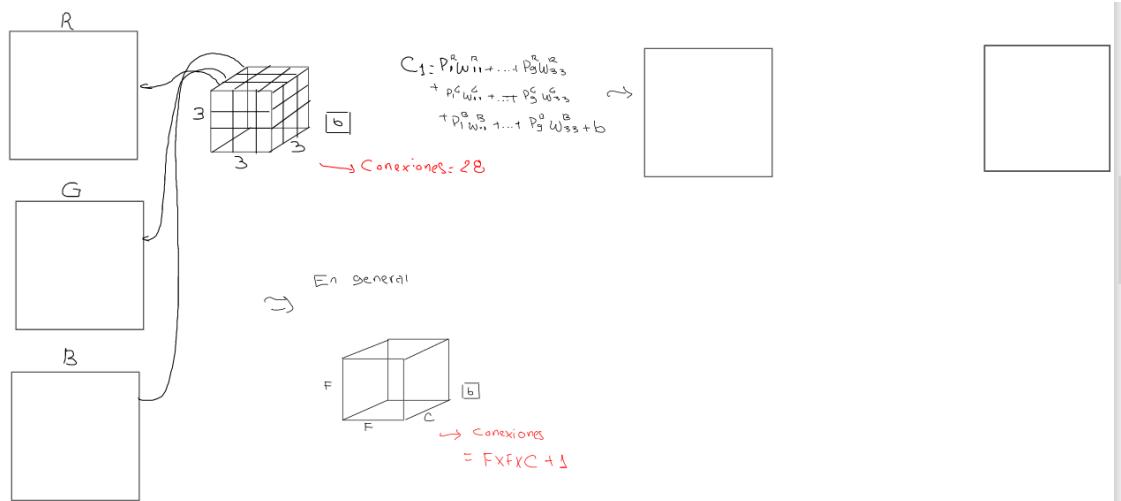
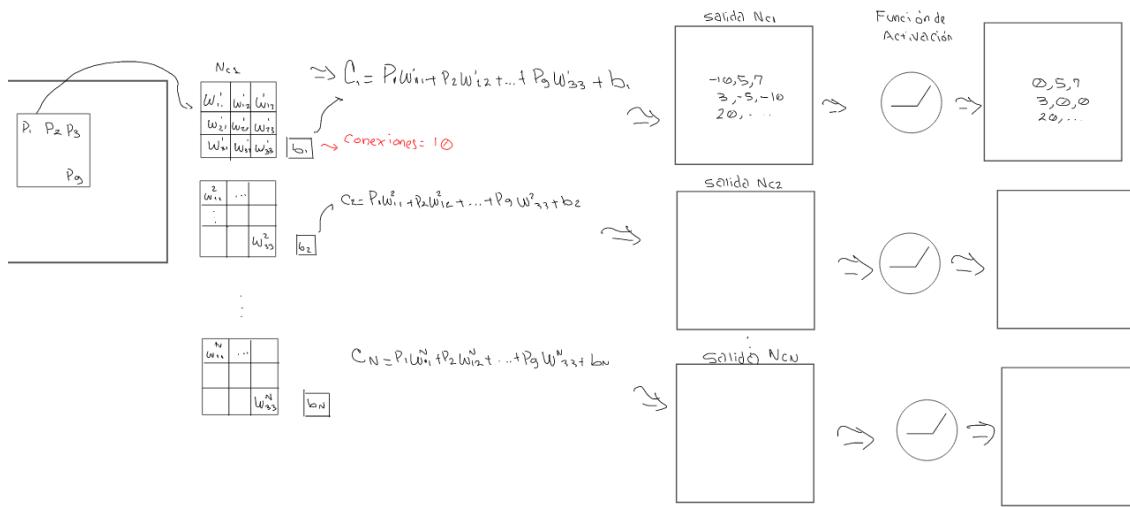


untosSomosMásFuerza <https://poloclub.github.io/cnn-explainer/>

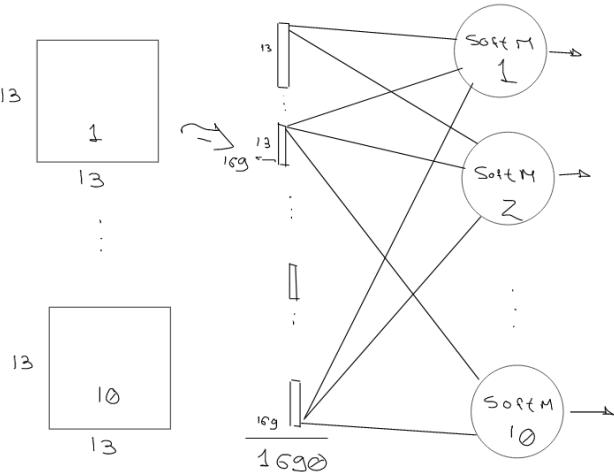
Estructura General de una CNN



Fuente: Deep Learning. Teoría y Aplicaciones.. Jesus Alfonso López. 2021



Flatten

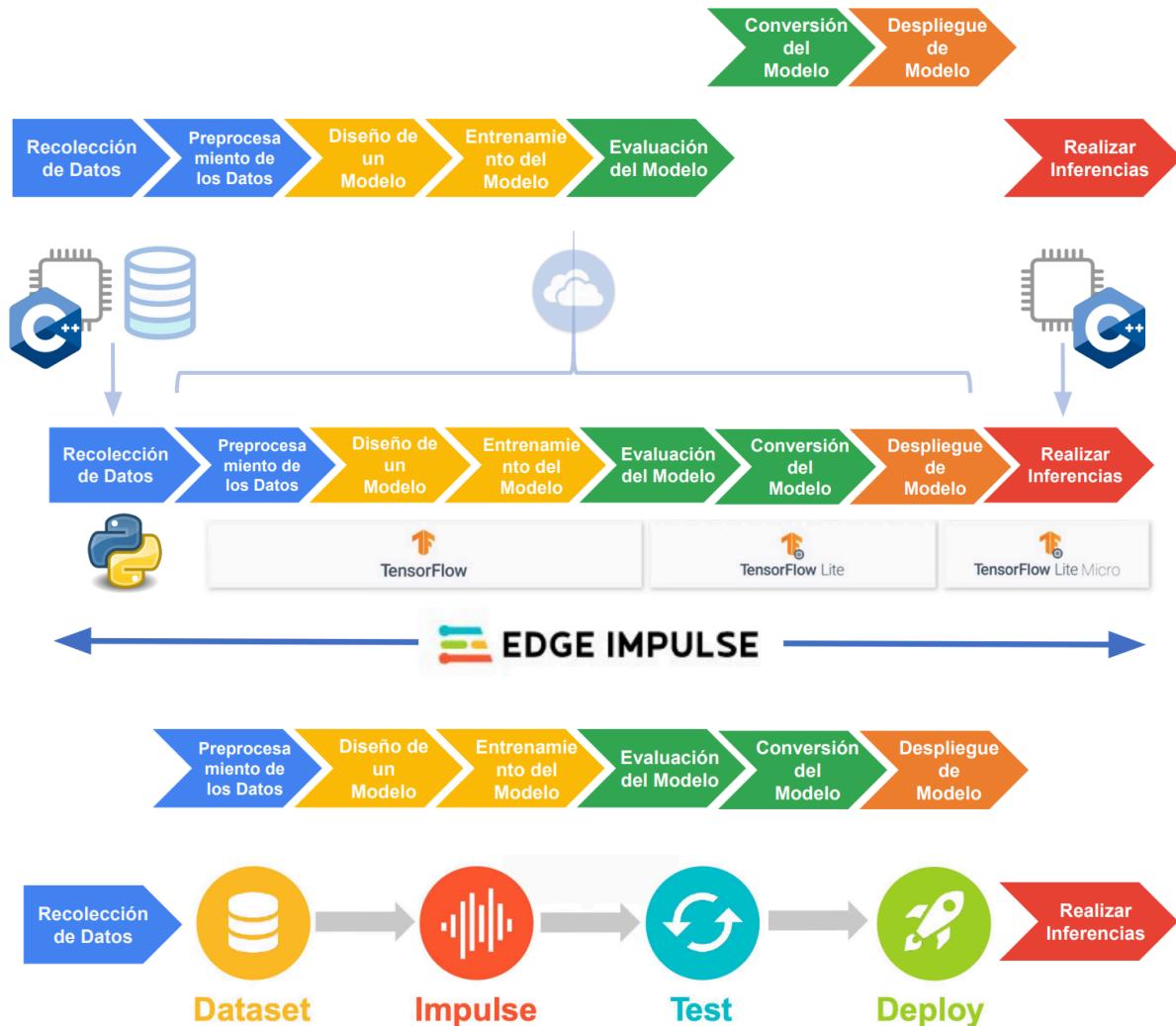


Introducción al Edge Impulse

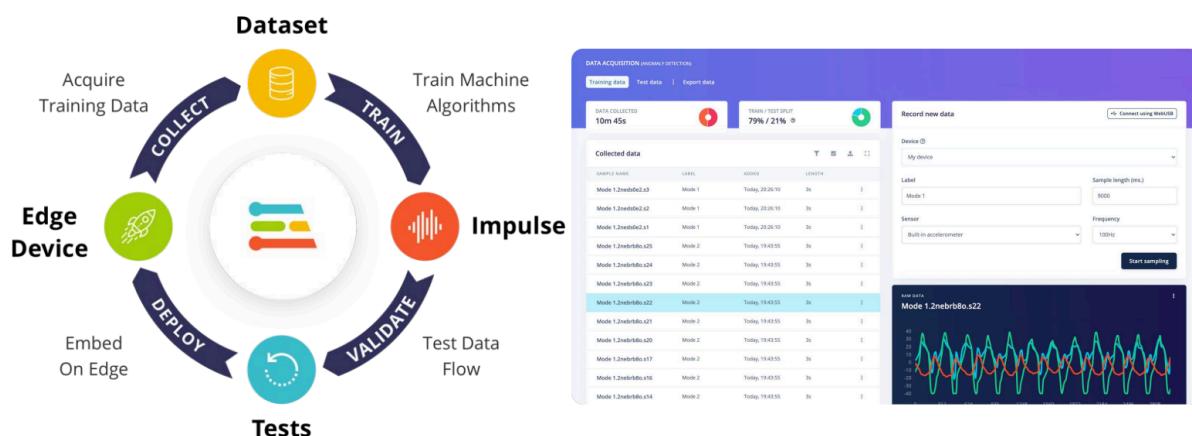
TinyML - Revisión Flujo de Trabajo

Deep (Machine) Learning Flujo de Trabajo

Tiny ML Learning Flujo de Trabajo (¿Qué implica?)

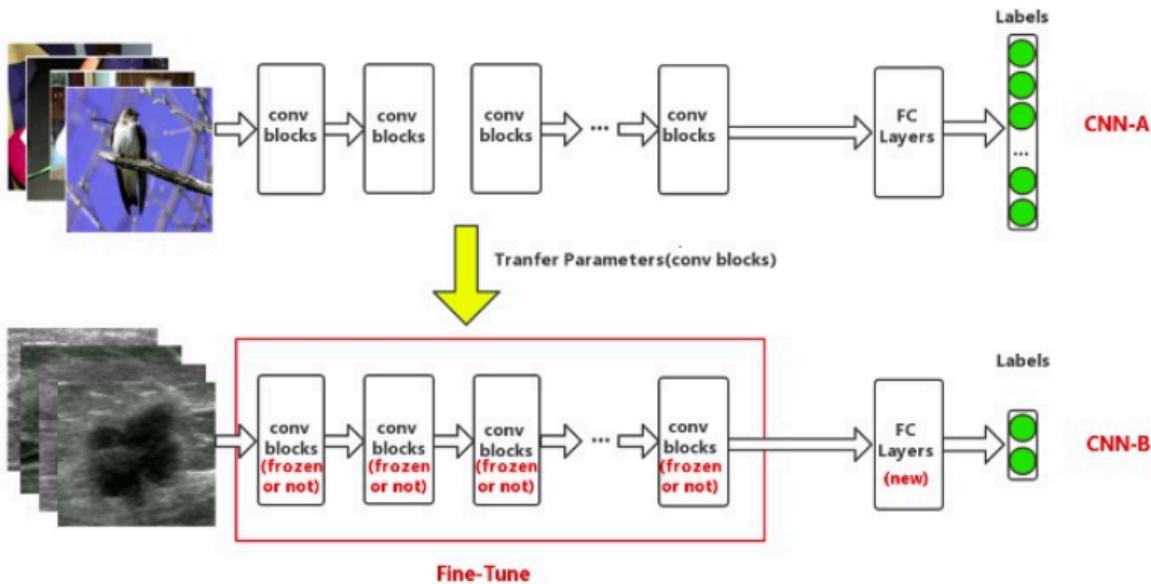


Edge Impulse Studio- Platforma para TinyML



<https://www.edgeimpulse.com/>

Transfer Learning



Introducción al TensorFlow Lite



arm

TensorFlow Lite es un framework multiplataforma listo para producción en el despliegue de modelos de IA en dispositivos móviles y sistemas embebidos.

¿Qué puede hacer Tensorflow Lite?

- Optimizar sus modelos.
- Disponer de diferentes herramientas para integración fácil en dispositivos móviles, embebidos y aplicaciones basadas en microcontroladores.
- Tomar ventajas de aceleradores especiales en hardware como Edge TPU y GPU.
- En lugar de utilizar el backend completo de TensorFlow, TensorFlow Lite utiliza una versión reducida y optimizada de la biblioteca de cálculo de TensorFlow.
- Además, el modelo TFLite está diseñado para ser más eficiente en términos de tamaño y velocidad de ejecución que el modelo original, lo que lo hace más adecuado para dispositivos con recursos limitados.

Optimización del Modelo - ¿Por qué es necesaria?

- Reducción del tamaño.
- Reducción de latencia.
- Compatibilidad con aceleradores.

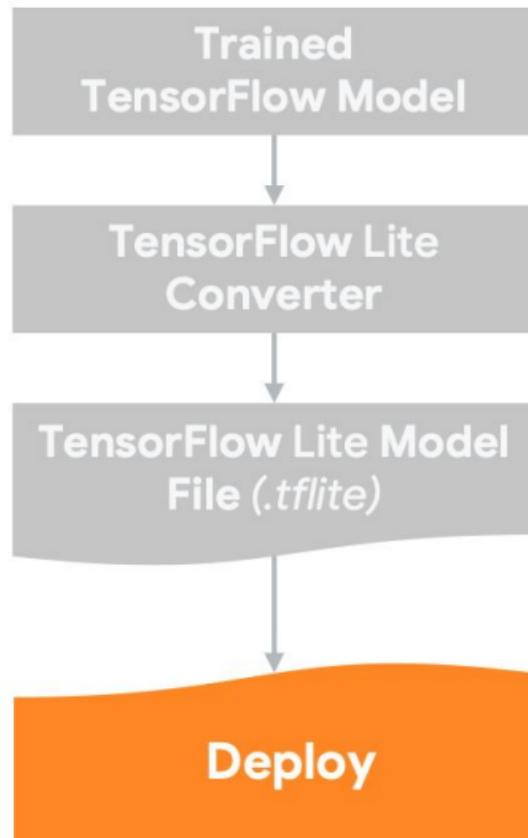
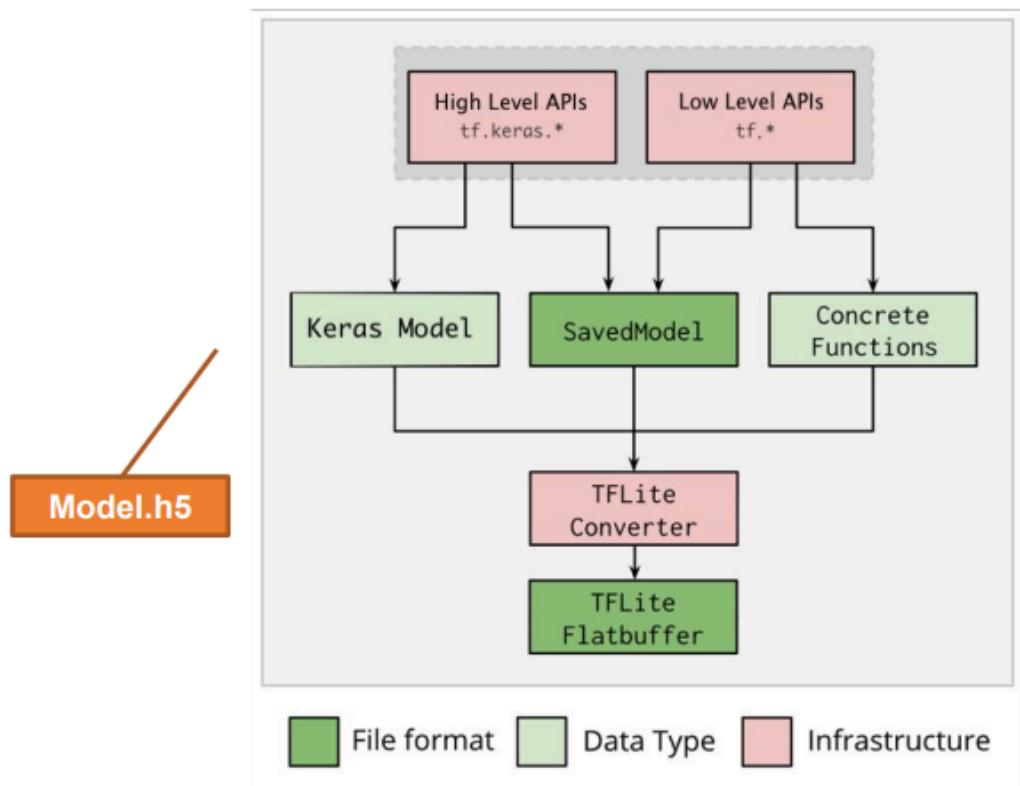


Diferencias TensorFlow y TensorFlow Lite

TensorFlow	TensorFlow Lite
Topología	TensorFlow
Pesos	Variable
Tamaño Binario	Variables
Cómputo Distribuido	No importa
Background del Desarrollador	Necesario
	TensorFlow Lite
	Fijos
	Fijos
	Alta Prioridad
	No necesario
	Desarrollador de Aplicaciones

Flujo de Trabajo para la Conversión

El diagrama siguiente ilustra el flujo de trabajo de alto nivel para la conversión de un modelo.



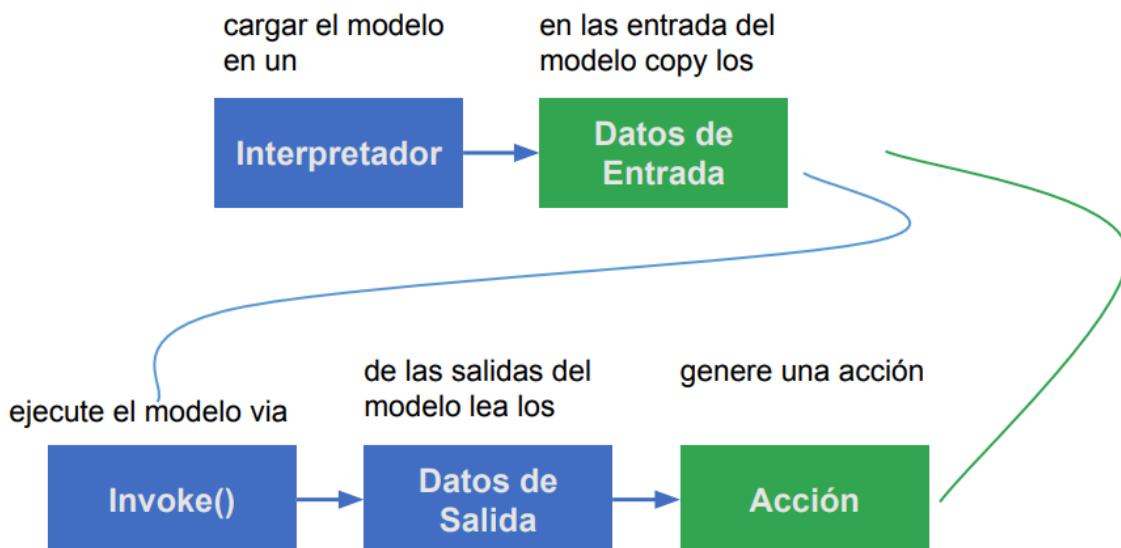
¿Que se hace en la conversión del modelo?

- **Utilizar capas y operaciones soportadas por TensorFlow Lite:** TensorFlow Lite tiene un conjunto limitado de operaciones y capas que admite en comparación con TensorFlow completo.
- **Eliminar capas y operaciones innecesarias:** Si el modelo original utiliza capas o operaciones que no son necesarias para la tarea en cuestión, se pueden eliminar. Por ejemplo, si el modelo original tiene capas de regularización que no son necesarias para la tarea, se pueden eliminar para simplificar el modelo.
- **Fusión de operaciones:** El TFLiteConverter puede fusionar múltiples operaciones en una sola operación. Por ejemplo, se pueden fusionar varias capas de convolución en una sola capa para mejorar la velocidad de ejecución del modelo.
- **Eliminación de operaciones innecesarias:** El TFLiteConverter puede eliminar operaciones que no son necesarias para la tarea en cuestión para reducir el tamaño del modelo y mejorar su velocidad de ejecución.
- **Optimización de memoria:** El TFLiteConverter puede optimizar el uso de la memoria del modelo para reducir su tamaño y mejorar su velocidad de ejecución

¿Cómo se convierte el modelo a TensorFlow Lite?

```
# Se entrena el modelo en Keras  
history = modelo.fit(x_train, y_train, validation_data=(x_test, y_test),  
                      epochs=10, batch_size=32, verbose=0, shuffle=True)  
  
# Se guarda el modelo en formato Keras  
modelo.save('cifar_10_model.h5')  
  
# Se convierte el modelo a TensorFlow Lite  
converter =  
tf.lite.TFLiteConverter.from_keras_model(modelo)  
modelo_tflite = converter.convert()
```

¿Cómo se usa el modelo de TensorFlow Lite?



¿Cómo se usa el modelo de TensorFlow Lite?

```

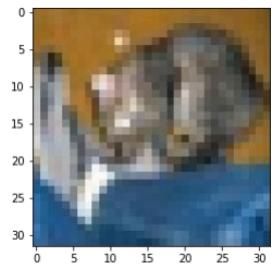
1. interpreter = tf.lite.Interpreter("/content/cifar10_quant_model.tflite")
2. interpreter.allocate_tensors()

3. set_input_tensor(interpreter, image)

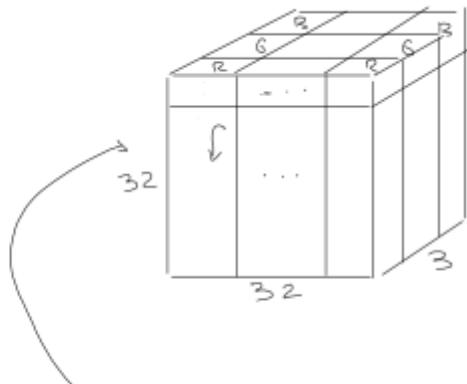
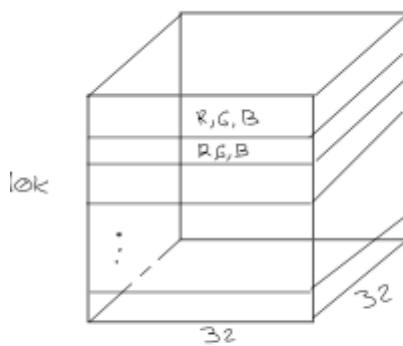
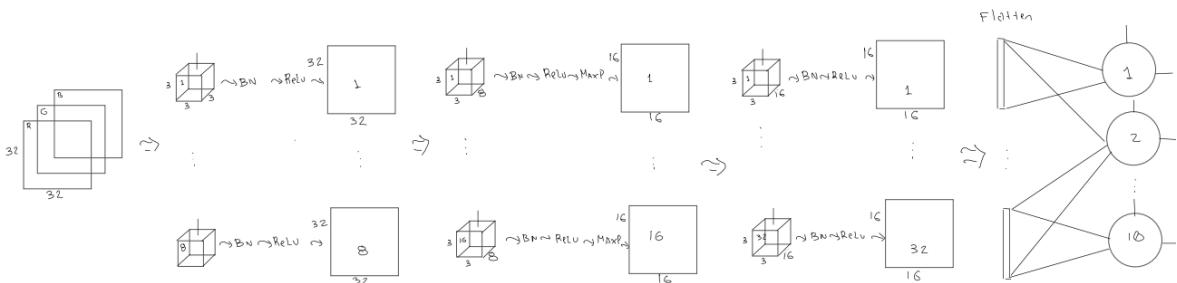
4. interpreter.invoke()

5. output_details = interpreter.get_output_details()[0]
6. img_pred = np.argmax(interpreter.get_tensor(output_details['index'][0]))

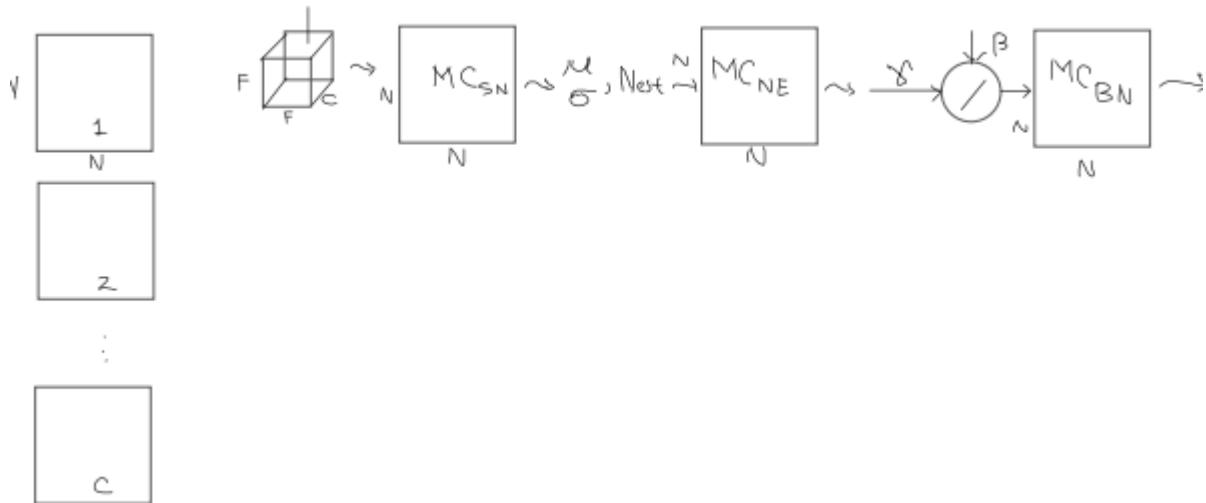
```



$3 \Rightarrow \text{"CAT"}$



$$X_{\text{test}}(10K, 32, 32, 3) \rightsquigarrow \text{Imagen}(1, 32, 32, 3) \\ \rightsquigarrow X_{\text{test}}[0] \approx (32, 32, 3)$$



Datos secuenciales

- Voz
- Sonido
- Sensores inerciales
- Series temporales

Aplicaciones con datos secuenciales

- Predicción del siguiente valor de la secuencia.



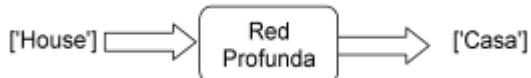
- Clasificación de la secuencia de entrada.



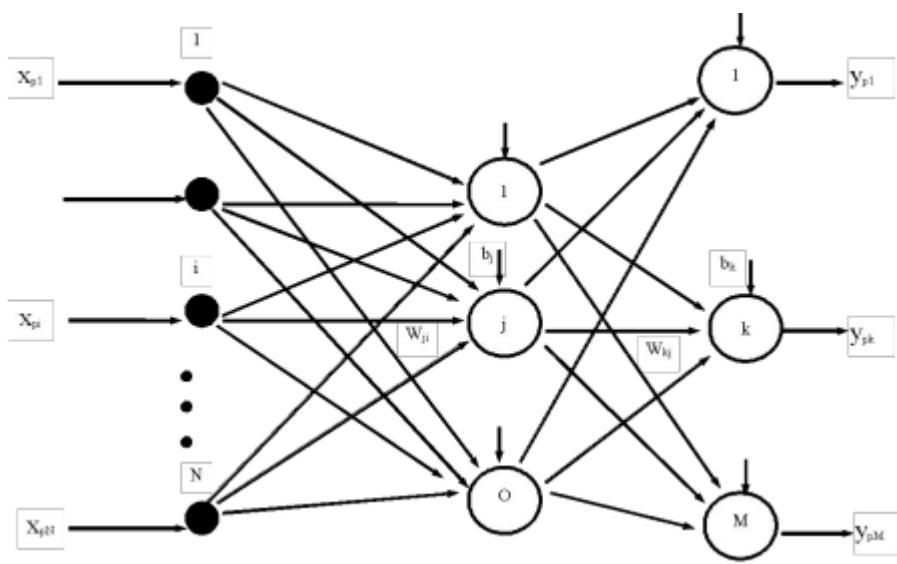
- Generación de unas secuencias en función de la entrada.



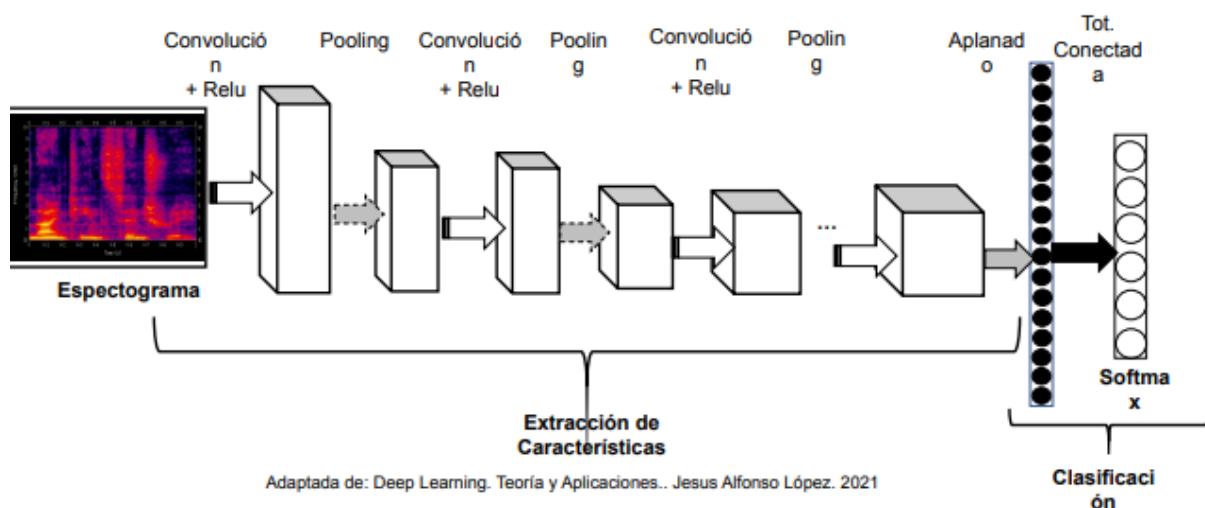
- Predicción de una nueva secuencia dependiendo de la secuencia de entrada



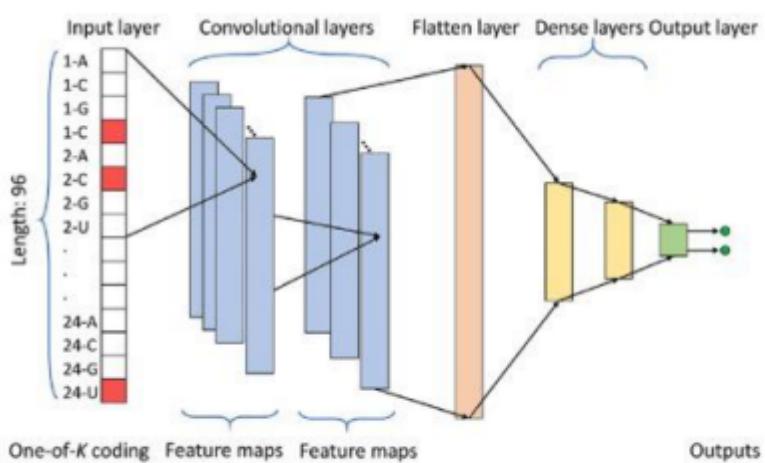
Red MPL



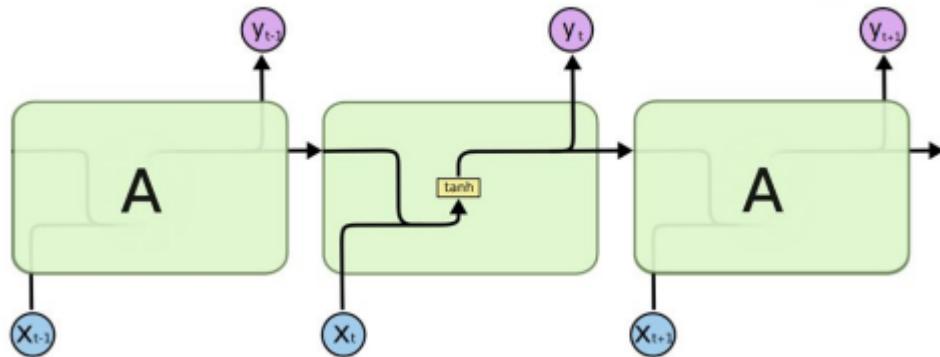
Red Convolucional 2D



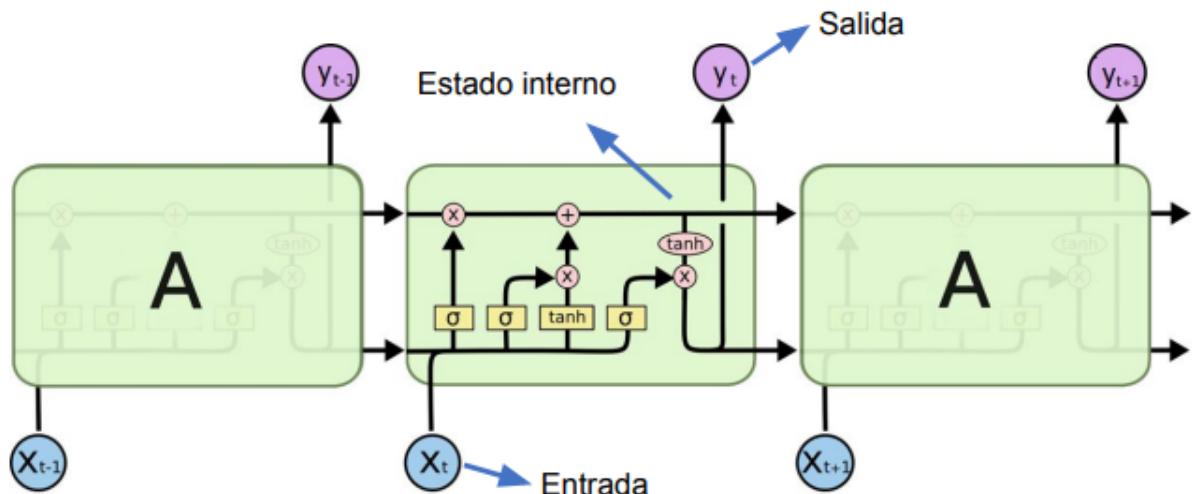
Red convolucional 1D



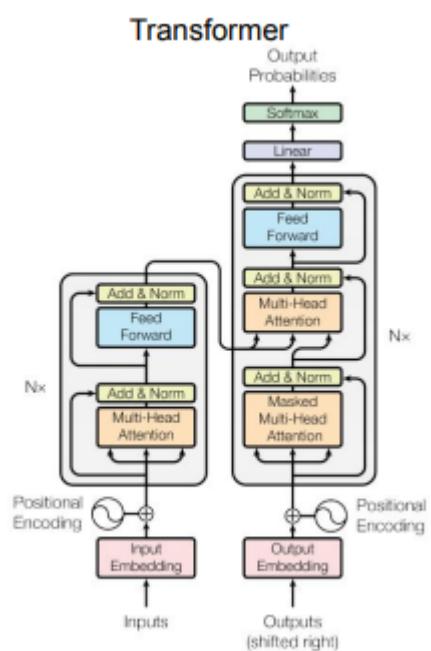
Unidad recurrente básica



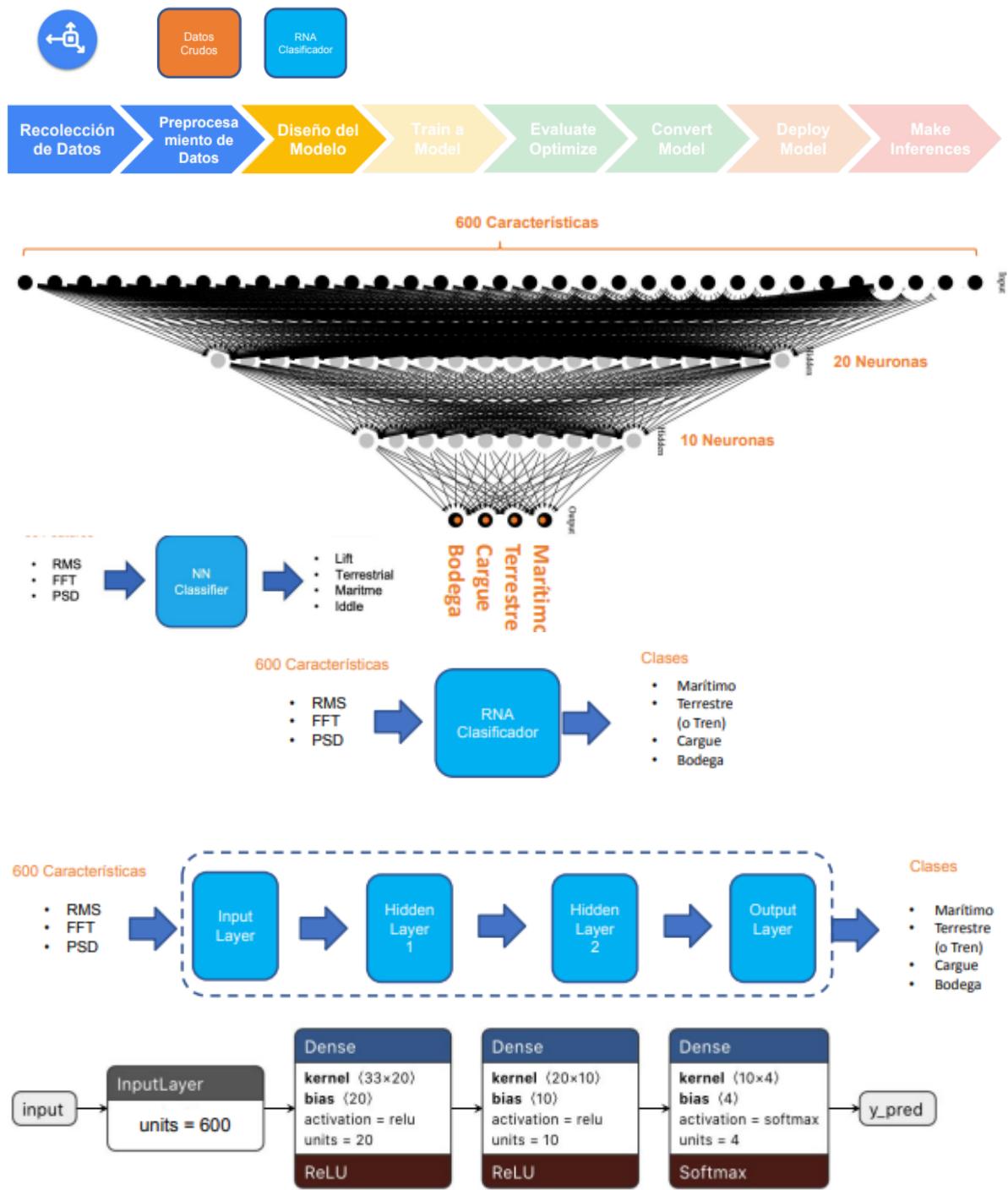
Unidad LSTM



Auto-atención



Pre-procesamiento de los datos - Diseño del Modelo (Clasificador basado en RNA)



Entrenar, Evaluar, Convertir y Desplegar el Modelo





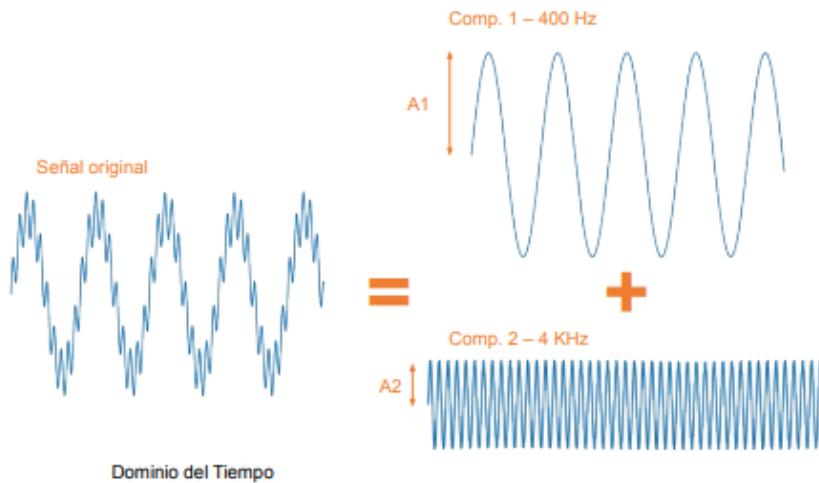
**Feature extraction:
FFT**

Extracción de Características
3 Valores RMS (Root Mean Square), uno por cada eje (x, y, z)

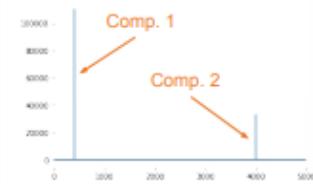
$$x_{\text{RMS}} = \sqrt{\frac{1}{n} (x_1^2 + x_2^2 + \dots + x_n^2)}.$$

\downarrow
 $n=200$

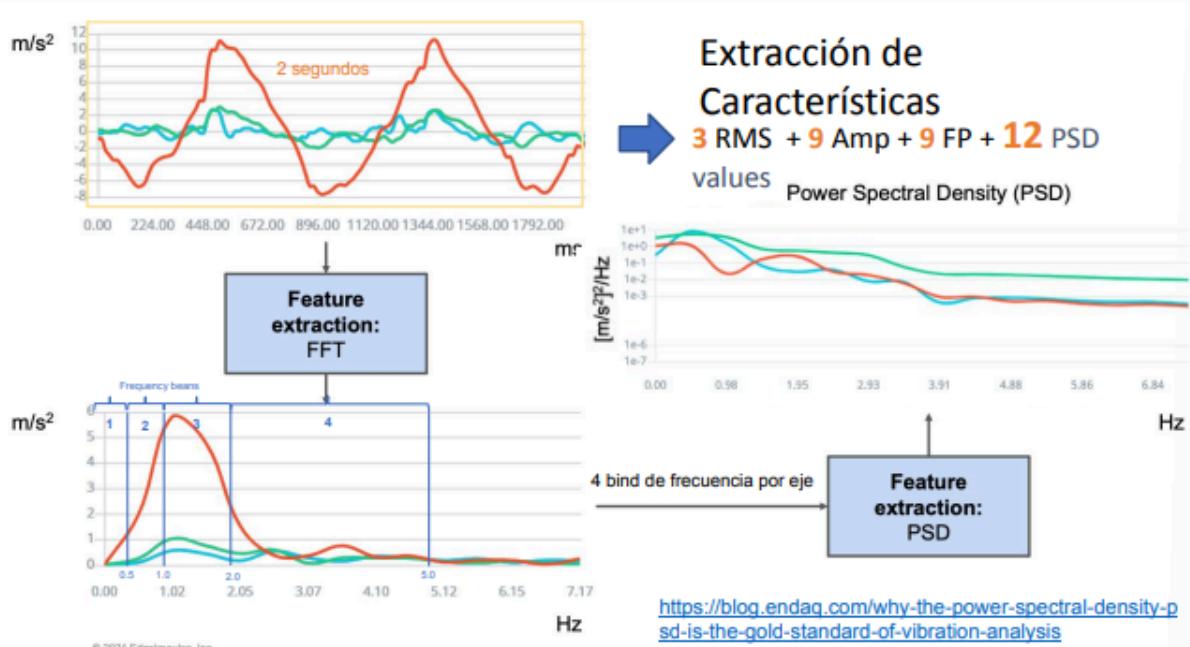
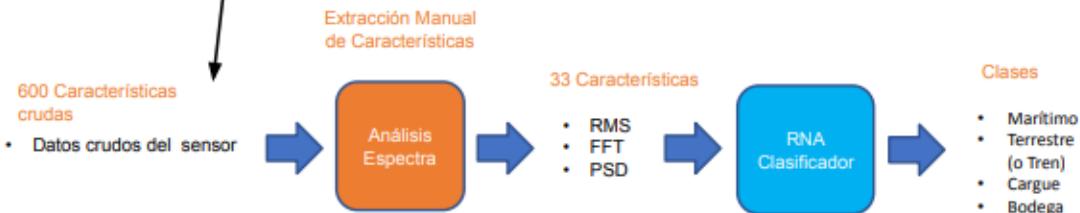
Transformada Rápida de Fourier (FFT)



```
from scipy.fft import fft
yf = fft(raw signal)
plt.plot(xf, np.abs(yf));
```



Dominio de la Frecuencia



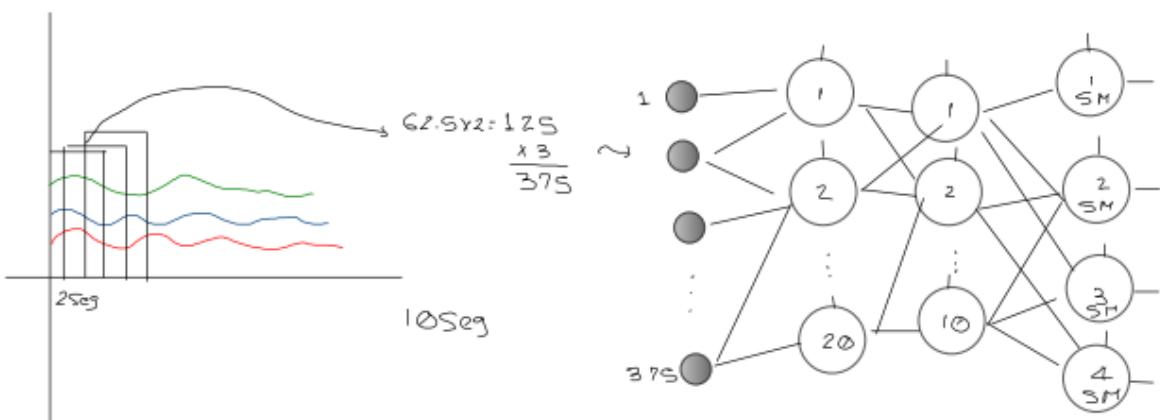
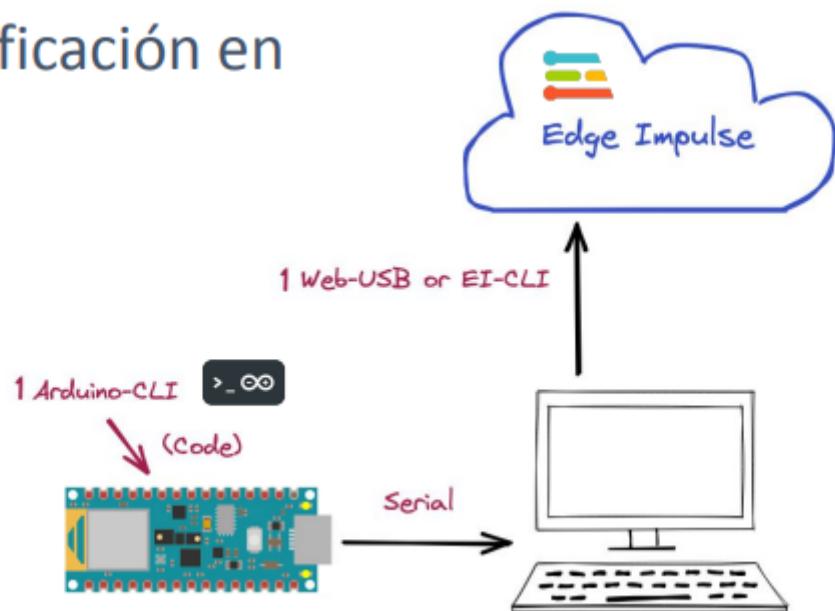
11 Características

accX RMS	accY RMS	accZ RMS
accX Peak	accY Peak	accZ Peak
accX Peak	accY Peak	accZ Peak 1 Freq
accX Peak	accY Peak	accZ Peak 1 Height
accX Peak	accY Peak	accZ Peak 2 Freq
accX Peak	accY Peak	accZ Peak 2 Height
accX Spec	accY Spec	accZ Peak 3 Freq
accX Spec	accY Spec	accZ Peak 3 Height
accX Spec	accY Spec	accZ Spectral Power 0.1 - 0.5
accX Spec	accY Spec	accZ Spectral Power 0.5 - 1.0
accX Spec	accY Spec	accZ Spectral Power 1.0 - 2.0
accX Spec	accY Spec	accZ Spectral Power 2.0 - 5.0

11 Características

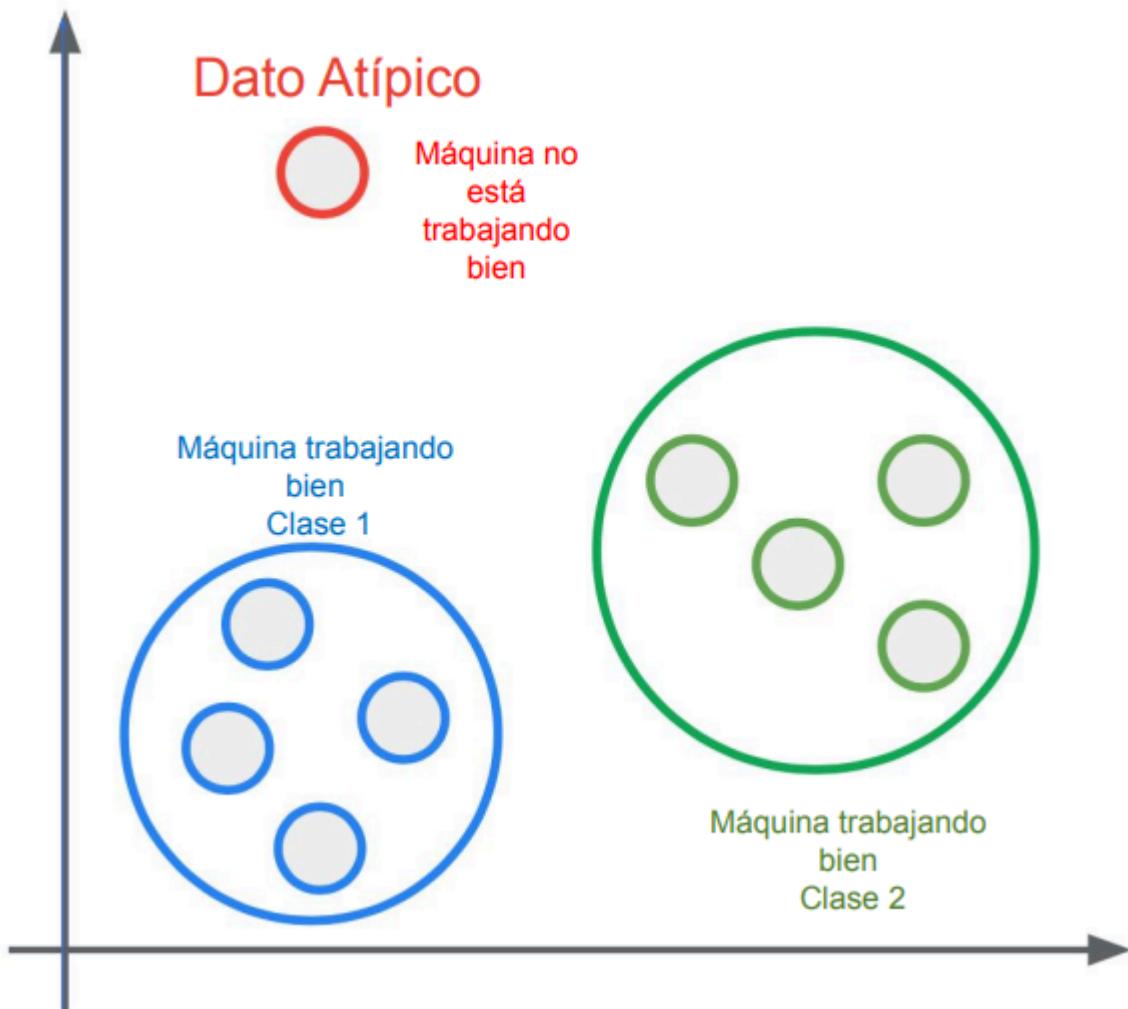
11 Características

Clasificación en Vivo

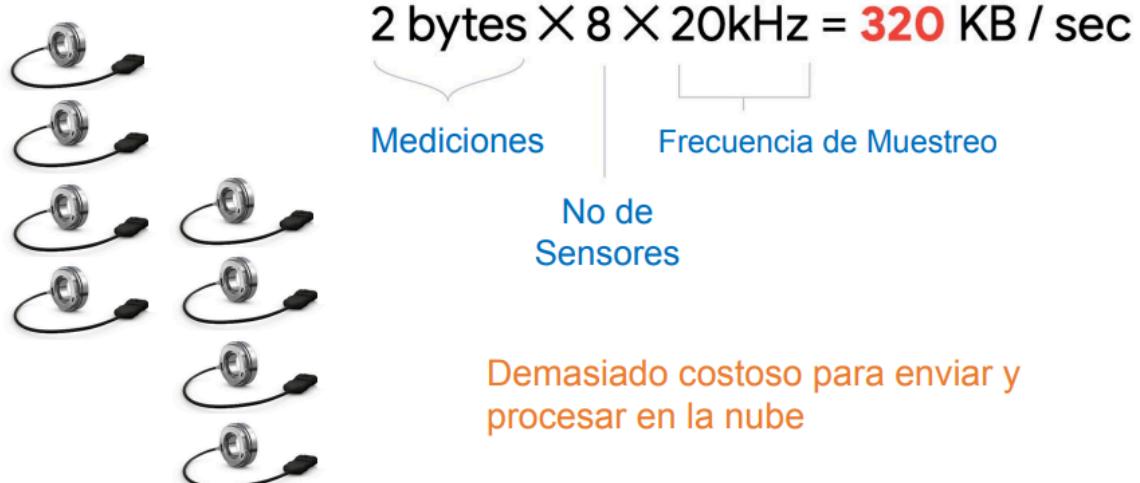


Detección de Anomalías con Edge Impulse

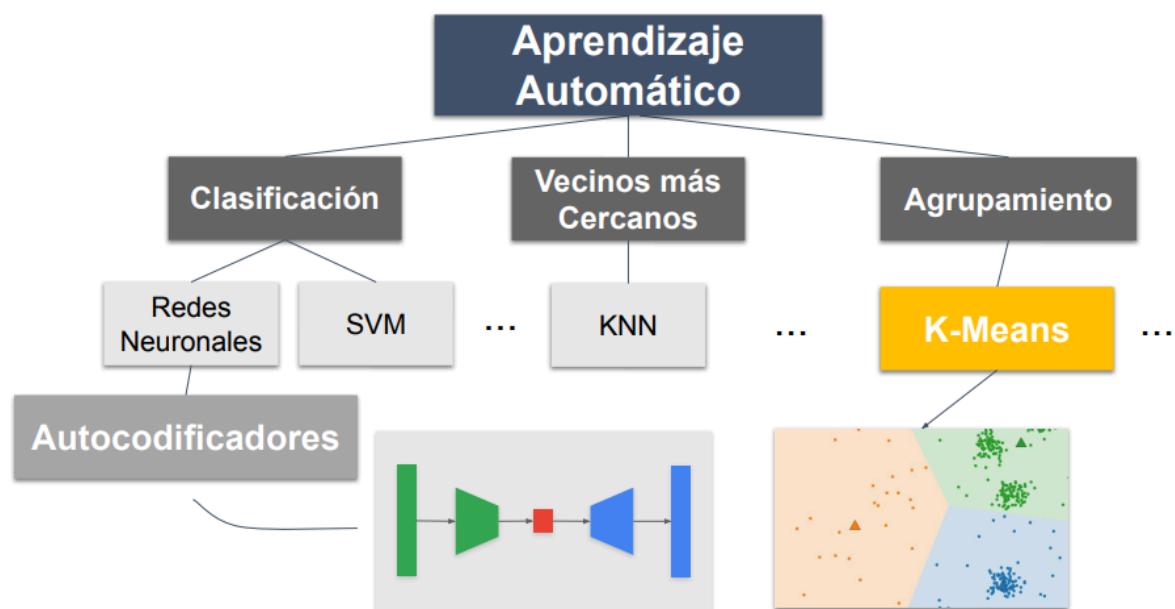
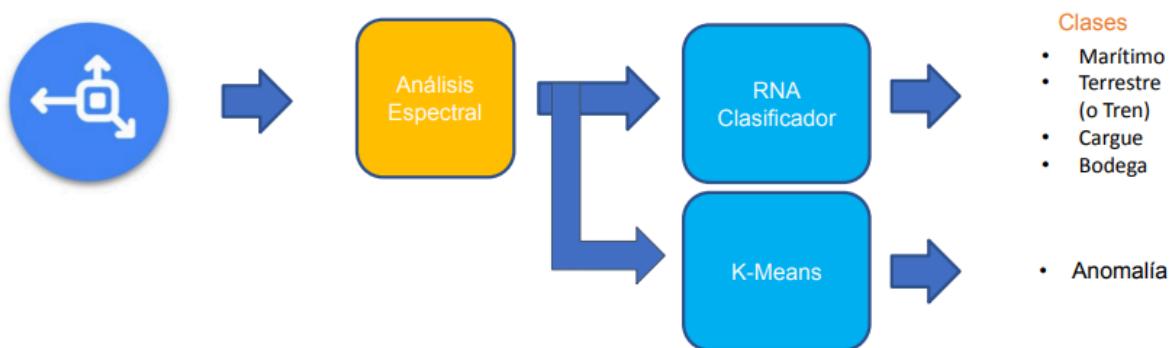
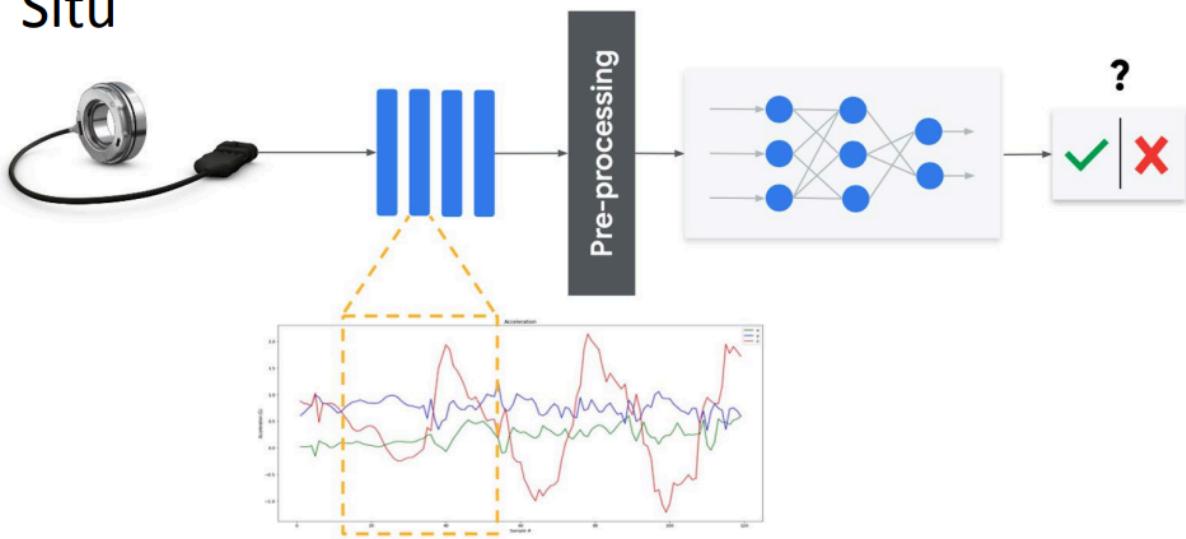
¿Qué es Detección de Anomalías? En el análisis de datos, la detección de anomalías es la identificación de elementos, eventos u observaciones "raros" o "extraños" que generan sospechas porque difieren significativamente de la mayoría de los datos.

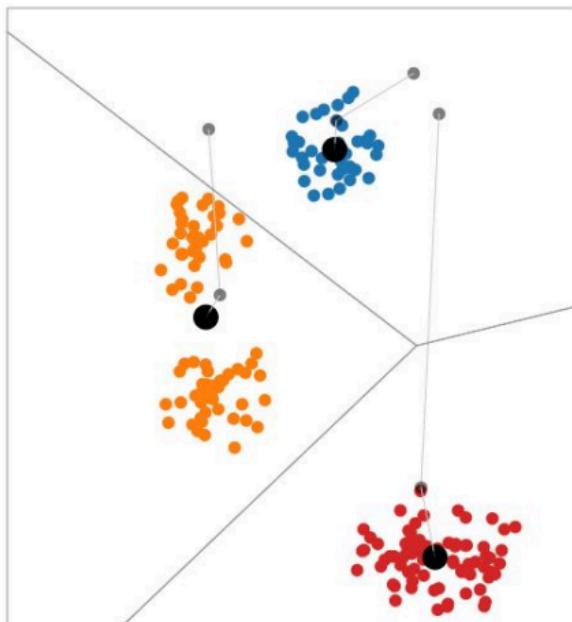


Sensores: Acelerómetros



Se hace necesario realizar procesamiento In Situ





K-Means

<http://alekseynp.com/viz/k-means.html>

Optimización de modelos en Tensorflow Lite

TensorFlow Lite es un framework multi plataforma listo para producción en el despliegue de modelos de AA en dispositivos móviles y sistemas embebidos.



arm

¿Qué puede hacer Tensorflow Lite?

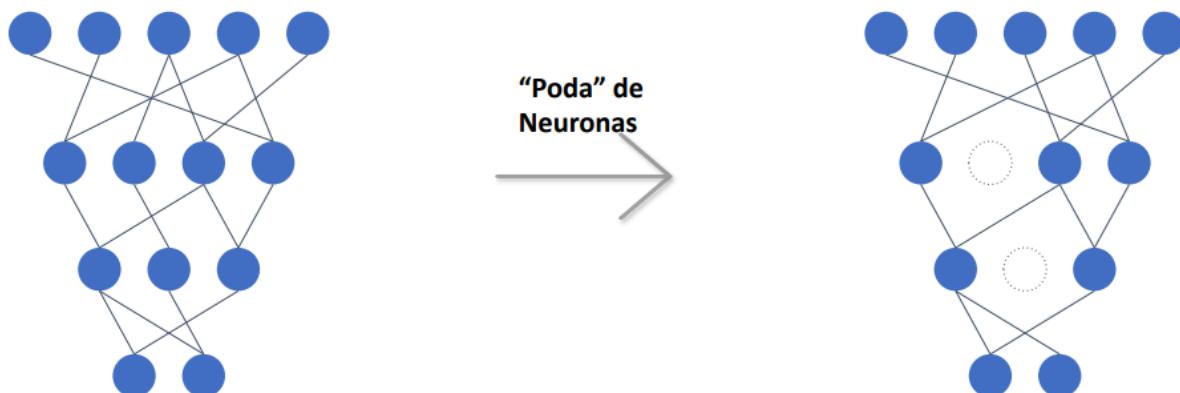
- Optimizar sus modelos
- Disponer de diferentes herramientas para integración fácil en dispositivos móviles, embebidos y aplicaciones basadas en microcontroladores.
- Tomar ventajas de aceleradores especiales en hardware como Edge TPU y GPU.
- <https://www.tensorflow.org/lite>

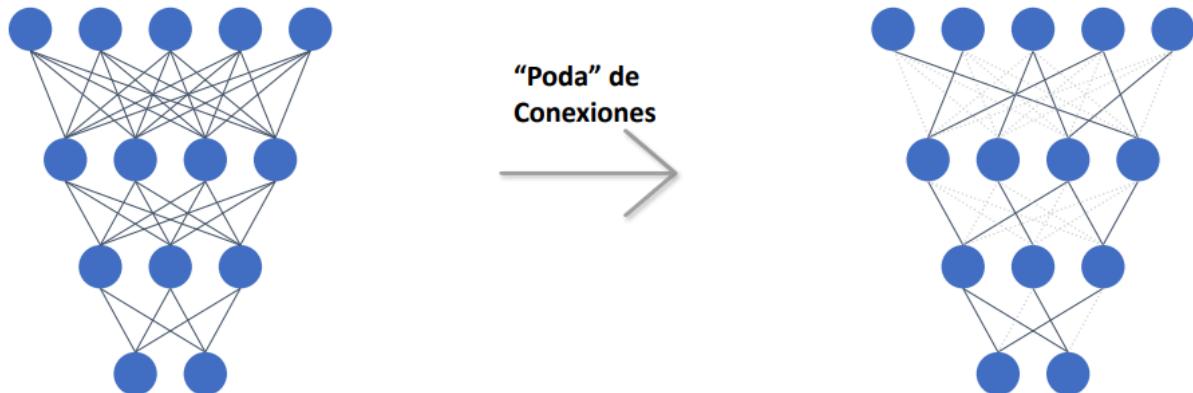
¿Por qué es necesaria?

- Reducción del tamaño
- Reducción de latencia
- Compatibilidad con aceleradores

Optimización del Modelo - Diferentes opciones de optimización en TF Lite

- Pruning (Poda)





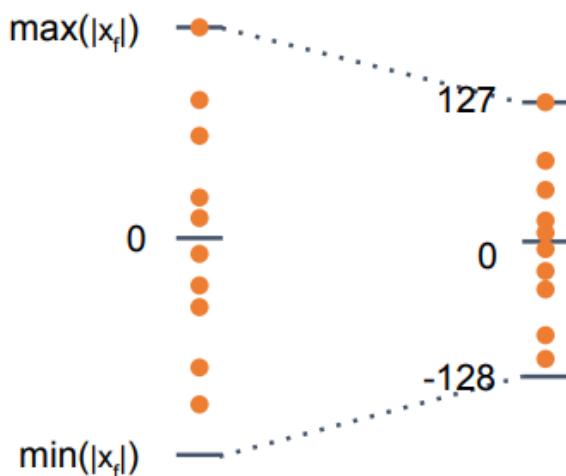
https://www.tensorflow.org/lite/performance/model_optimization#pruning

More info: [An introduction to weight pruning by Tivadar Danka](#)

- Quantization

La cuantificación es una optimización que funciona reduciendo la precisión de los números utilizados para representar los parámetros de un modelo, que por defecto son números de coma flotante de 32 bits. Esto da como resultado un tamaño de modelo más pequeño, una mejor portabilidad y un cálculo más rápido.

Reduciendo la precisión

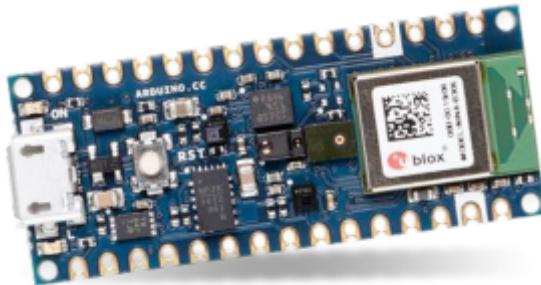


¿Por qué hacemos la cuantización?



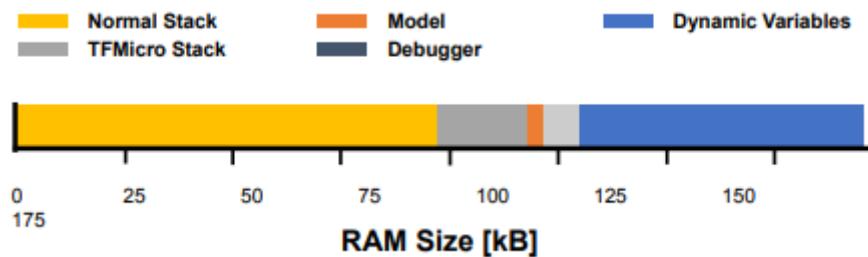
- **Tamaño**

Tamaño de almacenamiento: los modelos de red neuronal más pequeños ocupan menos espacio de almacenamiento en el dispositivo, y al pasar de 32 bits a 8 bits, obtenemos fácilmente una reducción de 4x en la memoria. Por ejemplo el Nano 33 Ble Sense solo tiene 256KB de RAM (memoria) y 1MB de Flash (almacenamiento)



- **Almacenamiento & Tamaño de RAM**

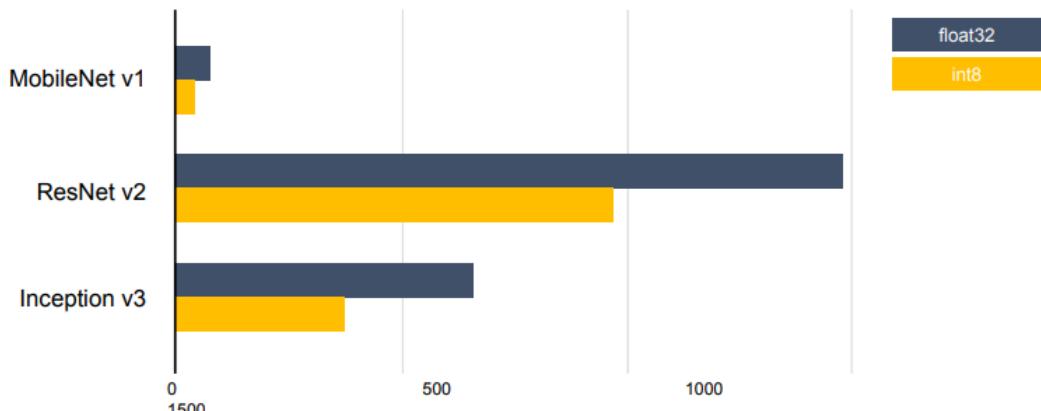
Menos uso de memoria: los modelos más pequeños usan menos RAM cuando se ejecutan, lo que libera memoria para que la usen otras partes de su aplicación y puede traducirse en un mejor rendimiento y estabilidad.



- **Latencia**

Formato Int8 (vs. fp32) severamente reduce el cálculo para ejecutar la inferencia utilizando un modelo, lo que resulta en una latencia más baja. Las optimizaciones de latencia también pueden tener un impacto notable en el consumo de energía.

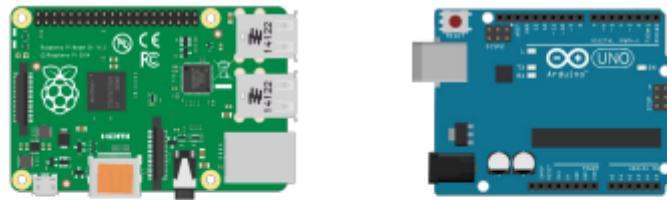
Int8 vs Float (Tiempo de CPU por inferencia)



Quantized Modelos cuantizados son hasta 2–4x más rápidos en CPU y 4x mas pequeños.

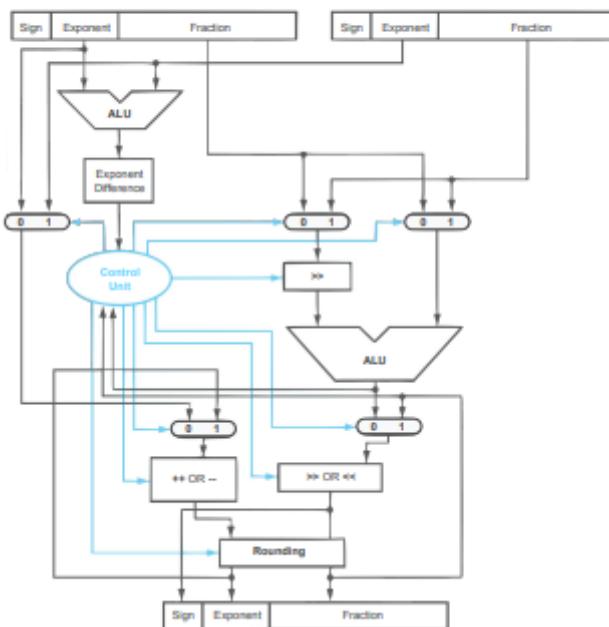
- Portabilidad

No todos los sistemas integrados son iguales. Se suele sacrificar la portabilidad entre sistemas por eficiencia.



Specific HW Implementation of a Library

Single Precision IEEE 754 Floating-Point Standard

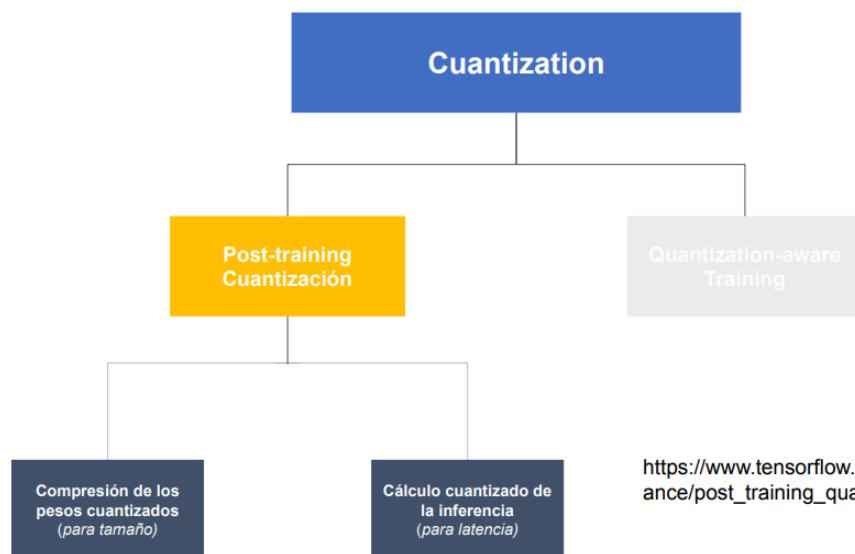
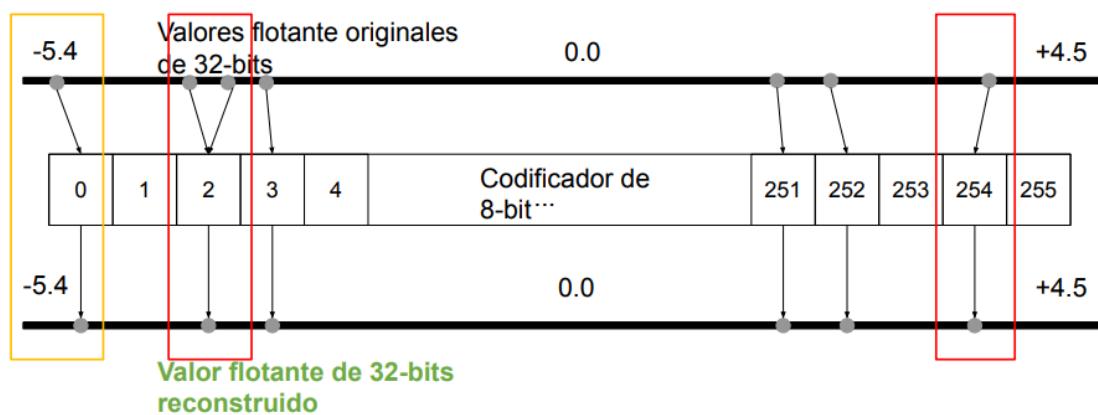
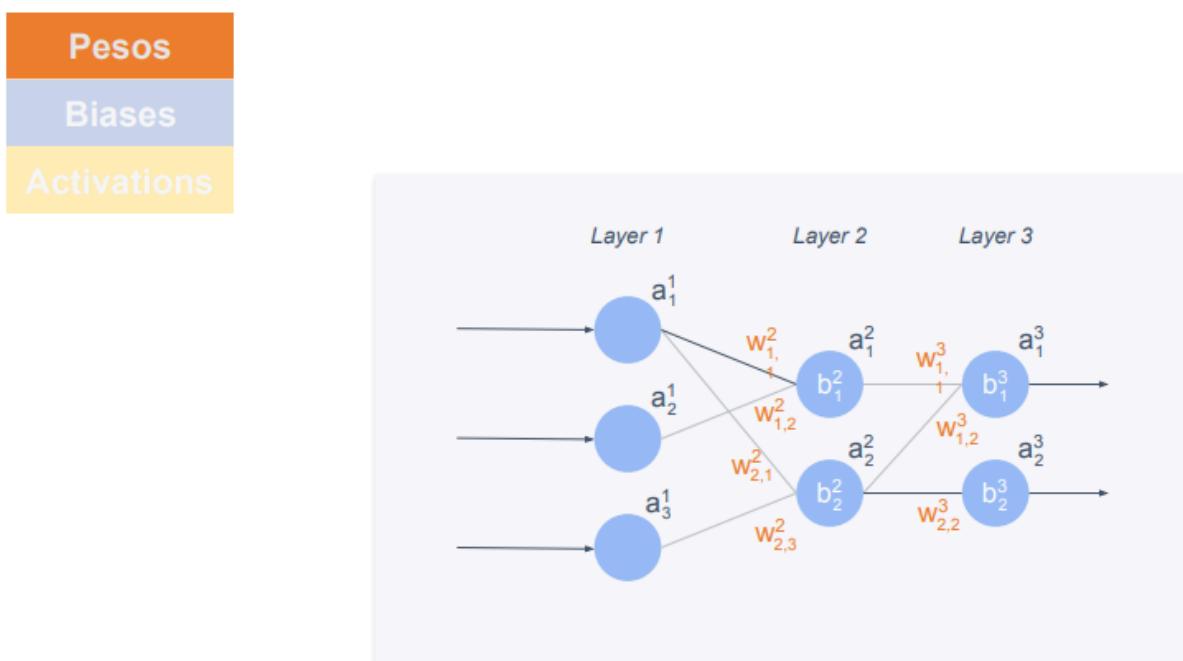


Option 2

Lower Code Portability

Cost (\$)	✓
Power (W)	✓
Eng. Effort	✓

¿Cómo hacemos la cuantización?



Descomprima cada valor de peso de un entero de 8 bits a un valor de punto flotante fp32 antes multiplicándolo por el valor de entrada:

```
output = ... inputn * decompress(q_weightn)
```

Donde:

```
decompress(quantized_code) {
    return float((quantized_code / 255.0) * (max - min)) + min;
}
```

Imagine que reducimos artificialmente la precisión de cada entrada al producto punto, de modo que ya no utilicen el rango completo de un flotante de 32 bits:

```
output = ... quantize(inputn, step) * quantize(weightn, step)
```

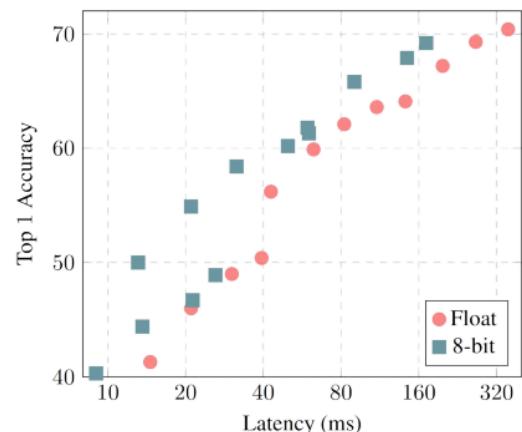
Donde:

```
quantize(x, step) {
    return round(x*step) / step;
}
```

e.g., quantize(3.14, 1.0) = 3.0 (rounding to nearest whole number)
and quantize(3.14, 0.1) = 3.1 (rounding to nearest 1/10)

Compromiso Precisión-Latencia

La cuantificación funciona bien, pero el rendimiento puede sufrir una pérdida de precisión durante la inferencia.

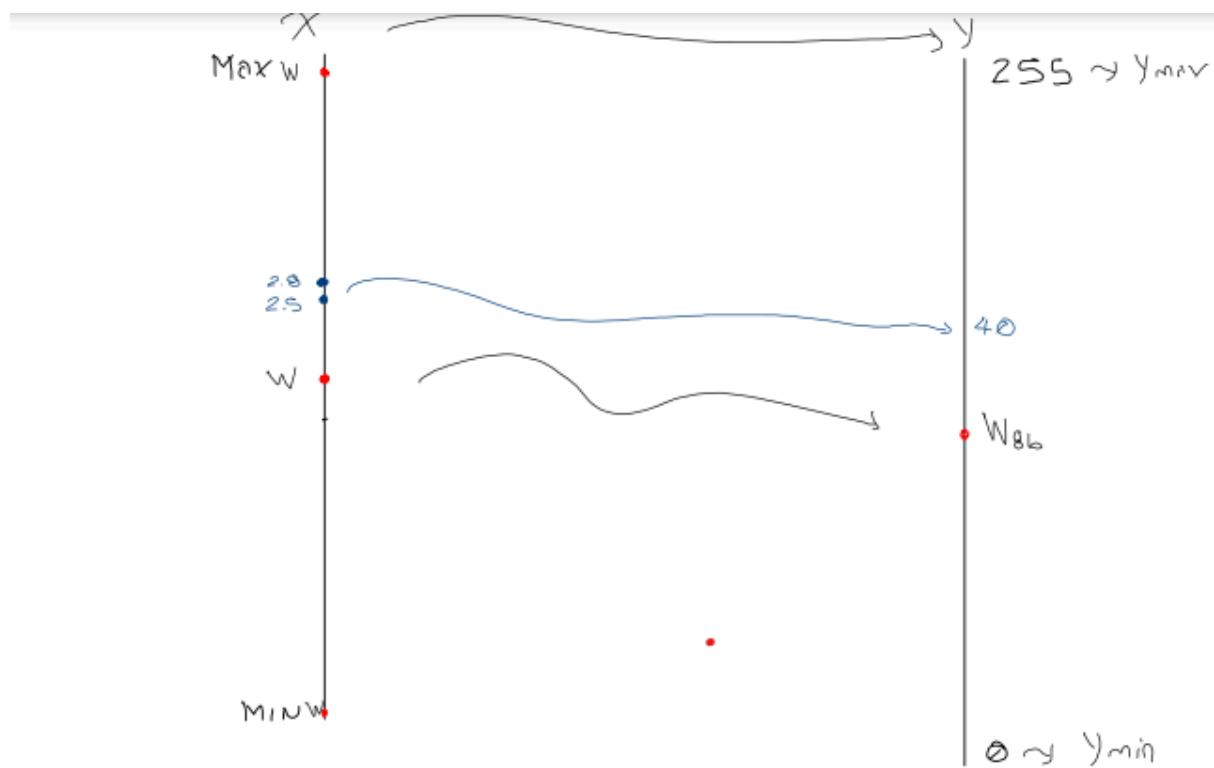


	Floating-point Baseline	Post-training Quantization (PTQ)	Accuracy Drop
MobileNet v1 1.0 224	71.03%	69.57%	▼1.46%
MobileNet v2 1.0 224	70.77%	70.20%	▼0.57%
Resnet v1 50	76.30%	75.95%	▼0.35%

Resumen

Hacer todos los cálculos en números enteros de ocho bits ofrece algunas ventajas convincentes:

- **Aritmética más rápida.** Necesita muchas menos compuertas para implementar un sistema de suma y multiplicación de enteros en ocho bits que una operación de coma flotante de 32 bits.
- **Menor demanda de memoria.** Solo accedemos a ocho bits en lugar de treinta y dos, lo que reduce la carga en el sistema de memoria en un 75 %.
- **Requisitos de recursos reducidos.** Muchos microcontroladores y DSP de gama baja carecen de hardware de punto flotante, por lo que evitarlos aumenta la portabilidad.



$$Y = (Y_{\max} - Y_{\min}) \left(\frac{W - W_{\min}}{MaxW - MinW} \right) + Y_{\min}$$

$$Y = \text{Round} \left(255 \left(\frac{W - MinW}{MaxW - MinW} \right) \right)$$

$$\begin{array}{c}
 \left| \begin{array}{cccc} 9.56 & -4.28 & 2.51 & -1.97 \\ \vdots & & & \\ 6.7 & -3.5 & 5.2 & -4.0 \end{array} \right|_{3 \times 4} \xrightarrow{} \left| \begin{array}{cccc} 245 & 5 & 50 & 10 \end{array} \right|_{3 \times 4} \\
 W_{32\text{bits}} \qquad \qquad \qquad W_{8\text{bits}}
 \end{array}$$

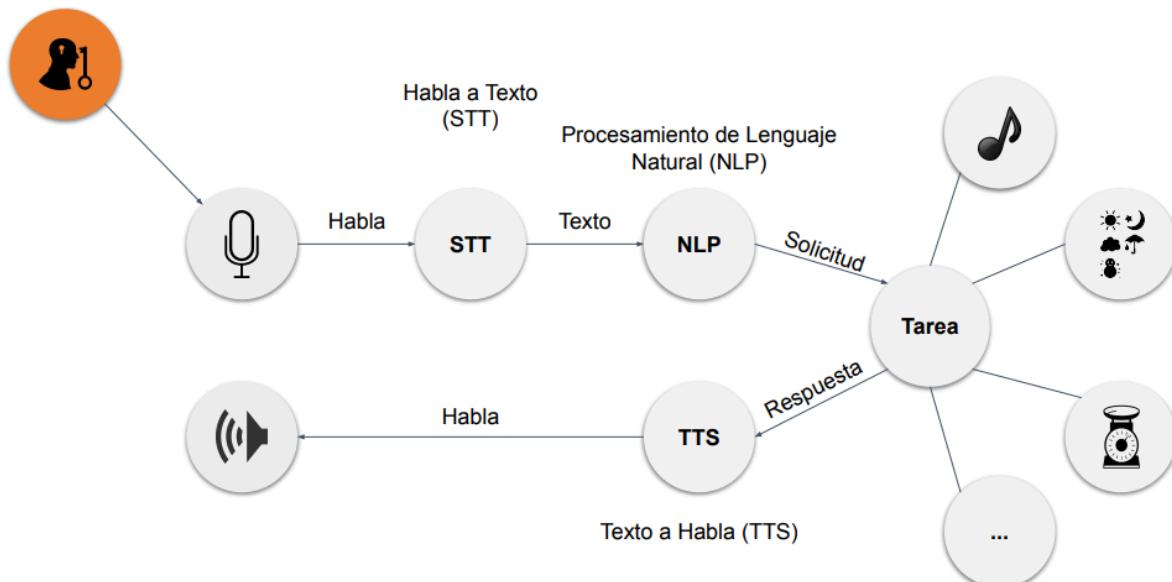
$$W = \frac{W_{8b}}{255} (MaxW - MinW) + MinW$$

Detección de palabras clave (Keyword Spotting)

Keyword Spotting vs. Reconocimiento General del Habla

La detección de palabras clave es uno de los ejemplos más exitosos de TinyML

- Bajo consumo, continuo, en el dispositivo
- El ASR general aún requiere modelos más grandes que consumen mucha energía.
Algunos pueden ejecutarse en dispositivos móviles



Desafíos y Limitaciones



Latencia y Ancho de Banda



Precisión y Personalización



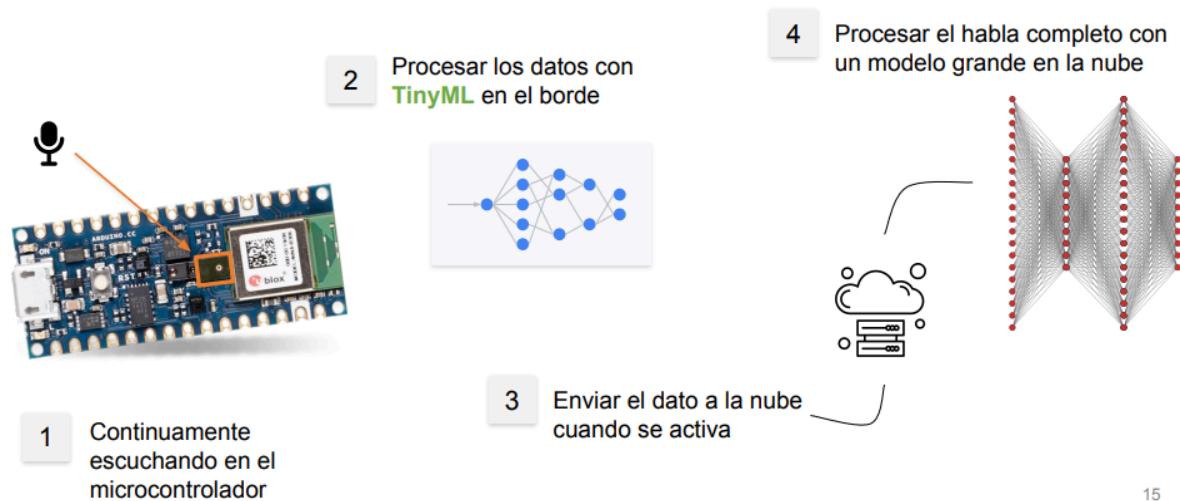
Seguridad y Privacidad



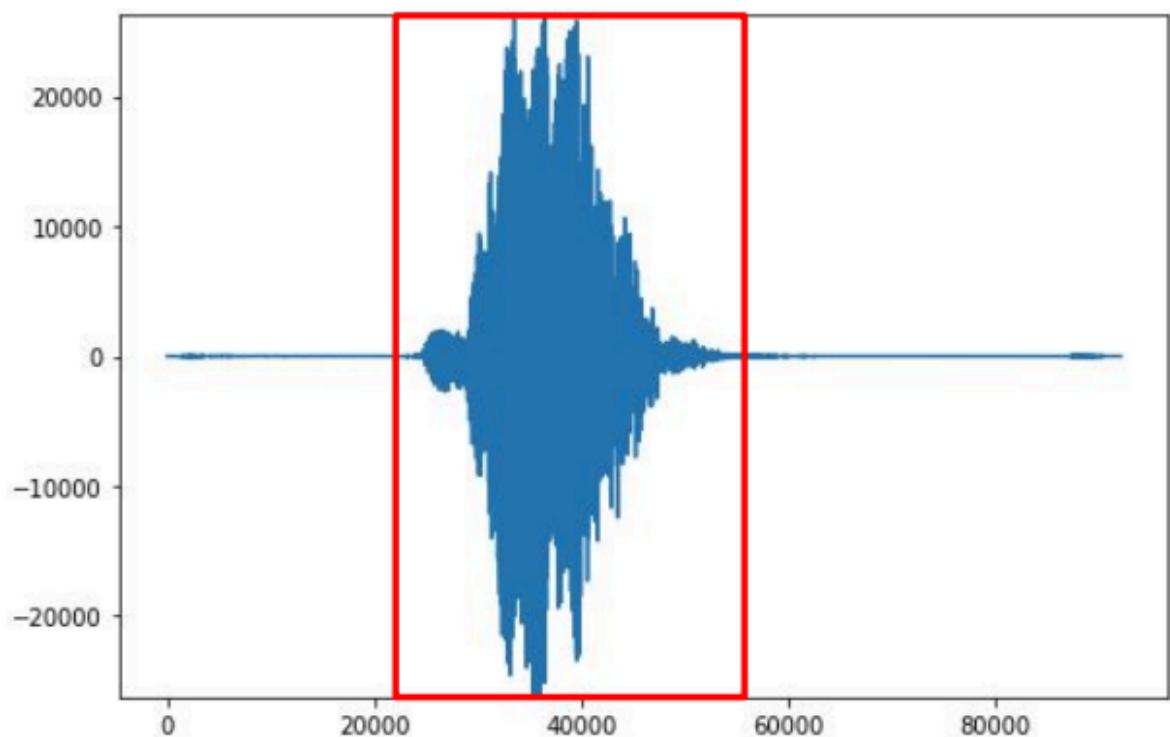
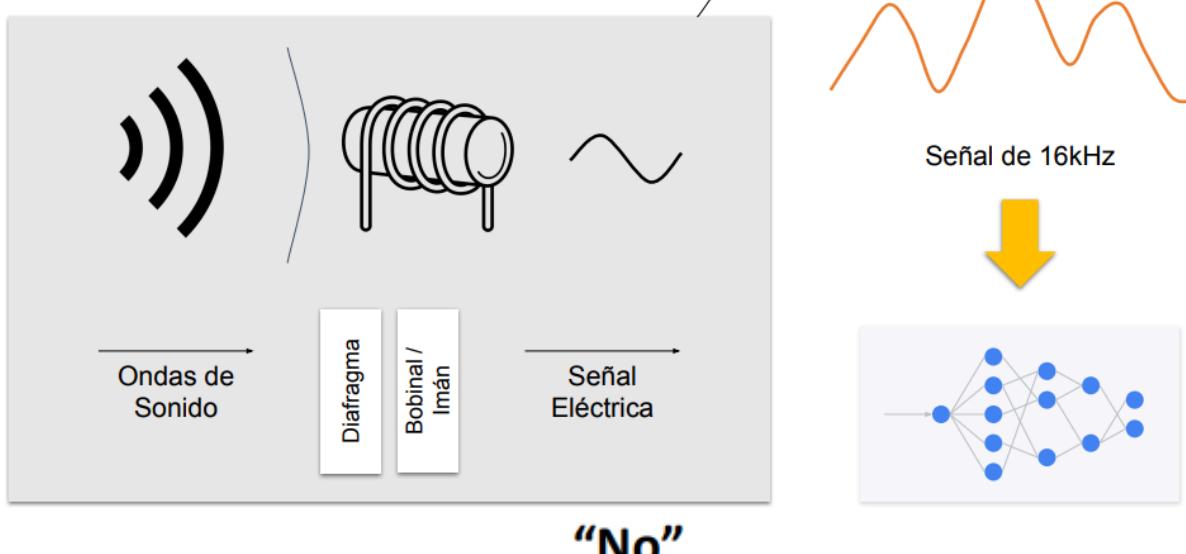
Batería y Memoria

- **Latencia:** Proporcione resultados rápidamente, responda en tiempo real al usuario
- **Ancho de banda:** Minimizar los datos enviados a través de la red (lento y costoso)
- **Precisión:** Escuche continuamente, pero solo se active en el momento adecuado.
- **Personalización:** Activación para el usuario y no para el ruido de fondo.
- **Seguridad:** Salvaguardar los datos que se envían a la nube.
- **Batería:** Energía limitada, funciona con baterías tipo moneda.
- **Memoria:** Ejecutar en dispositivos con recursos limitados

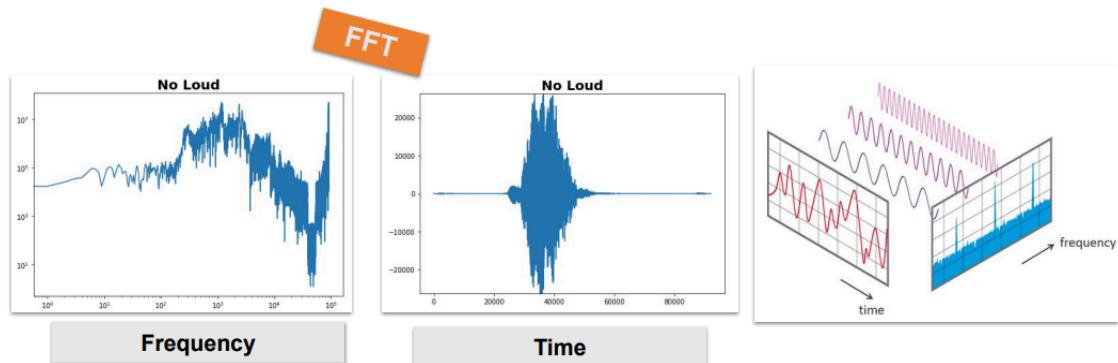
Anatomía de una Aplicación de KWS



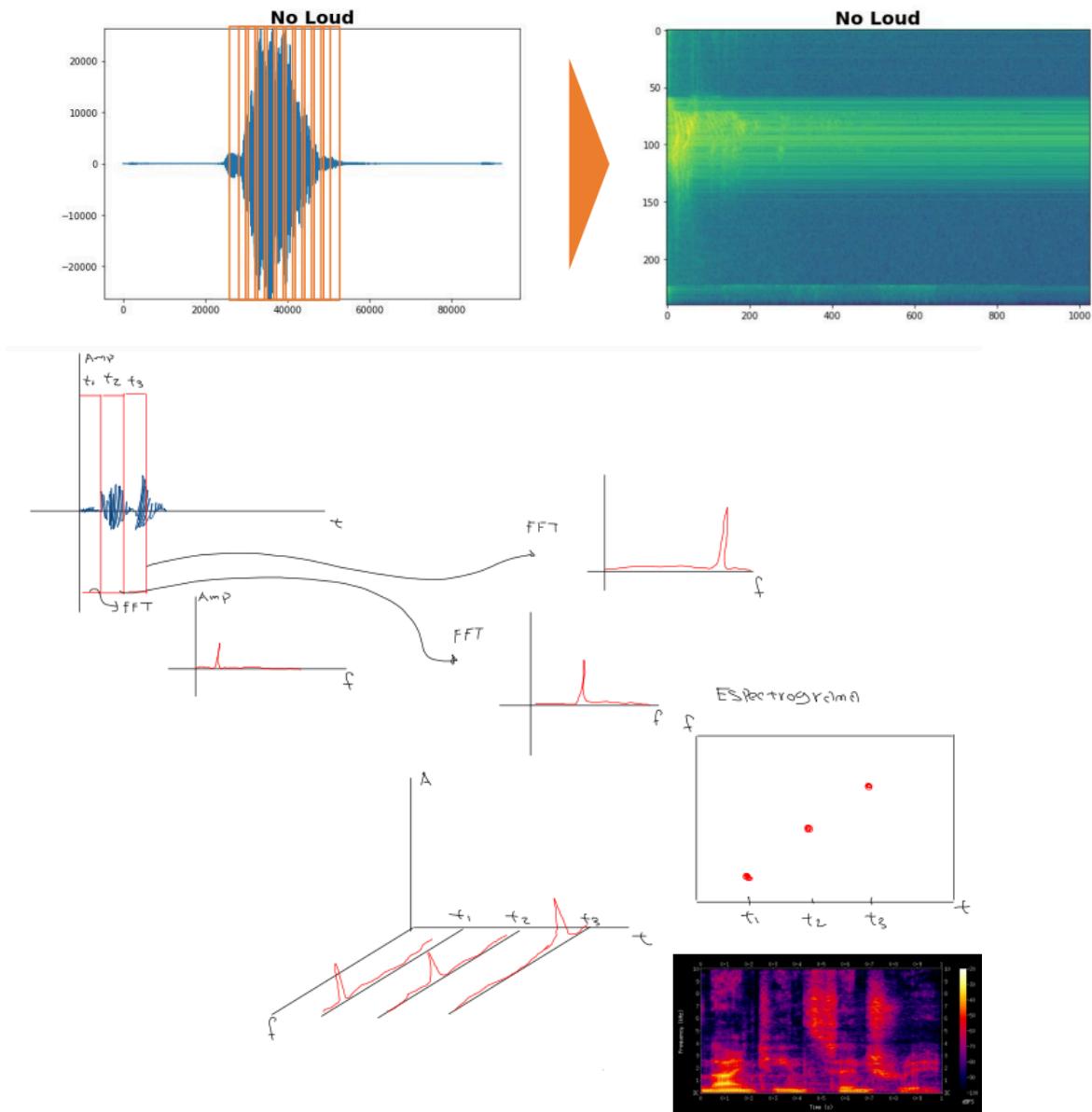
Sensor = Micrófono



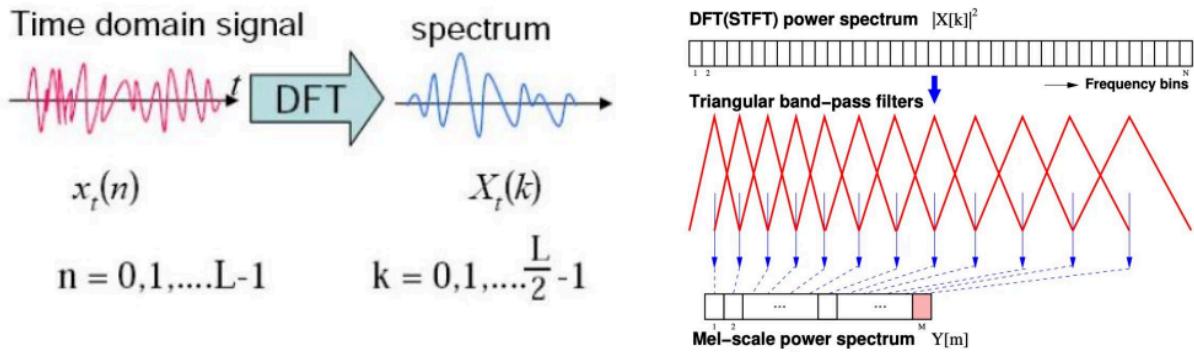
Cómo Procesar la Señal Sonora?



Preprocesamiento de Datos : Espectrogramas

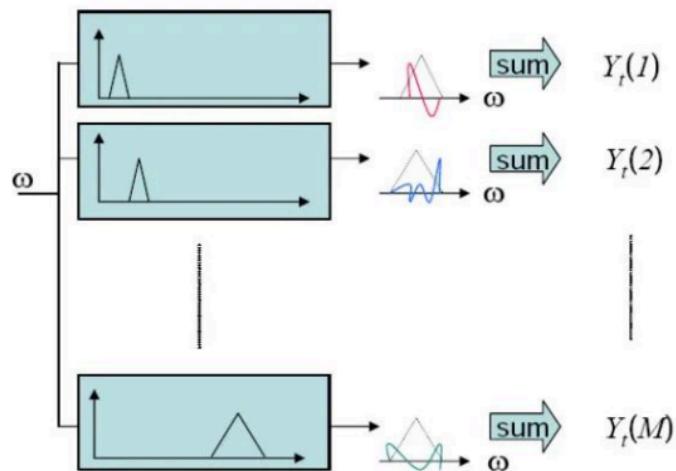


Preprocesamiento de Datos : MFCC



$$Y_t[m] = \sum_{k=1}^N W_m[k] |X_t[k]|^2$$

where k : DFT bin number ($1, \dots, N$)
 m : mel-filter bank number ($1, \dots, M$)



Ejemplo de MFCC

