Introduction To Deep Learning

Workshop 1

NetBrain.ml

Last Edit: 11/08/19

What is Deep Learning?

Machine Learning (ML) is a branch of artificial intelligence which focuses on **developing the ability for machines to learn**.

Deep learning is the subfield of ML which applies ML algorithms in multiple layers, to **promote more end-to-end development.**

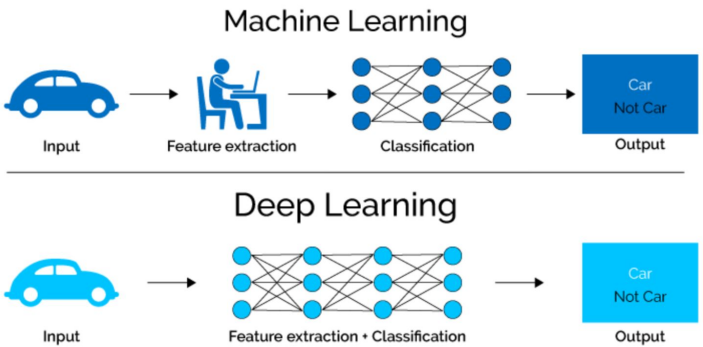
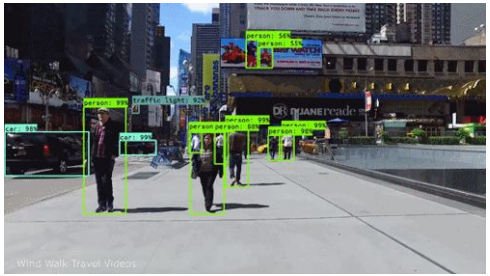
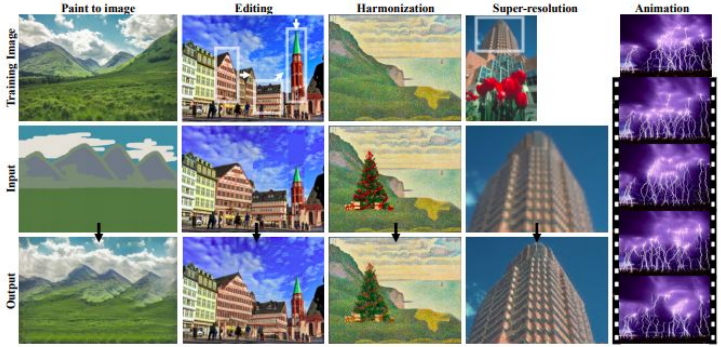


Figure 1: Machine Learning VS Deep Learning

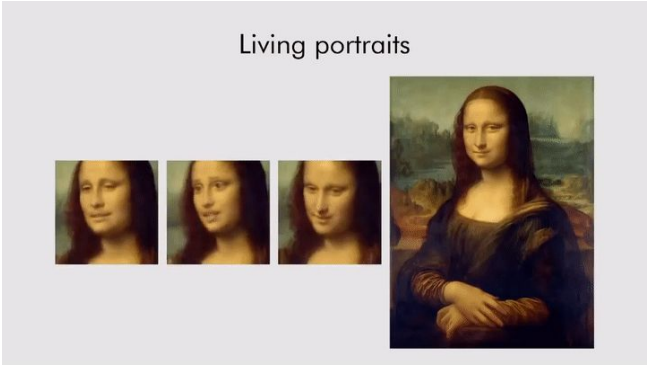
Why you should learn deep learning

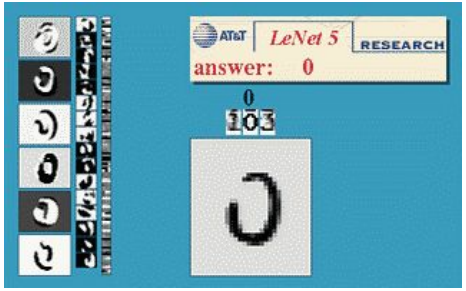


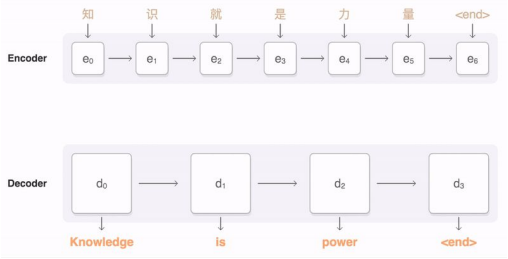


Dog: 96%
Cat: 29%
Duck: 2%
Bird: 0%

Dog: 36%
Cat: 94%
Duck: 2%
Bird: 1%







Logistics

We will program in Python

Learn how to use Tensorflow, PyTorch, and other DL dependencies

Use Kaggle or Google Collab for demos

At the end, we will host a mini-challenge, where we will provide a dataset.

Link: https://netbrainml.github.io/workshops

Workshop Schedule

This workshop series will focus on various topics in the field of artificial intelligence from introductory concepts to more state of the art applications.

Workshop 1: Fundamentals of Deep Learning
Topics: Basic concepts, Linear/Logistic Regression, Standard Feedforward Networks

Workshop 2: Convolutional and Recurrent Models
Topics: Convolutional/Recurrent Networks, Gatings, Embeddings

Workshop 3: Mechanisms for Optimal Performance
Topics: Optimizers, Blocking, Initialization, Normalization, Skip Connections, Transfer Learning, and more

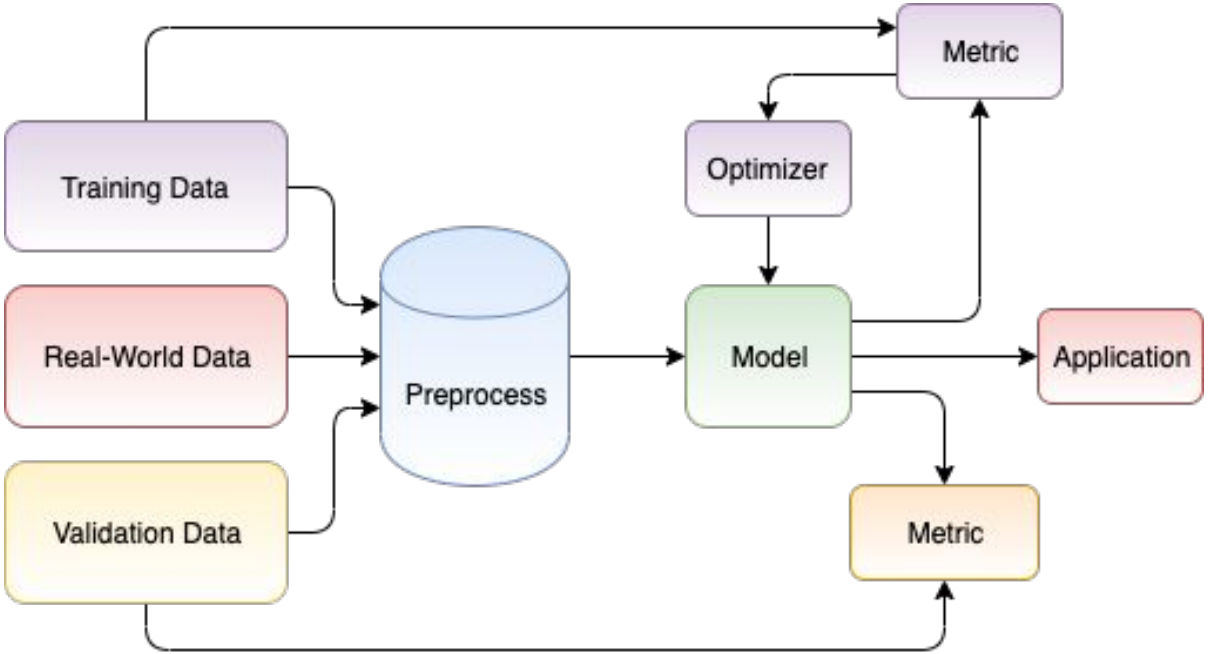
Generalization of Deep Learning

Data	Model	Loss/Optimizer
Representation of the task (ex. Images of cats and dogs)	The algorithm itself that performs the task with its parameters (ex. Linear Regression)	Calculates how poorly the model does and modifies the parameters to better handle task (ex. MSE, Gradient Descent)

General Steps of Deep Learning

1) Collect data
   Depending on the application, we obtain the relevant data for learning
2) Preprocess data
   We format the data so that our model can perform well
3) Build model
   Based on the requirements, we design and implement a model architecture
4) Train model
   The model learns how to produce correct output from the data
5) Validate the model
   To validate the model's performance, the model is tested with unseen data

Visualization of General Steps



What is in our dataset?

Features: The part of the dataset that describes your data

Labels: The part of the dataset you want to predict

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Linear / Logistic Regression

Goal: Find the best fit line on the dataset using the equation

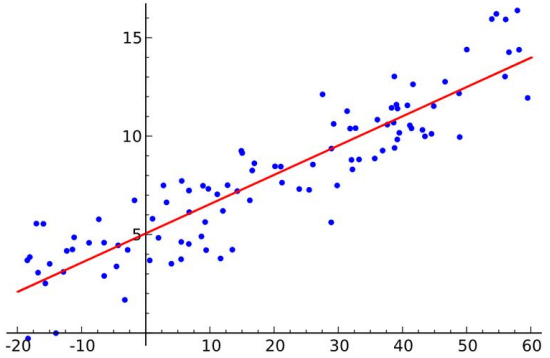
Linear Regression: $Y = mX+b$

Logistic regression builds on linear regression by applying nonlinear function:

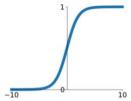
Logistic Regression: $Y = g(mX + b)$

We are finding what the correct slope and y-intercept is (known as weight and bias)

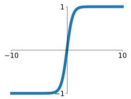
Weight (m) and bias (b) are our *learnable parameters*



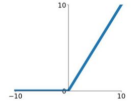
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



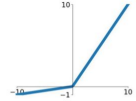
tanh
$\tanh(x)$



ReLU
$\max(0, x)$

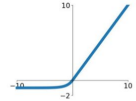


Leaky ReLU
$\max(0.1x, x)$



Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Single Layer Perceptron

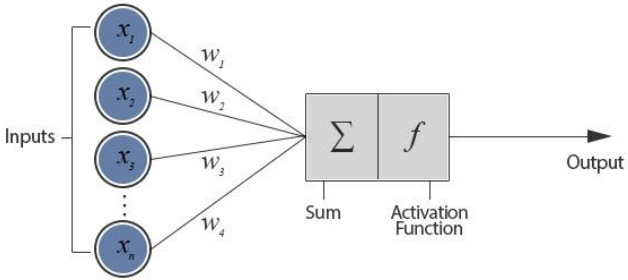
Goal: Having ***multi-dimensional*** linear/logistic regressions to handle more complex tasks

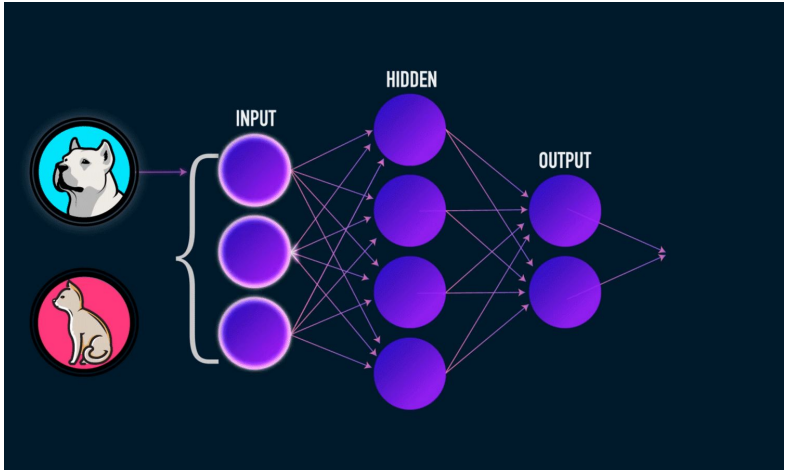
So, now we have more than one value/vector to represent each of our parameters (a matrix)

Neuron: Each regression operator node

Perceptron: A type of feed-forward neural network used for classification

Note: Universal Approximation Theorem states that any arbitrary function can be approximated by these networks. So any application can be handled by this model given enough nodes.





Multi-Layer Perceptron (MLP)

Goal**:** Stacking ***multiple single layer*** models to handle more complex tasks

So, now we have more than one layer to represent our model (a cascade of matrices)

This is the standard feedforward network typically used as a baseline.

Note: Adding more layers increases training time and does not always improve performance (*)