

PyTorch for Deep Learning Bootcamp

[Notas del Curso]

Instructor: Andrei Neagoie y Daniel Bourke

Plataforma: Udemy

Enlace: <https://www.udemy.com/course/pytorch-for-deep-learning/>

Fecha de Inicio: 25/12/24

Objetivo:

Aprender los fundamentos de PyTorch para el desarrollo de modelos de aprendizaje profundo

Temario:

1. Introducción
2. Fundamentos de PyTorch
3. Redes Neuronales Convolucionales
4. Redes Neuronales Recurrentes
5. Aprendizaje Automático Avanzado
6. Proyectos



Machine Learning

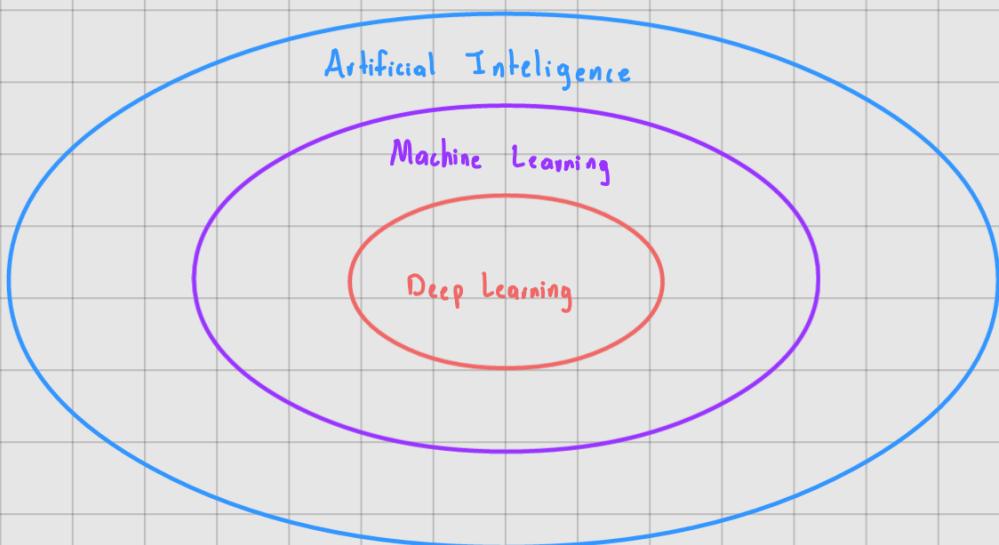
Es una rama de la Inteligencia Artificial (AI) centrada en tratar de emular el aprendizaje humano, a través de algoritmos y herramientas computacionales, con el propósito de automatizar tareas y mejorando su precisión a través de la exposición a información.

En otras palabras, son algoritmos encargados de convertir datos en números y encontrar patrones en ellos.

Deep Learning

Es un subconjunto de ML que utiliza redes neuronales artificiales para "aprender" de los datos. Algunos de los algoritmos más populares son:

- Redes Neuronales Convolucionales (CNN)
- Redes de Memoria Larga a Corto Plazo (LSTM)
- Redes Neuronales Recurrentes (RNN)
- Redes Generativas Adversativas (GAN)
- Redes de Base Radial (RBFN)
- Perceptrón Multicapa (MLP)
- Mapas Auto-organizados (SOM)



Programación Tradicional vs Machine Learning

- Programación Tradicional:



1. Cortar vegetales
2. Sazonar el pollo
3. Precalentar el horno
4. Cocinar el pollo por media hora
5. Añadir vegetales



Inputs

t Reglas

Output

- Algoritmo de Machine Learning:



1. Cortar vegetales
2. Sazonar el pollo
3. Precalentar el horno
4. Cocinar el pollo por media hora
5. Añadir vegetales

Inputs

t Output

Reglas

Un algoritmo de machine learning se encarga de encontrar patrones dada información de entrada y salida deseada.

¿Por qué usar Machine Learning?

Para problemas muy complejos facilita la obtención de modelos predictivos. Es ideal para problemas donde se tiene a disposición una gran cantidad de información.

ML y DL pueden utilizarse para cualquier problema, siempre y cuando sea posible convertir la información en números y sea programable para encontrar patrones en ellos. Sin embargo, de acuerdo con Google:

Dude que no sea tan simple :P

"If you can build a [↓] simple ruled-based system that doesn't require machine learning, do that."

Proveniente de "Google's Machine Learning Handbook".

¿Para qué es útil el Deep Learning?

- **Problemas con largas listas de pasos**: Cuando el acercamiento tradicional falla ML/DL puede ayudar.
- **Ambientes en constante cambio**: El DL puede adaptarse ("Aprender") a nuevos escenarios.
- **Desubrir patrones con enormes colecciones de datos**: A mayor cantidad de datos, mayor precisión.

¿Para qué no es recomendable ML/DL?

- **Cuando se necesita explicar**: Los patrones aprendidos por los modelos no suelen ser interpretables.
- **Cuando el acercamiento tradicional es mejor**: Si se puede obtener con un sistema simple de reglas.
- **Cuando los errores son inaceptables**: Los modelos son probabilísticos, si la precisión es inaceptable.
- **Cuando no se tiene suficiente data**: Para tener buena precisión se necesitan muchos datos.

Machine Learning vs Deep Learning

Data en un formato
estandarizado de
filas y columnas

los algoritmos tradicionales de ML se utilizan en **datos estructurados** y uno de sus algoritmos más poderosos es "gradient boosted machine" como **XGBoost**. Se utiliza en ambientes de producción (en el día a día).

Mientras que los algoritmos de DL son usados en **datos no estructurados**, suelen utilizarse **redes neuronales artificiales**.

Audios
Imágenes
Videos

Algunos de los algoritmos más utilizados son:

a) Datos Estructurados

- Random Forest
- Gradient Boosted Models
- Naive Bayes
- Nearest Neighbour
- Support Vector Machine

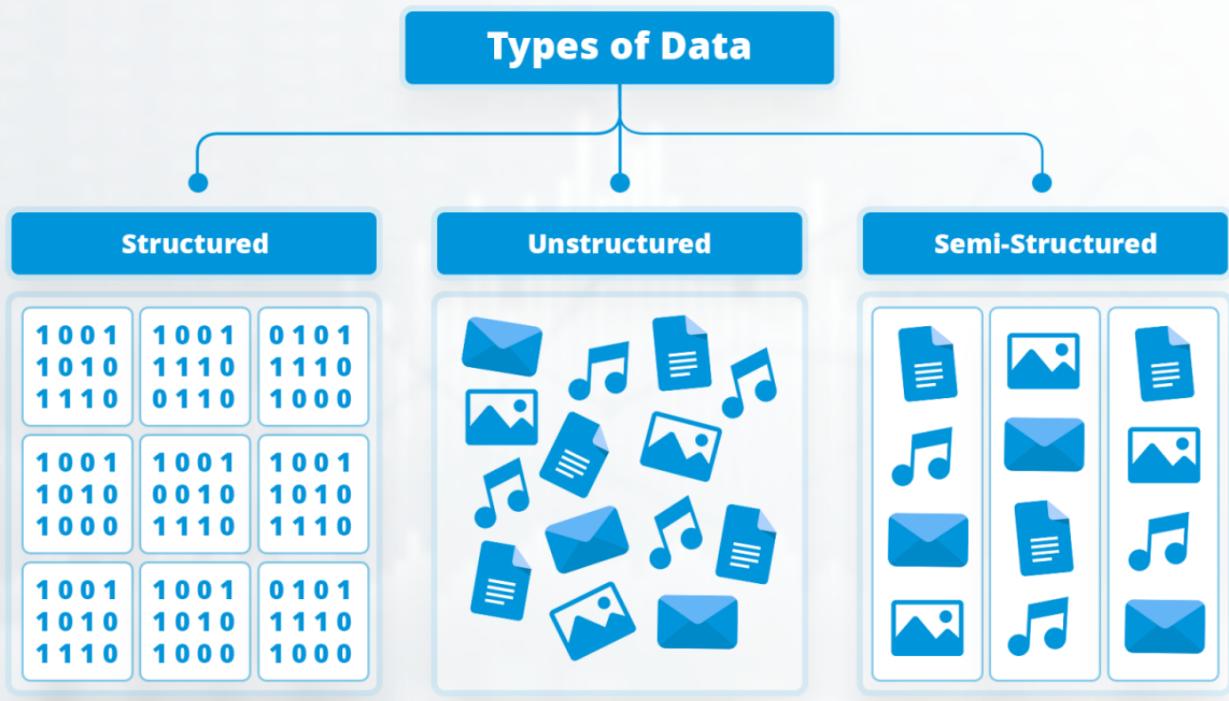
b) Datos No Estructurados

- Para este curso
- Neural Networks ← estudiaremos estos
 - Fully Connected Neural Networks
 - Convolutional Neural Networks
 - Recurrent Neural Networks
 - Transformer

Los algoritmos de DL se destacan por poseer múltiples capas de algoritmos.

Dependiendo de cómo se plantea el problema o resolver, se pueden aplicar distintos algoritmos de todo tipo.

Datos Estructurados vs No Estructurados



Astera
Enabling Data-Driven Innovation

Structured Data vs Unstructured Data

Can be displayed in rows, columns and relational databases



Numbers, dates and strings



Estimated 20% of enterprise data (Gartner)



Requires less storage



Easier to manage and protect with legacy solutions



Cannot be displayed in rows, columns and relational databases



Images, audio, video, word processing files, e-mails, spreadsheets



Estimated 80% of enterprise data (Gartner)



Requires more storage



More difficult to manage and protect with legacy solutions



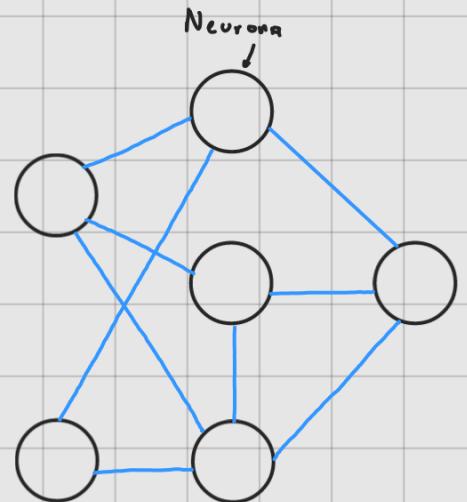
Anatomía de Redes Neuronales



Inputs

$\rightarrow [[116, 78, 15],$
 $[117, 43, 96],$
 $[125, 87, 23]]$

Numerical Encoding
 (Representación a través
 de vector, matriz o tensor)



Learns Representation
 (Patterns, features, weights)

$\rightarrow [[0.983, 0.004, 0.013],$
 $[0.110, 0.889, 0.001],$
 $[0.023, 0.027, 0.985]]$

Representation outputs
 (Features, weight matrix, weight tensors, learn representation)

→ Cat, dog, Kitten

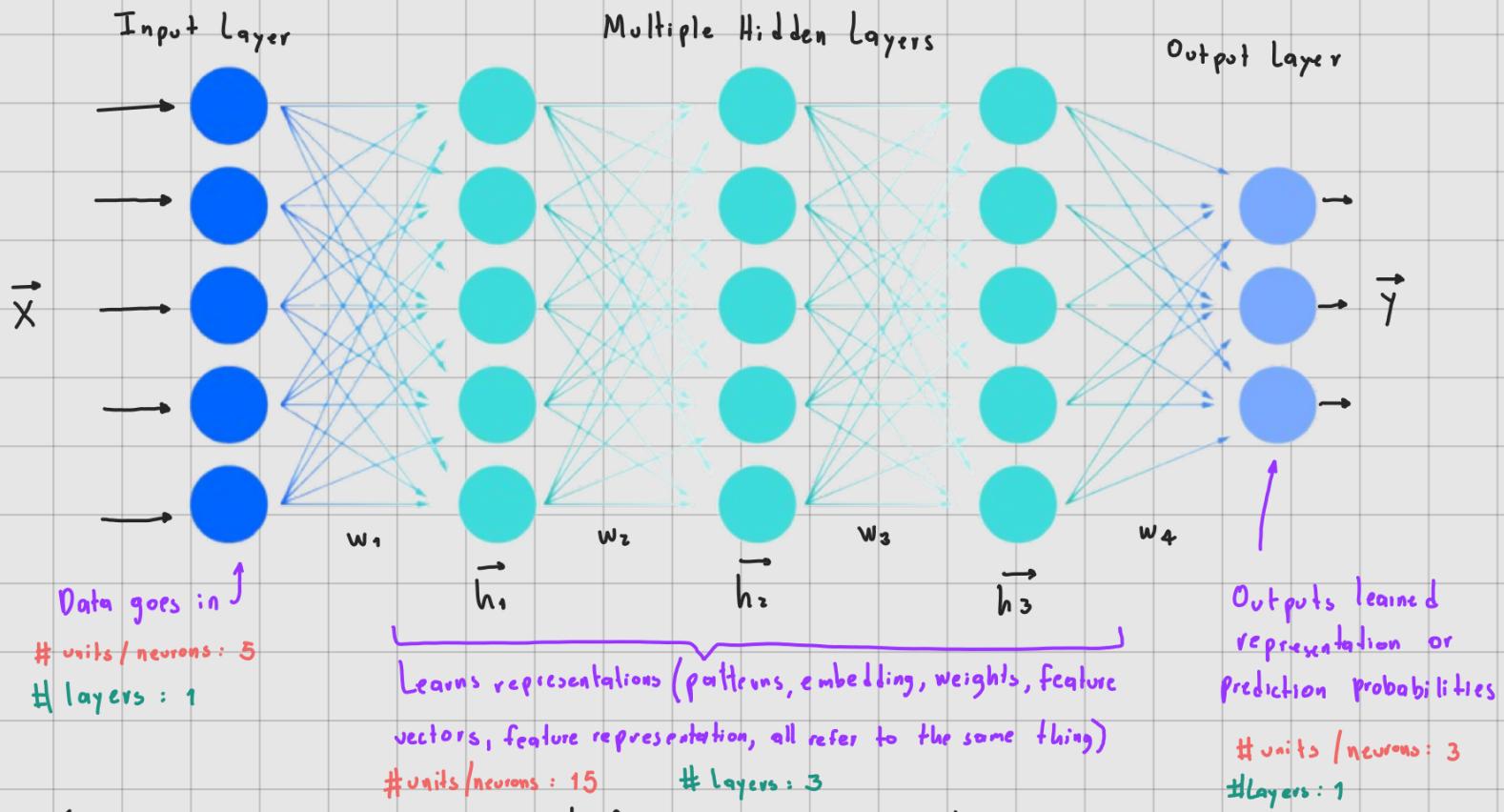
→ Not a natural disaster

→ "Hey Siri, what's the weather today?"

Outputs

Representación
 entendible por
 humanos

A grosso modo, una red neuronal se compone de la siguiente forma:



(Cada capa suele ser una combinación de funciones lineares y no lineares.

Es muy importante elegir la red neuronal apropiada para representar nuestra información. En términos generales para:

- Imágenes - Redes Neuronales Convolucionales
- Audio
- NLP

} Transformer

Procesamiento de lenguaje natural

Sin embargo, todas siguen los mismos fundamentos.

Tipos de Aprendizaje

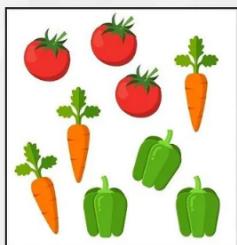
En el curso nos centraremos en este

- Aprendizaje Supervisado:

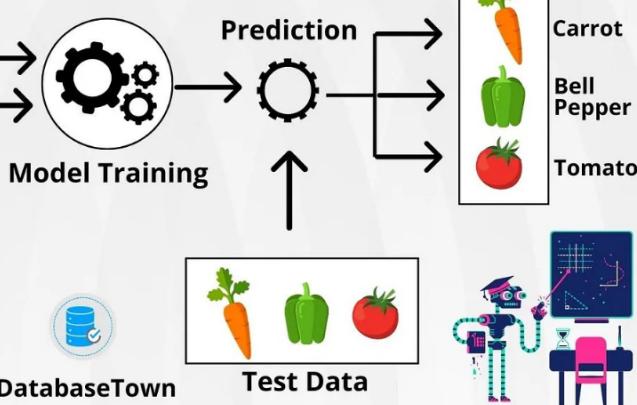
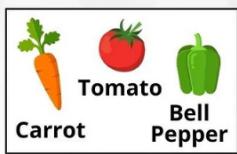
SUPERVISED LEARNING

Supervised machine learning is a branch of artificial intelligence that focuses on training models to make predictions or decisions based on labeled training data.

Labeled Data



Labels



Cuando se tiene data y labels como en el ejemplo del pollo.

Se tiene una gran cantidad de información de tipo inputs y outputs deseados; mientras más datos consuma, mejor será la precisión de la red.

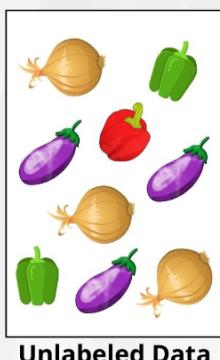
y Semi-Supervisado:

- Aprendizaje No Supervisado:

UNSUPERVISED LEARNING

Unsupervised learning is a type of machine learning where the algorithm learns from unlabeled data without any predefined outputs or target variables.

Input Raw Data



Unlabeled Data

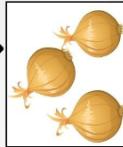
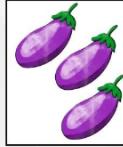
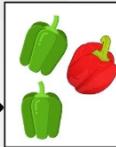
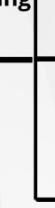
DatabaseTown

Interpretation

Algorithms

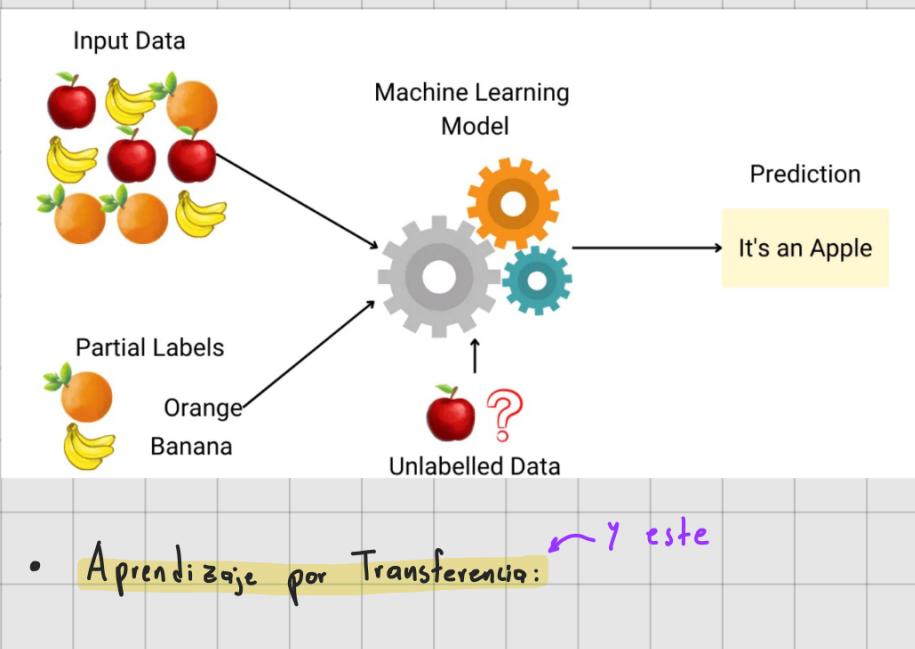


Processing



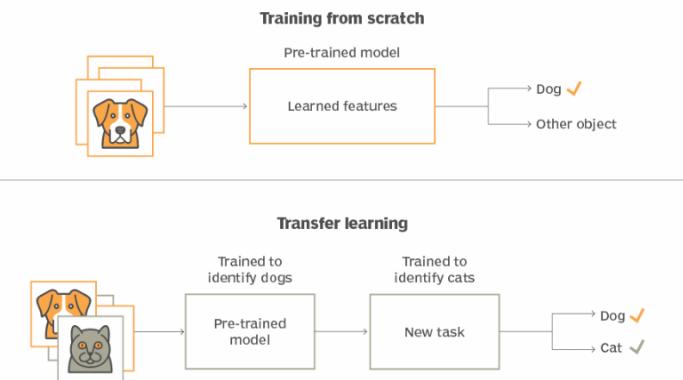
No se tiene un output deseado, ni labels para las variables. Solo se le dan datos de input al algoritmo para que determine el patrón en la data.

- Aprendizaje Semi-Supervisado:



- Aprendizaje por Transferencia:

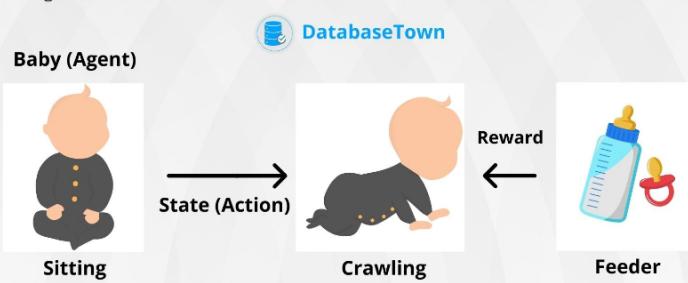
How transfer learning works



- Aprendizaje por Refuerzo:

REINFORCEMENT LEARNING

Reinforcement learning is a machine learning paradigm that focuses on how agents learn to interact with an environment to maximize cumulative rewards.



Algorithms and Approaches in Reinforcement Learning

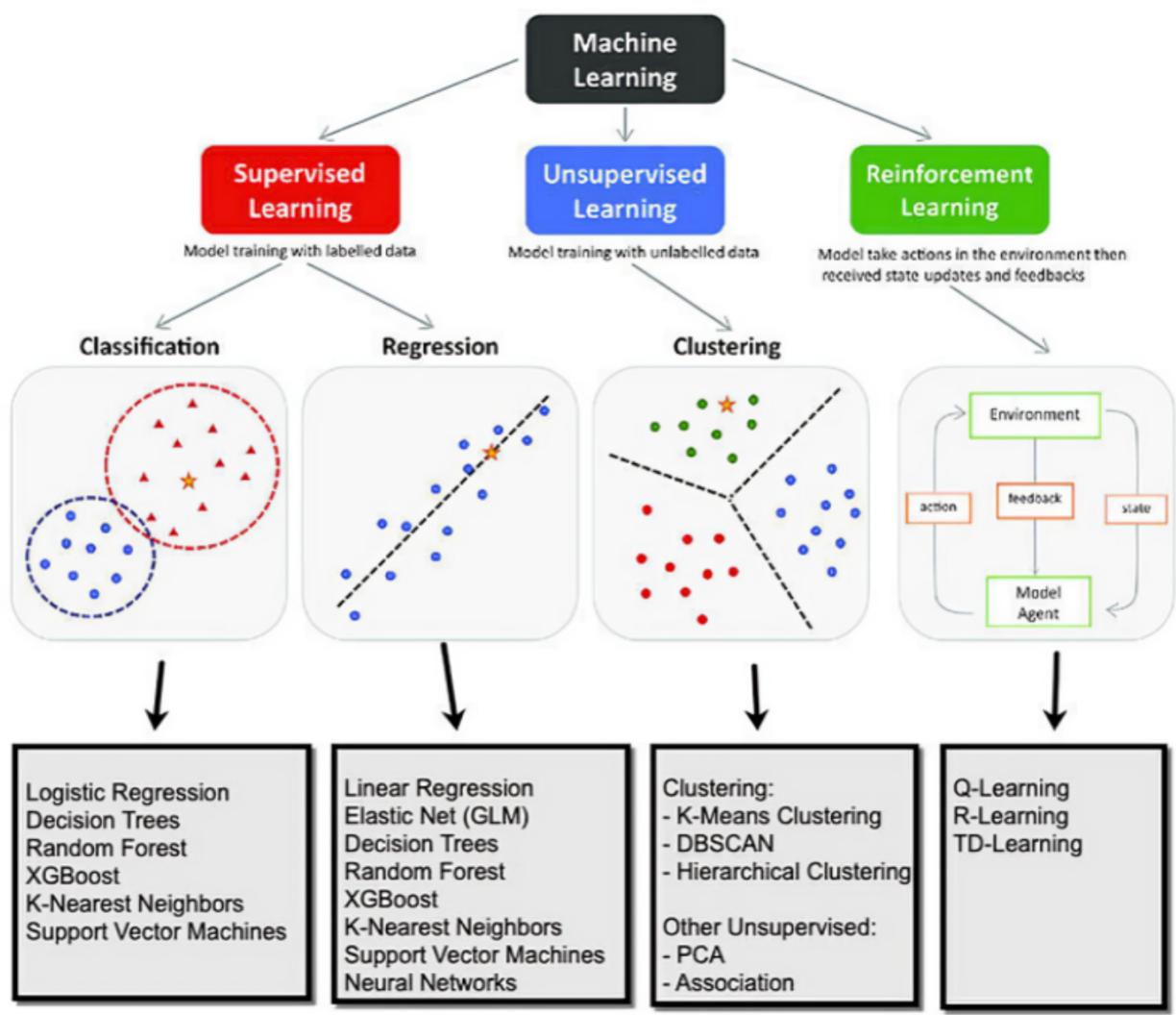
- Q-learning
- Deep Q-networks (DQN)
- Policy Gradients Methods
- Proximal Policy Optimization (PPO)

Se tiene cierta data con labels y una gran parte sin estos labels, el algoritmo determina los patrones en la data y los labels de variables sin clasificar previamente.

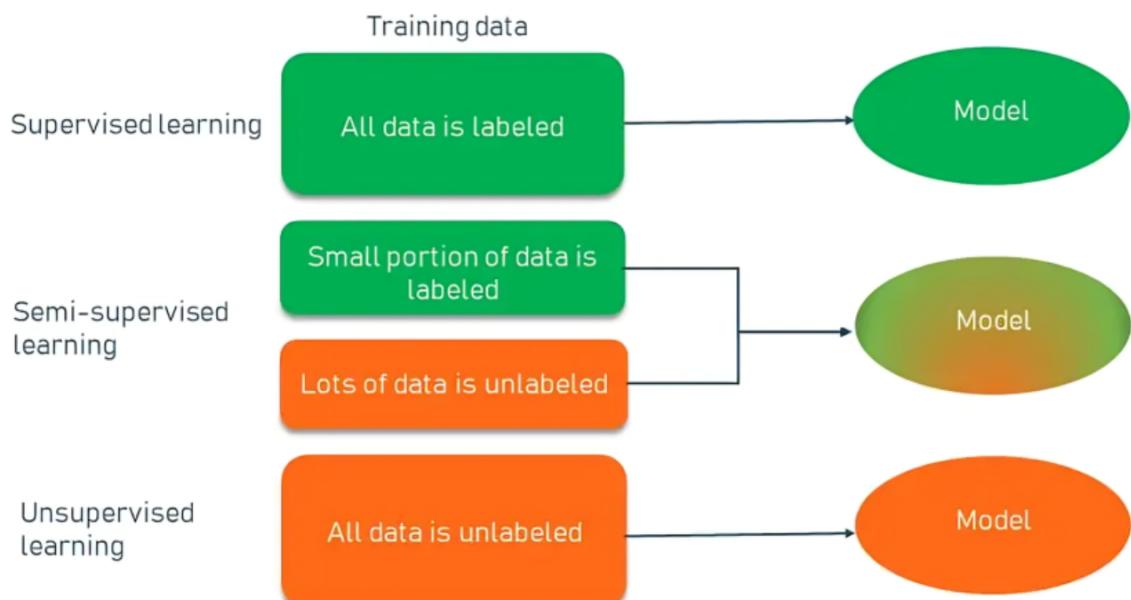
Transfiere los patrones aprendidos de un modelo a otro. Es una de las técnicas de ML/DL más poderosas.

Utiliza un paradigma de "recompensa y castigo" para procesar data. Asemeja un proceso de aprendizaje de prueba y error. Aprende a través de feedback con cada acción y encuentra los mejores caminos de procesamiento para obtener los resultados.

Diferencias entre los Tipos de Aprendizaje



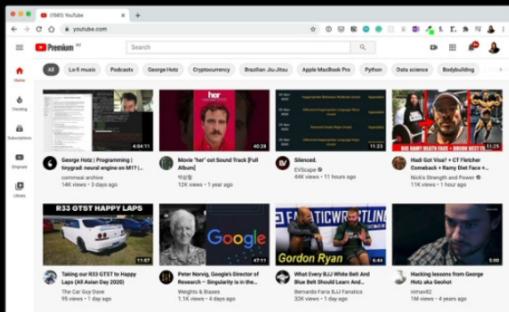
SUPERVISED LEARNING vs SEMI-SUPERVISED LEARNING vs UNSUPERVISED LEARNING



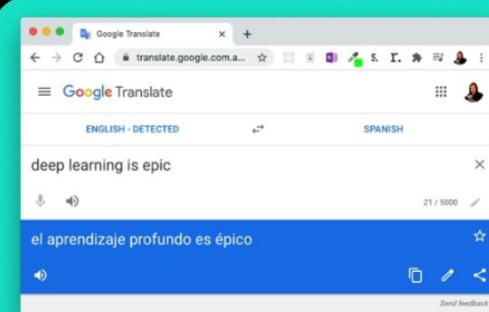
Aplicaciones de Deep Learning

(some)

Deep Learning Use Cases



Recommendation



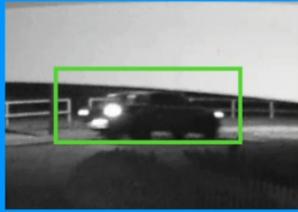
Translation



"Hey Siri, who's the biggest big dog of them all?"

Speech recognition

Sequence to sequence
(seq2seq)



Computer Vision

To: daniel@mrdourke.com
Hey Daniel,

This deep learning course is incredible!
I can't wait to use what I've learned!

Not spam

To: daniel@mrdourke.com
Hay daniel...

Congratulations! U win \$1139239230

Spam

Natural Language Processing (NLP)

Classification/regression

Udemy

Fundamentos de Pytorch

"Pytorch is an optimized library for deep learning using GPUs and CPUs."



Algunas de las características y de PyTorch son:

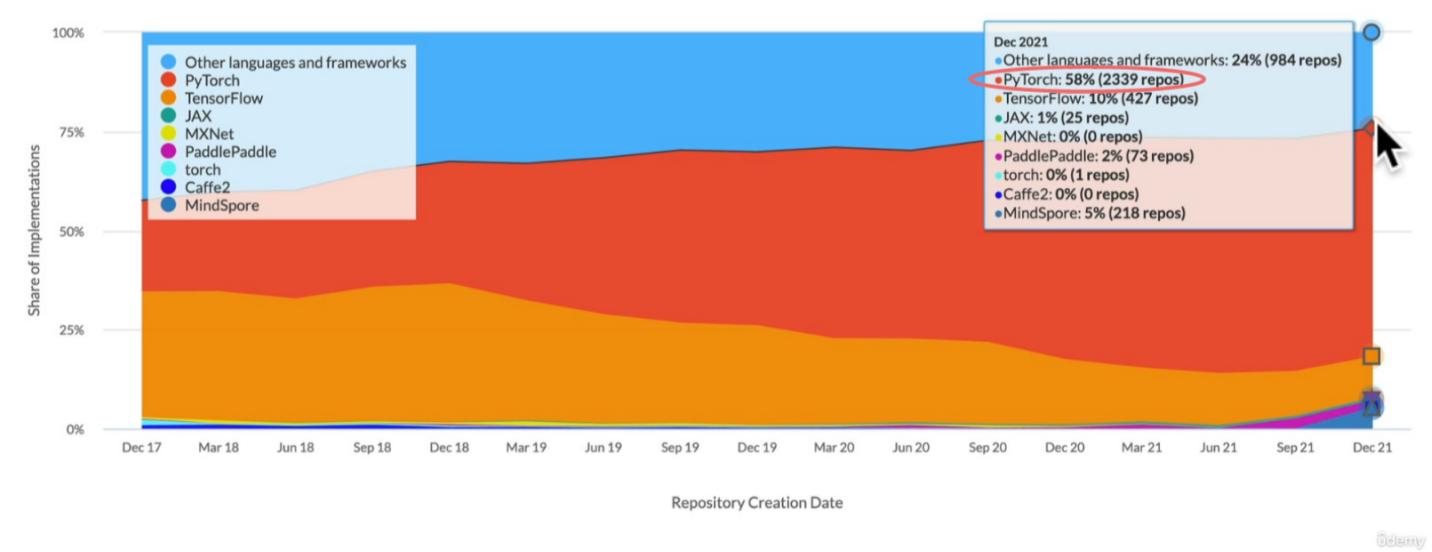
- Es el framework más popular de DL para investigación.
- Permite escribir rápidamente y eficientemente código de DL en Python.
- Es capaz de correr en 1 o varios GPUs.
- Provee acceso a distintos modelos ya escritos de DL (Torch Hub / torchvision.models)
- Provee soporte a todo el ecosistema de DL : pre-procesamiento de dato, construcción de modelos, despliegue del modelo en la aplicación o en la nube.
- Amigable y sencillo para nuevos usuarios.
- Diseñado por Facebook/Meta , ahora es open-source y utilizado en empresas como: Tesla, Microsoft y Open AI.

¿Por qué PyTorch?

De todos los papeles publicados utilizando DL, PyTorch es el más popular (58%).

Frameworks

Paper Implementations grouped by framework



De acuerdo con François Chollet, creador del framework Keras :

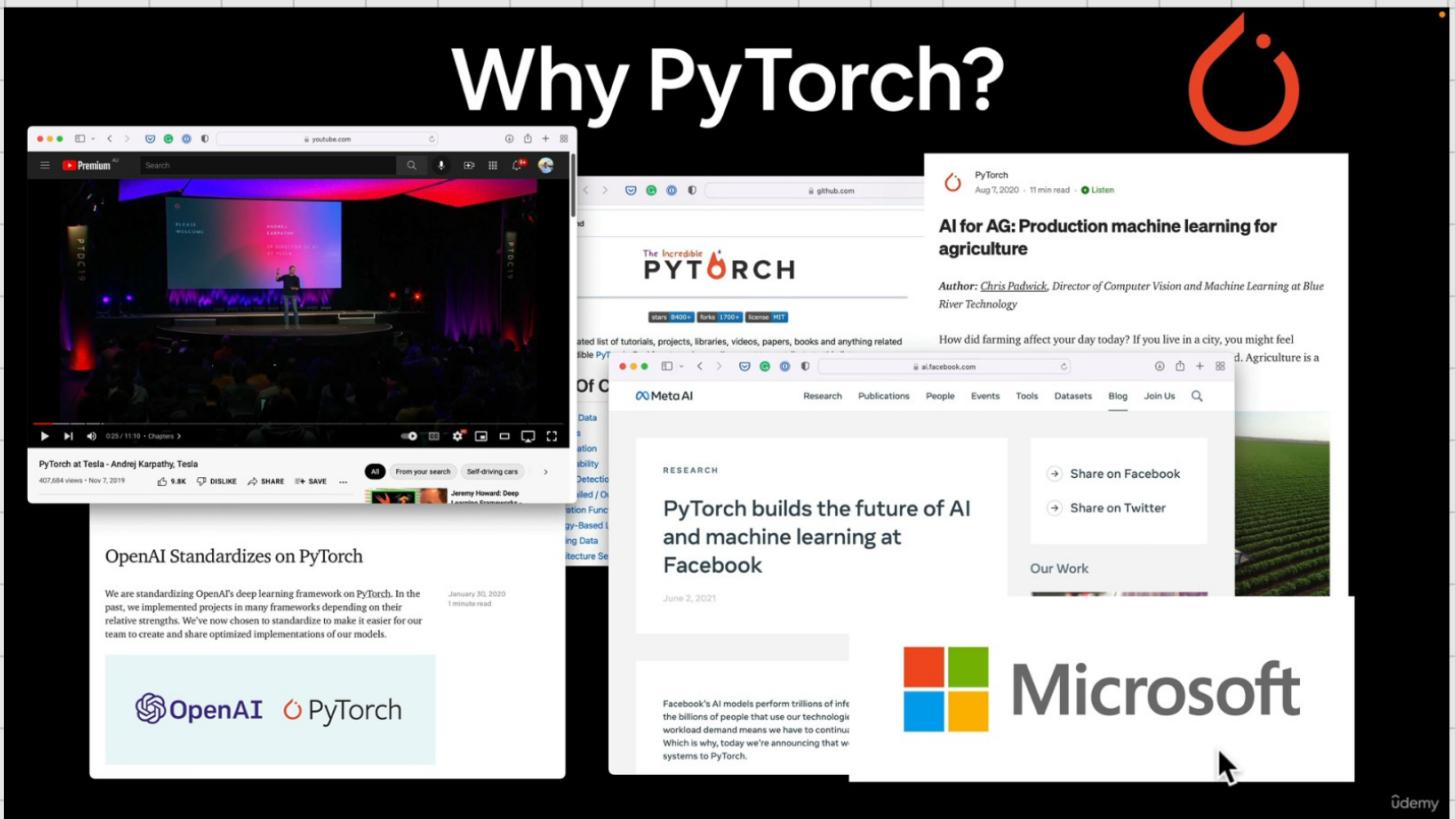
François Chollet (@fchollet) posted on Nov 21, 2020, at 7:03 AM via Twitter for Android:

With tools like Colab, Keras, and TensorFlow, virtually anyone can solve in a day, with no initial investment, problems that would have required an engineering team working for a quarter and \$20k in hardware in 2014

and PyTorch

Source: [@fchollet Twitter](#)

En resumen :



Cabe mencionar que permite ejecutar código de ML/DL acelerado en GPUs y TPUs.

¿Qué es un GPU / TPU?

Un GPU es una unidad de procesamiento de gráficos, lo que esencialmente lo hace muy rápido con cálculos numéricos simples.



Mientras que un TPU, una unidad de procesamiento de tensores, es un circuito integrado con el propósito de acelerar código de AI para una aplicación específica. En comparación con CPUs y GPUs, los TPUs son diseñados para cálculos con mayor volumen de datos pero pierde en precisión.



En general, distintos tipos de procesadores son mejores para distintos modelos de ML/DL .

- TPUs - Redes Neuronales Convolucionales (CNN)
- GPUs - Redes Neuronales Completamente Conectadas (FCNN)
- CPUs - Redes Neuronales Recurrentes (RNN)

Cabe mencionar que ML/DL tienen mucho que ver con tensores .

CUDA



Es una arquitectura de computación paralela, desarrollada por NVIDIA, diseñada para aprovechar la capacidad del procesamiento de gráficos de los GPUs para realizar cálculos intensivos en distintas áreas como: ML/DL, Data Science y el procesamiento de datos. Algunas de sus características son:

- Los algoritmos de ML/DL pueden ser ejecutados simultáneamente en múltiples núcleos de procesamiento del GPU, acelerando significativamente los tiempos de entrenamiento y pruebas.
- Permite utilizar la memoria y los recursos del GPU de manera eficiente, reduciendo la carga del CPU.
- Proporciona funciones optimizadas para operaciones comunes en ML/DL como: convoluciones, multiplicación de matrices y activaciones.

Algunos de los frameworks de ML compatibles con CUDA son:

- Tensorflow
- PyTorch
- Keras
- Caffe
- Microsoft Cognitive Toolkit (CNTK)

¿Qué es un Tensor?

Es una estructura de datos matemática fundamental en el aprendizaje automático ML y el aprendizaje profundo DL. Es una generalización de vectores y matrices, que es capaz de representar datos de cualquier dimensión, en otras palabras es un arreglo multidimensional de números reales:

$$T = t_{i_1, i_2, \dots, i_k} \in \mathbb{R}^{n_1, n_2, \dots, n_k}$$

Donde n_1, n_2, \dots, n_k son las dimensiones del tensor, i_1, i_2, \dots, i_k son los índices que identifican cada elemento del tensor y t_{i_1, i_2, \dots, i_k} es el valor del elemento en la posición i_1, i_2, \dots, i_k .

Podemos clasificarlos de la siguiente forma:

- Escalar - Tensor de Rango 0
- Vector - Tensor de Rango 1
- Matriz - Tensor de Rango 2
- Tensor 3D - Tensor de Rango 3
- Tensor ND - Tensor de Rango n

Dentro de ML / DL se utilizan tensores para representar datos de cualquier dimensión, como imágenes (tensores 3D), secuencias de texto (matrices) o datos de sensor (vectores).

Algunos de los conceptos basados en tensores son:

- **Redes Neuronales :**

Los tensores representan los pesos y los resultados de las operaciones en las redes neuronales.

- **Procesamiento de lenguaje Natural:**

Se utilizan tensores para representar secuencias de texto y realizar operaciones como embedding y atención.

- **Procesamiento de Imágenes:**

Los tensores representan operaciones como convoluciones, pooling y normalización.

- **Optimización:**

Los tensores se utilizan para computar gradientes y realizar actualizaciones de pesos durante el entrenamiento.

Las operaciones realizadas con tensores son:

- **Suma y Resta Tensorial:**

La suma de 2 tensores A y B del mismo orden resulta en un tensor C de mismo orden.

$$C = A + B = B + A$$

Donde los componentes del tensor resultante son representadas por:

$$(C)_{ij} = (A+B)_{ij} \rightarrow C_{ij} = A_{ij} + B_{ij}$$

Y de forma matricial:

$$C = A + B$$

- Multiplicación con un Escalar:

Sea un tensor de segundo orden A y un escalar λ , se define D , tal que:

$$D = \lambda A \rightarrow (D)_{ij} = \lambda (A)_{ij}$$

Con la forma matricial como:

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \rightarrow \lambda A = \begin{pmatrix} \lambda A_{11} & \lambda A_{12} & \lambda A_{13} \\ \lambda A_{21} & \lambda A_{22} & \lambda A_{23} \\ \lambda A_{31} & \lambda A_{32} & \lambda A_{33} \end{pmatrix}$$

También se cumple para cualquier vector \vec{v} :

$$(\lambda A) \cdot \vec{v} = \lambda (A \cdot \vec{v})$$

- Producto Escalar:

El producto escalar de un tensor de 2º orden A y un vector \vec{x} resulta ser otro vector \vec{y} :

$$\begin{aligned} \vec{y} &= A \cdot \vec{x} = (A_{jk} \hat{e}_j \otimes \hat{e}_k) \cdot (x_l \hat{e}_l) = A_{jk} x_l \underbrace{\delta_{kl}}_{\gamma_j} \hat{e}_j \\ &= \underbrace{A_{jk} x_k}_{\gamma_j} \hat{e}_j \rightarrow \vec{y} = \gamma_j e_j \end{aligned}$$

Por otro lado, el producto escalar de 2 tensores de 2º orden A y B es otro tensor de 2º orden, siguiendo la regla $A \cdot B \neq B \cdot A$:

$$(i) C = A \cdot B = (A_{ij} \hat{e}_i \otimes \hat{e}_j) \cdot (B_{kl} \hat{e}_k \otimes \hat{e}_l) = A_{ij} B_{kl} \delta_{jk} \hat{e}_i \otimes \hat{e}_l$$

$$= \underbrace{A_{ik} B_{kl}}_{A \cdot B} \hat{e}_i \otimes \hat{e}_l = C_{il} \hat{e}_i \otimes \hat{e}_l$$

$$(ii) D = B \cdot A = (B_{ij} \hat{e}_i \otimes \hat{e}_j) \cdot (A_{kl} \hat{e}_k \otimes \hat{e}_l) = B_{ij} A_{kl} \delta_{jk} \hat{e}_i \otimes \hat{e}_l$$

$$= \underbrace{B_{ik} A_{kl}}_{B \cdot A} \hat{e}_i \otimes \hat{e}_l = D_{il} \hat{e}_i \otimes \hat{e}_l$$

También cumplen las siguientes propiedades:

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

- Producto Vectorial:

El producto vectorial de un tensor de 2º orden A con un vector \vec{x} resulta en un tensor de 2º orden dado por:

$$A \wedge \vec{x} = (A_{ij} \hat{e}_i \otimes \hat{e}_j) \wedge (x_k \hat{e}_k)$$

$$= \epsilon_{ijk} A_{ij} x_k \hat{e}_i \otimes \hat{e}_l$$

Donde se emplea la definición $\hat{e}_j \wedge \hat{e}_k = \epsilon_{ijk} \hat{e}_l$.

- Convolución:

Aplicación de filtros a tensores.

- Pooling :

Reducción de dimensionalidad de vectores.

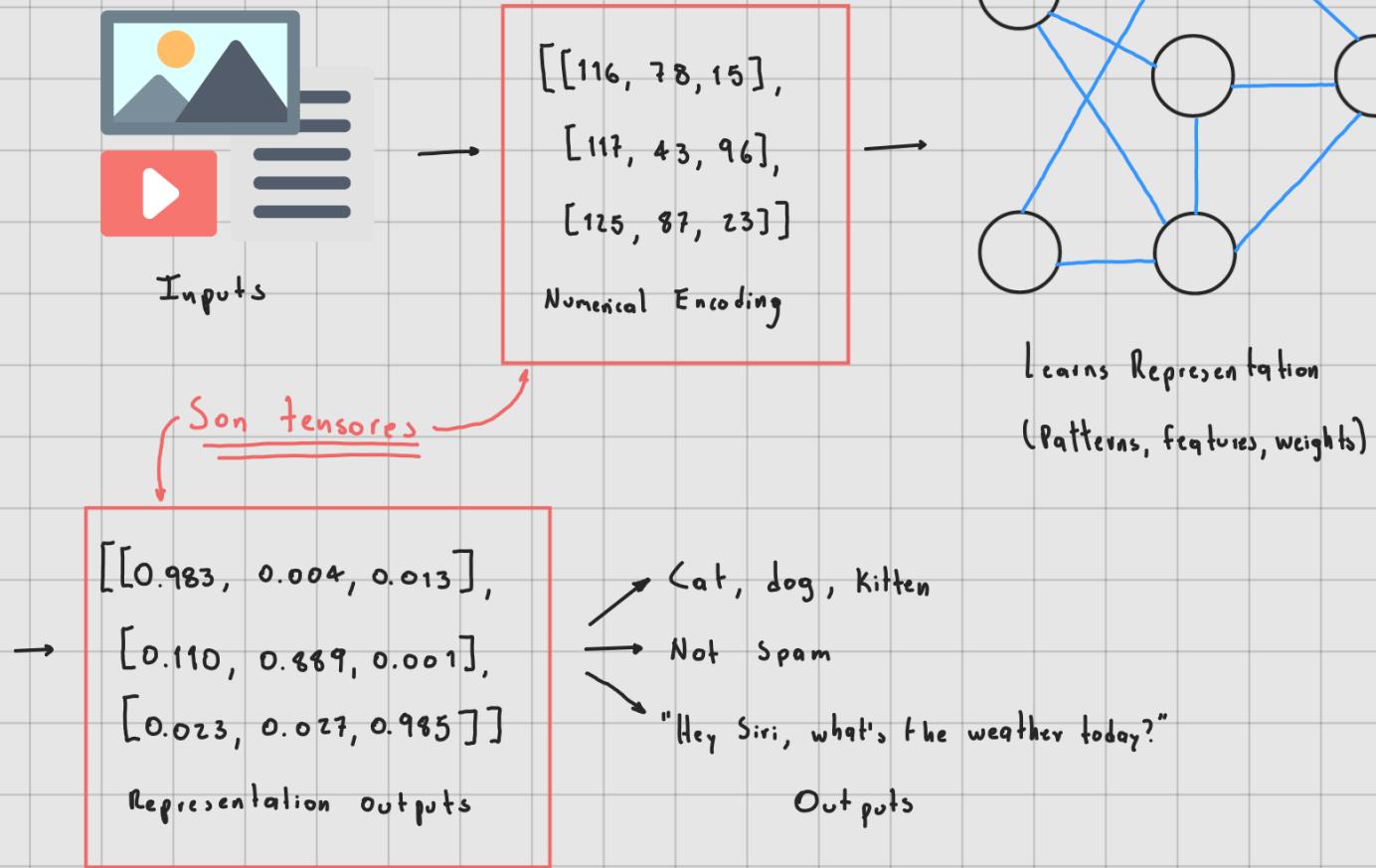
- Reshape y Transpose :

Reorganización de los elementos de un tensor.

Algunas de las bibliotecas populares para manejo de tensores son:

- Tensorflow
- PyTorch
- Numpy
- Keras
- Theano

Regresando a la anatomía de una red neuronal:



Bibliotecas como PyTorch o Tensorflow están basadas en tensores. A través de data no estructurada se codifica numéricamente, convirtiéndola en un tensor de algún tipo el cual es alimentado a la red neuronal; realizando distintas operaciones matemáticas a ese tensor. Después nuestra red neuronal regresa un tensor como output similar al input pero que ha sido manipulado de cierta forma, dependiendo del caso. Finalmente se transforma en data que una persona pueda entender e interpretar.

¿Qué temas abarcaría el curso?

- Fundamentos de PyTorch (Tensores y sus operaciones)
- Preprocesamiento de Data (Transformación tensorial)
- Construcción y uso de modelos pre-entrenados
- Tallaaje del modelo a la data (Aprendizaje de patrones)
- Realizar predicciones con el modelo (Uso de patrones)
- Evaluación del modelo
- Guardado y carga del modelo
- Uso de un modelo entrenado para hacer predicciones en datos personalizada

Filosofía del Curso

"ML/DL se encuentra entre la ciencia y el arte".



Precision



t

Libertad Creativa

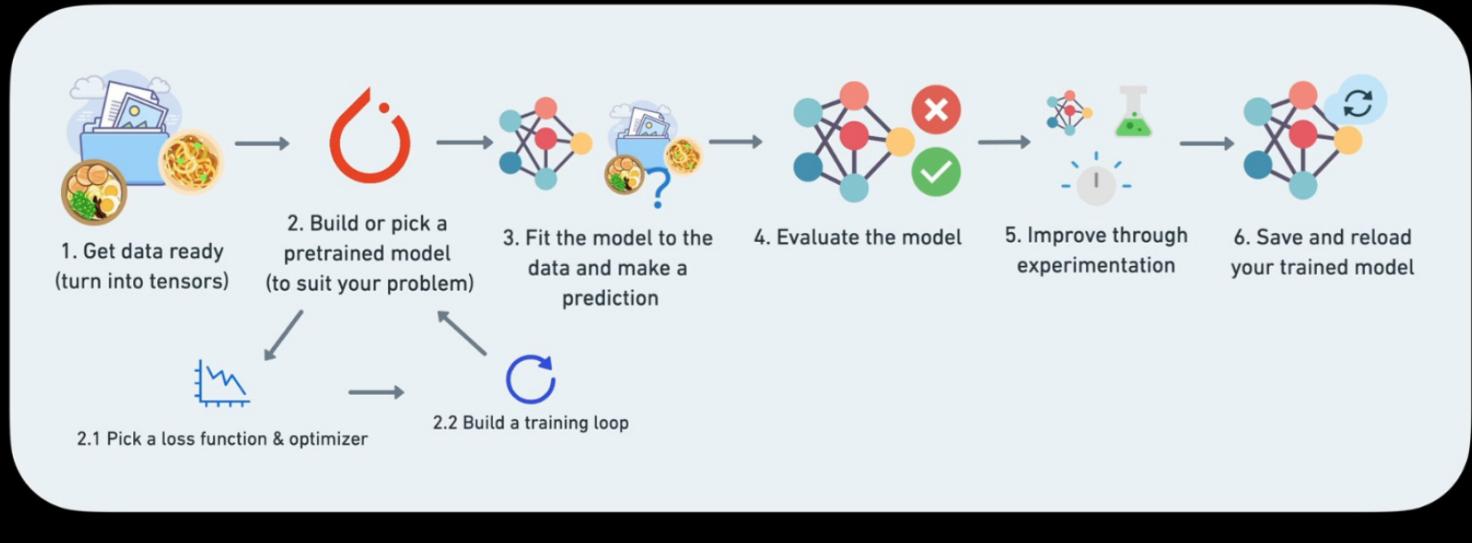
stablediffusionweb.com

PyTorch Workflow

What we're going to cover

A PyTorch workflow

(one of many)



Udemy

No es un proceso con un orden estricto, es un flujo.