Artificial Intelligence

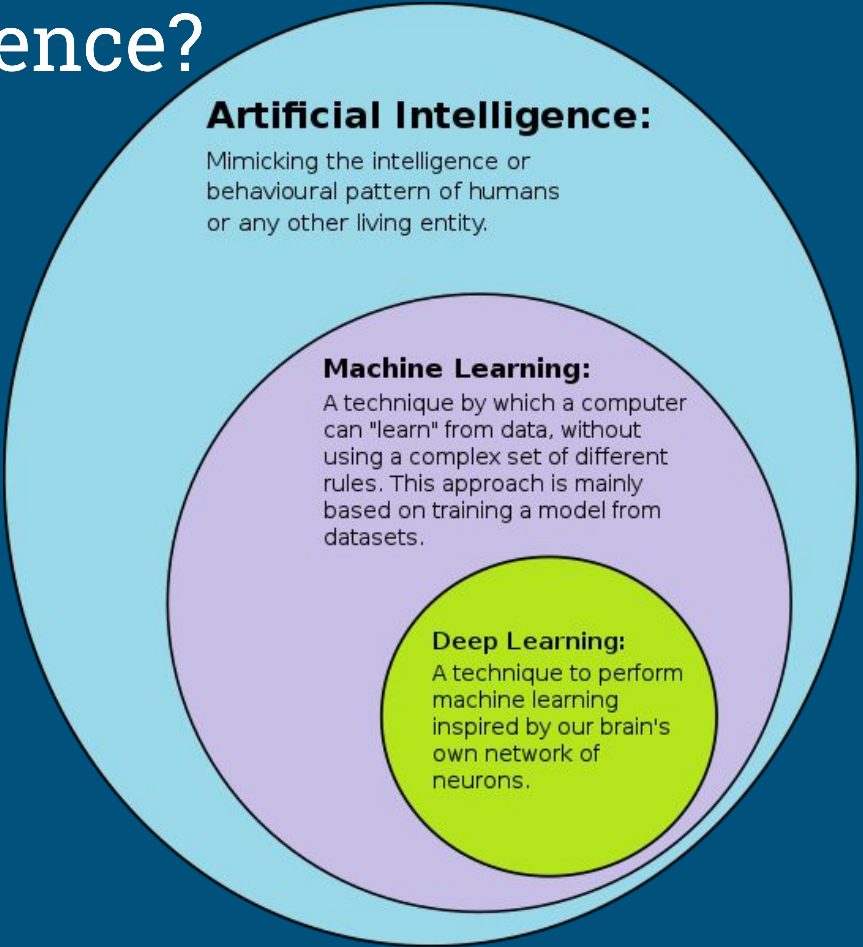
Key notes

What is Artificial Intelligence?

AI ➡ Any technique that enables computers to mimic human intelligence

ML➡ A subset of AI that includes abstruse statistical techniques that enable machines to improve at tasks with experience.

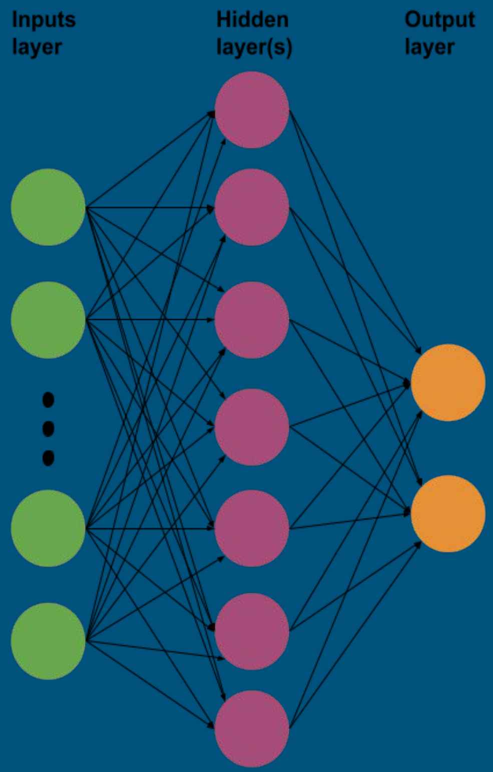
DL➡The subset of ML composed of algorithms that permit software to train itself to perform tasks by exposing multilayered neural networks to vast amount of data,


Artificial Intelligence:
Mimicking the intelligence or behavioural pattern of humans or any other living entity.
Machine Learning:
A technique by which a computer can "learn" from data, without using a complex set of different rules. This approach is mainly based on training a model from datasets.
Deep Learning:
A technique to perform machine learning inspired by our brain's own network of neurons.

ANN and DNN

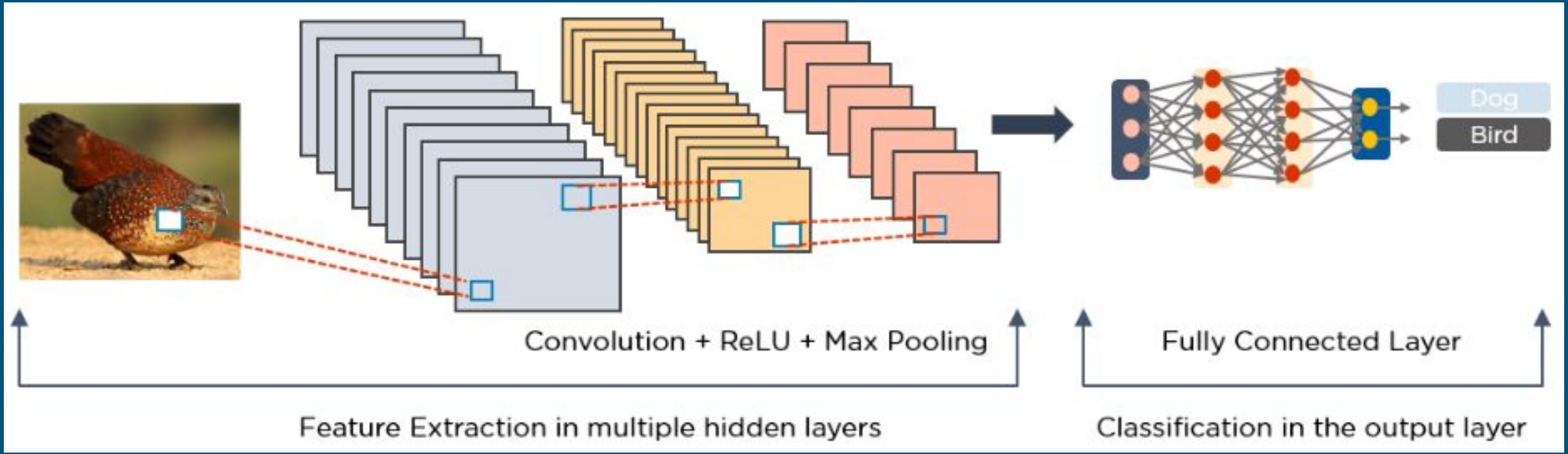
ANN → or else NN,are computing systems inspired by the biological neural networks that constitute animal brain. An ANN is based on a collection of connected units or notes (artificial neurons).

ANN has just one hidden layer, while **deep neural networks** are consider those with more than one hidden layers



Convolutional Neural Networks (CNNs)

CNN → consist of multiple layers that process and extract features from data. It is a specific architecture of neural Networks that are extremely effective at dealing with image data.


Dog
Bird
Convolution + ReLU + Max Pooling
Fully Connected Layer
Feature Extraction in multiple hidden layers
Classification in the output layer

Convolutional Neural Networks (CNNs)

Convolution Layer → has several filters to perform the convolution operation.

Rectified Linear Unit (ReLU) → performs operations on elements, the output is a rectified feature map.

Pooling Layer → is a down-sampling operation that reduces the dimensions of the feature map. The pooling layer then converts the resulting 2D arrays from the pooled feature map into a single, long, continuous linear vector by flattening it.

Fully connected Layer → forms when the flattened matrix from the pooling layer is fed as an input, which classifies and identifies the images.

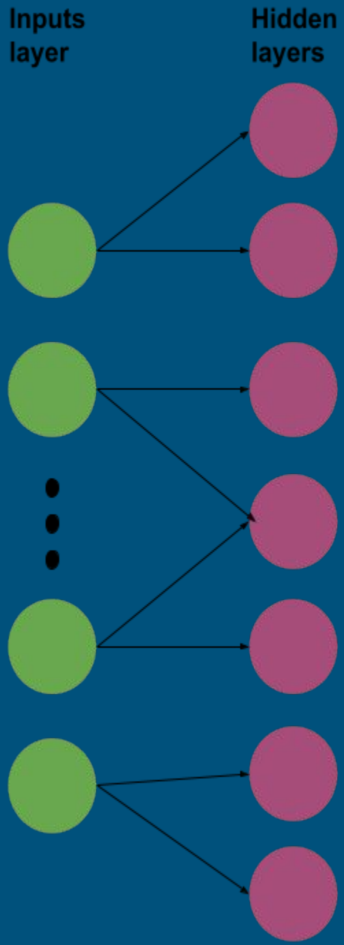
Convolutional layer

A convolutional layer is created when we apply multiple image filters to the input images.

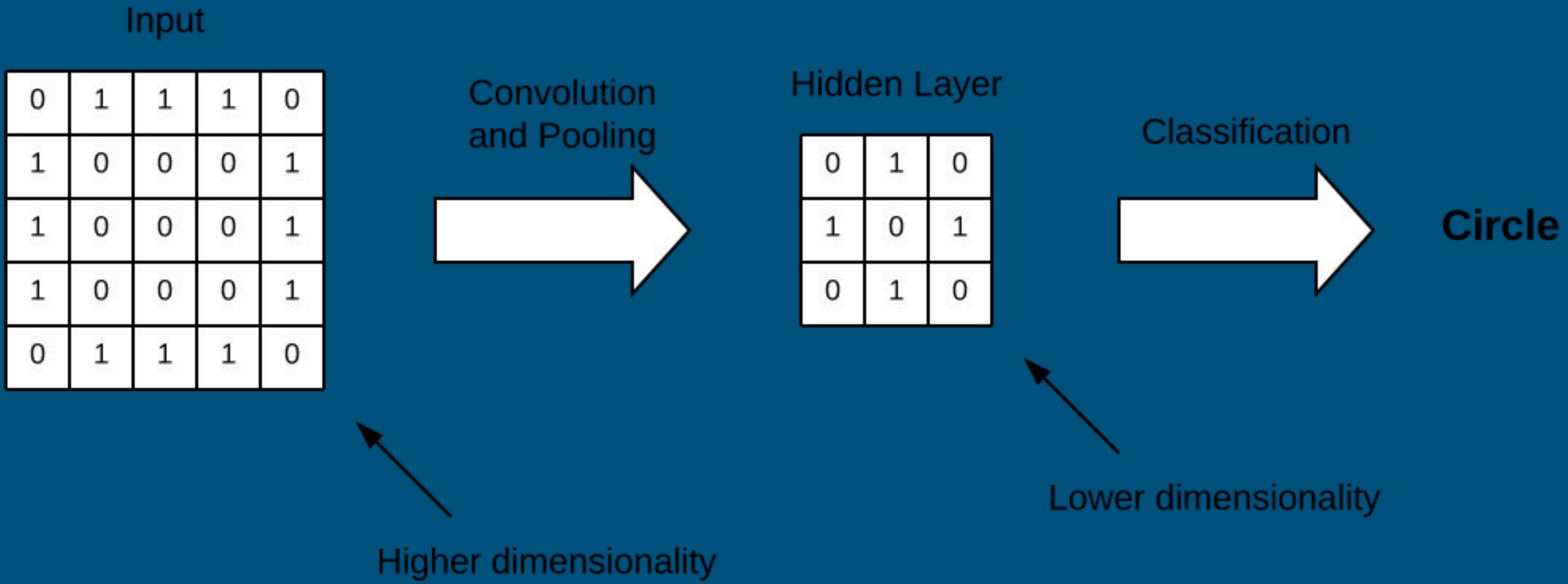
The layer will then be trained to figure out the best filter weight values.

A CNN helps reduce parameters by focusing on local connectivity.

Not all neurons will be fully connected. Instead, neurons are only connected to a subset of local neurons in the next layer (these end up being the filters)

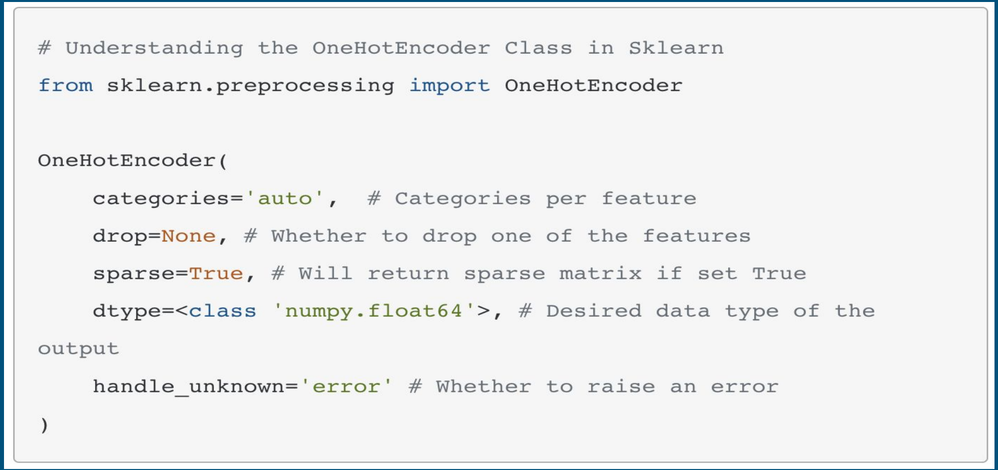


Convolutional Neural Network



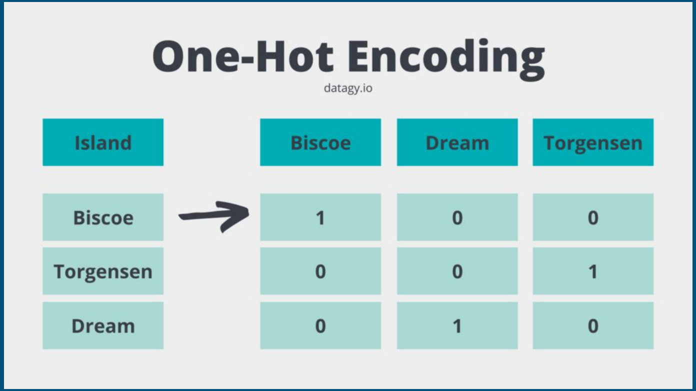
One-hot encoding

One-hot encoding is the process by which **categorical data are converted into numerical data** for use in ML. Categorical features are turned into binary features that are "one-hot" encoded, meaning that if a feature is represented by that columns it receives a 1 otherwise it receives a 0.



```
# Understanding the OneHotEncoder Class in Sklearn
from sklearn.preprocessing import OneHotEncoder

OneHotEncoder(
    categories='auto',  # Categories per feature
    drop=None, # Whether to drop one of the features
    sparse=True, # Will return sparse matrix if set True
    dtype=<class 'numpy.float64'>, # Desired data type of the
output
    handle_unknown='error' # Whether to raise an error
)
```

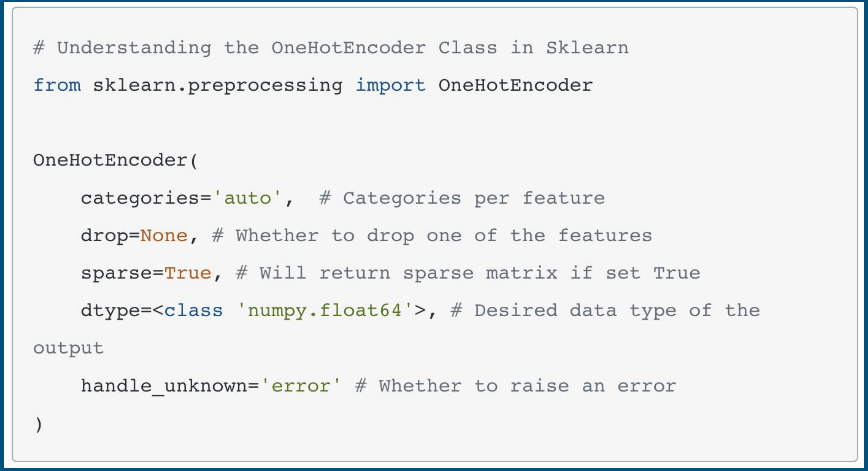


One-Hot Encoding

datagy.io

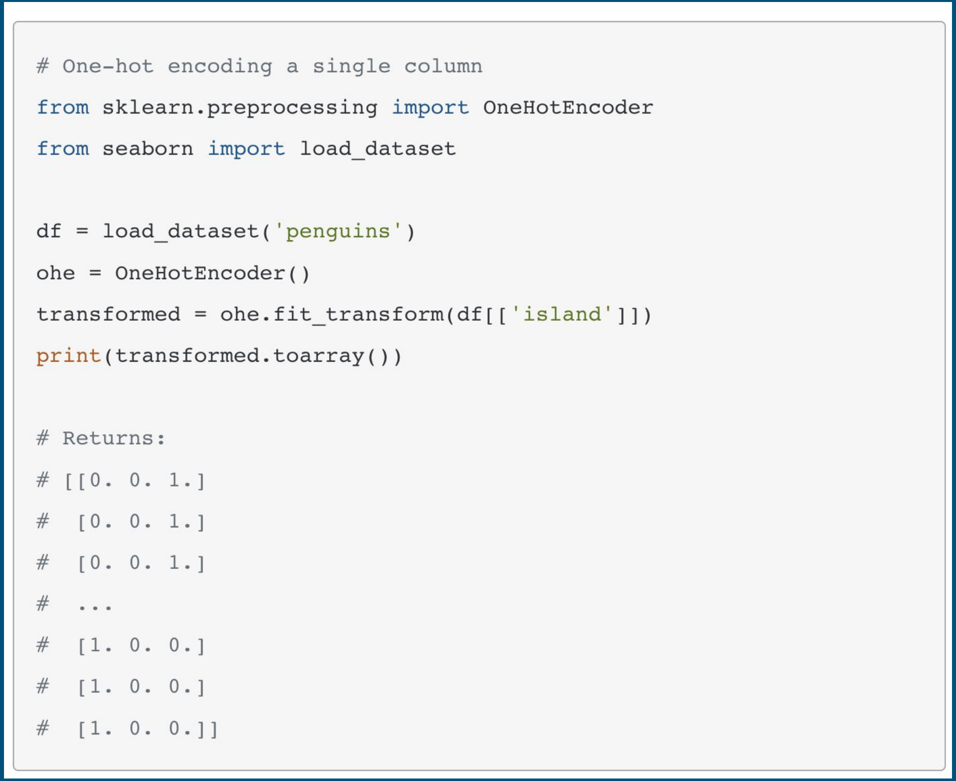
Island		Biscoe	Dream	Torgensen
Biscoe	→	1	0	0
Torgensen		0	0	1
Dream		0	1	0

One-hot encoding



```python
# Understanding the OneHotEncoder Class in Sklearn
from sklearn.preprocessing import OneHotEncoder

OneHotEncoder(
    categories='auto',  # Categories per feature
    drop=None, # Whether to drop one of the features
    sparse=True, # Will return sparse matrix if set True
    dtype=<class 'numpy.float64'>, # Desired data type of the
output
    handle_unknown='error' # Whether to raise an error
)
```



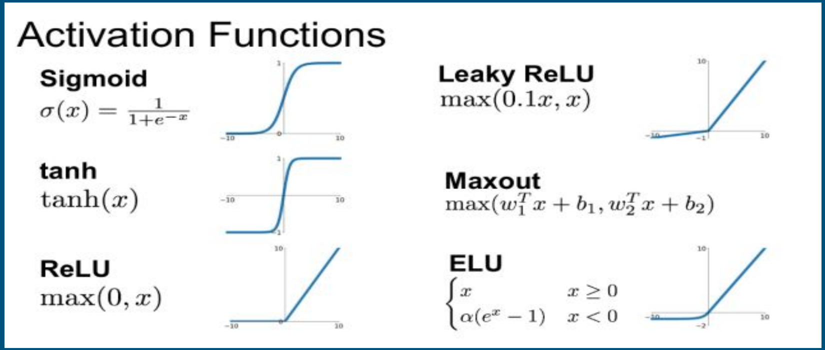
```python
# One-hot encoding a single column
from sklearn.preprocessing import OneHotEncoder
from seaborn import load_dataset

df = load_dataset('penguins')
ohe = OneHotEncoder()
transformed = ohe.fit_transform(df[['island']])
print(transformed.toarray())

# Returns:
# [[0. 0. 1.]
#  [0. 0. 1.]
#  [0. 0. 1.]
#  ...
#  [1. 0. 0.]
#  [1. 0. 0.]
#  [1. 0. 0.]]
```

More details here: https://datagy.io/sklearn-one-hot-encode/

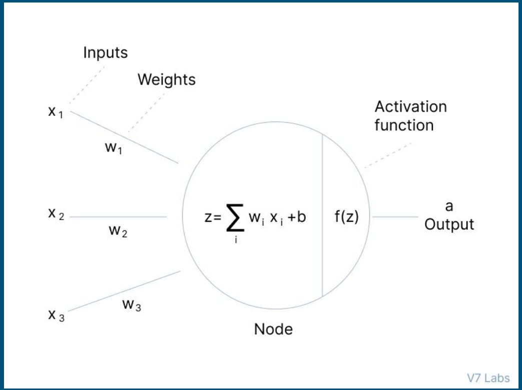
Activation Functions



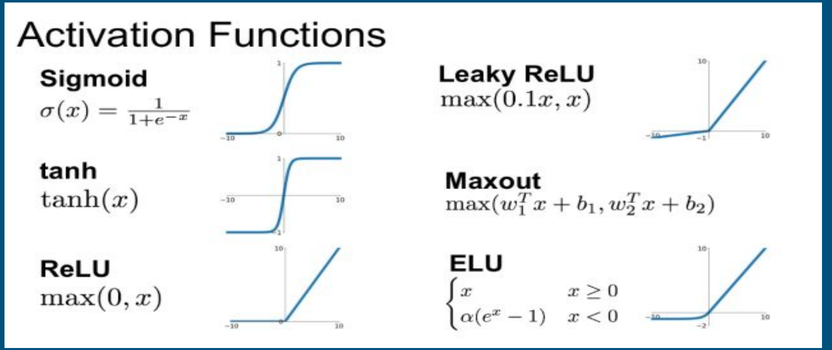
- Activation function **decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it**. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

- In a NN the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.

- Is piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.



Activation Functions



- Technically, the activation function is used within or after the internal processing of each node in the network, although networks are designed to use the same activation function for all nodes in a layer.
- A network may have three types of layers: input layers that take raw input from the domain, **hidden layers** that take input from another layer and pass output to another layer, and **output layers** that make a prediction.
- All hidden layers typically use the same activation function. The output layer will typically use a different activation function from the hidden layers and is dependent upon the type of prediction required by the model.

Activation for Hidden Layers

There are perhaps three activation functions you may want to consider for use in hidden layers; they are:

- Rectified Linear Activation (ReLU)
- Logistic (Sigmoid)
- Hyperbolic Tangent (Tanh)

ReLU – Rectified Linear Unit

- It is common because it is both simple to implement and effective at overcoming the limitations of other previously popular activation functions, such as Sigmoid and Tanh. Specifically, it is less susceptible to vanishing gradients that prevent deep models from being trained, although it can suffer from other problems like saturated or “*dead*” units.

Advantages of ReLU:

1. Computational Simplicity
2. Linear behavior
3. Train deep networks

Tensorflow, Keras and PyTorch

TensorFlow is an end-to-end open source deep learning framework developed by Google.

Keras is an effective high-level NN application programming interface (api).

Keras acts as an interface for the tensorflow library

PyTorch is a low-level API developed by Facebook for natural language processing and computer vision. It is more powerful version of numpy.

Tensorflow, Keras and PyTorch

- High and low level API
- Very fast
- Complex architecture and is hard to use
- Big datasets, hard debugging
- Hard to develop and write code
- Easy to deploy

- High level API
- Very slow (works on top of TF)
- Simpler architecture, simple to use
- Smaller datasets, easy debugging
- Easy to develop and best for newbies
- Deploy as tf or Flask

- low level API
- Very fast
- Complex architecture
- Big datasets, easier than tf to debug
- Easier to learn than TF
- Easy to deploy but not as tf