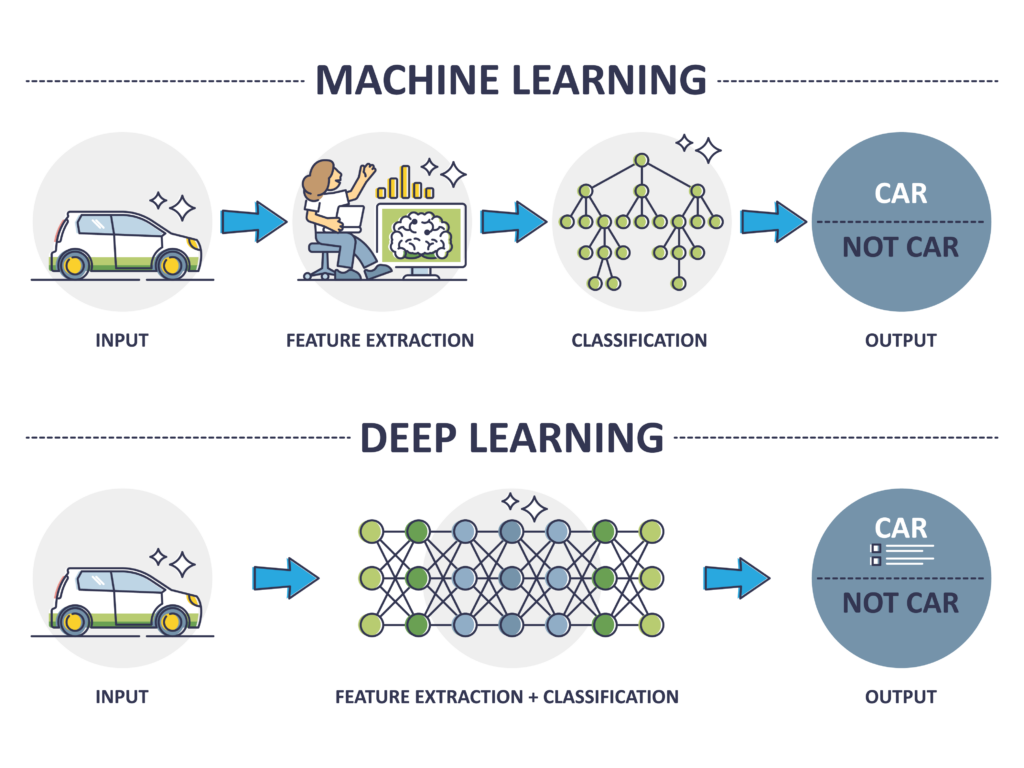
**DEEP LEARNING**

**Introduction to deep learning:-**

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don’t need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs).



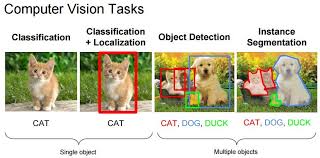
**Computer vision:-**

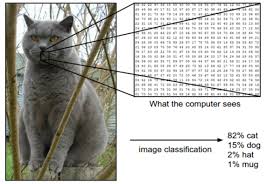
Computer Vision, often abbreviated as CV, is defined as a field of study that seeks to develop techniques to help computers “see” and understand the content of digital images such as photographs and videos.

Different types of computer vision include image segmentation, object detection, facial recognition, edge detection, pattern detection, image classification, and feature matching.

Examples:-

* Facial recognition.
* Self-driving cars.
* Robotic automation.
* Medical anomaly detection.
* Sports performance analysis.
* Manufacturing fault detection.
* Agricultural monitoring.
* Plant species classification.



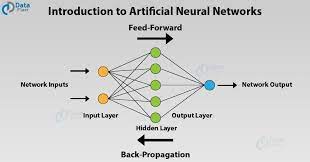


**Artificial neural network:-**

Artificial Neural Networks (ANN) are algorithms based on brain function and are used to model complicated patterns and forecast issues. The Artificial Neural Network (ANN) is a deep learning method that arose from the concept of the human brain Biological Neural Networks. The development of ANN was the result of an attempt to replicate the workings of the human brain. The workings of ANN are extremely similar to those of biological neural networks, although they are not identical. ANN algorithm accepts only numeric and structured data.

Example:-

Artificial neural networks are trained using a training set. For example, suppose you want to teach an ANN to recognize a cat. Then it is shown thousands of different images of cats so that the network can learn to identify a cat.



**Neurons & Perceptron:-**

Perceptron is a type of artificial neural network, which is a fundamental concept in machine learning. The basic components of a perceptron are:

Input layer.

Weights.

Bias.

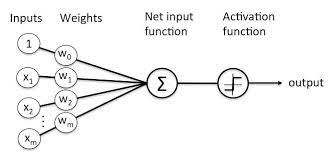
Activation function.

Output.

Training algorithm.

## Types of Perceptron:

1. Single layer: Single layer perceptron can learn only linearly separable patterns.
2. Multilayer: Multilayer perceptrons can learn about two or more layers having a greater processing power.

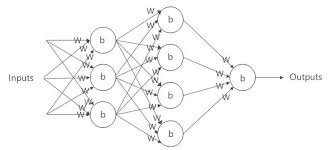


Neurons:-

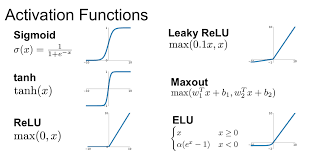
Neurons in deep learning models are nodes through which data and computations flow. Neurons work like this: They receive one or more input signals. These input signals can come from either the raw data set or from neurons positioned at a previous layer of the neural net

Each neuron in a neural network computes an output value by applying a specific function to the input values received from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers).

The three main types of learning in neural networks are supervised learning, unsupervised learning, and reinforcement learning.



**Activation function:-**



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

Ridge activation functions.

Folding activation functions.

Radial activation functions.

**Gradient descent:-**

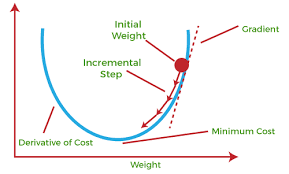
Gradient Descent is known as one of the most commonly used optimization algorithms to train machine learning models by means of minimizing errors between actual and expected results. Further, gradient descent is also used to train Neural Networks.

Types of gradient descent:-

Batch gradient descent

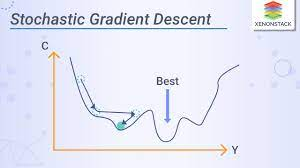
Stochastic gradient descent

Mini-batch gradient descent



**Stochastic gradient descent:-**

Stochastic gradient descent is an optimization algorithm often used in machine learning applications to find the model parameters that correspond to the best fit between predicted and actual outputs. It's an inexact but powerful technique. Stochastic gradient descent is widely used in machine learning applications.

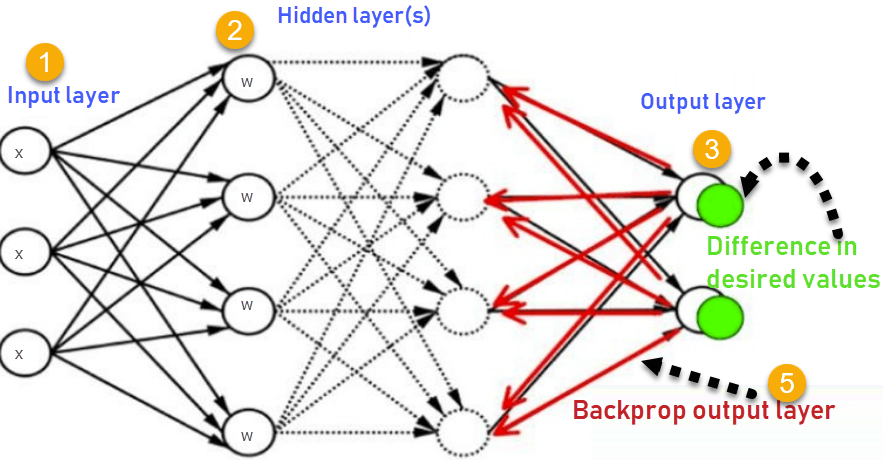


**Backpropagation:-**

Backpropagation, or backward propagation of errors, is an algorithm that is designed to test for errors working back from output nodes to input nodes. It is an important mathematical tool for improving the accuracy of predictions in data mining and machine learning. Essentially, backpropagation is an algorithm used to calculate derivatives quickly.

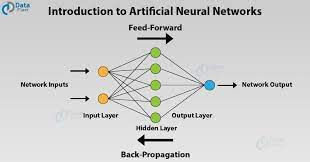
There are two leading types of backpropagation networks:

1. **Static backpropagation.** Static backpropagation is a network developed to map static inputs for static outputs. Static backpropagation networks can solve static classification problems, such as optical character recognition (OCR).
2. **Recurrent backpropagation.** The recurrent backpropagation network is used for fixed-point learning. Recurrent backpropagation activation feeds forward until it reaches a fixed value.

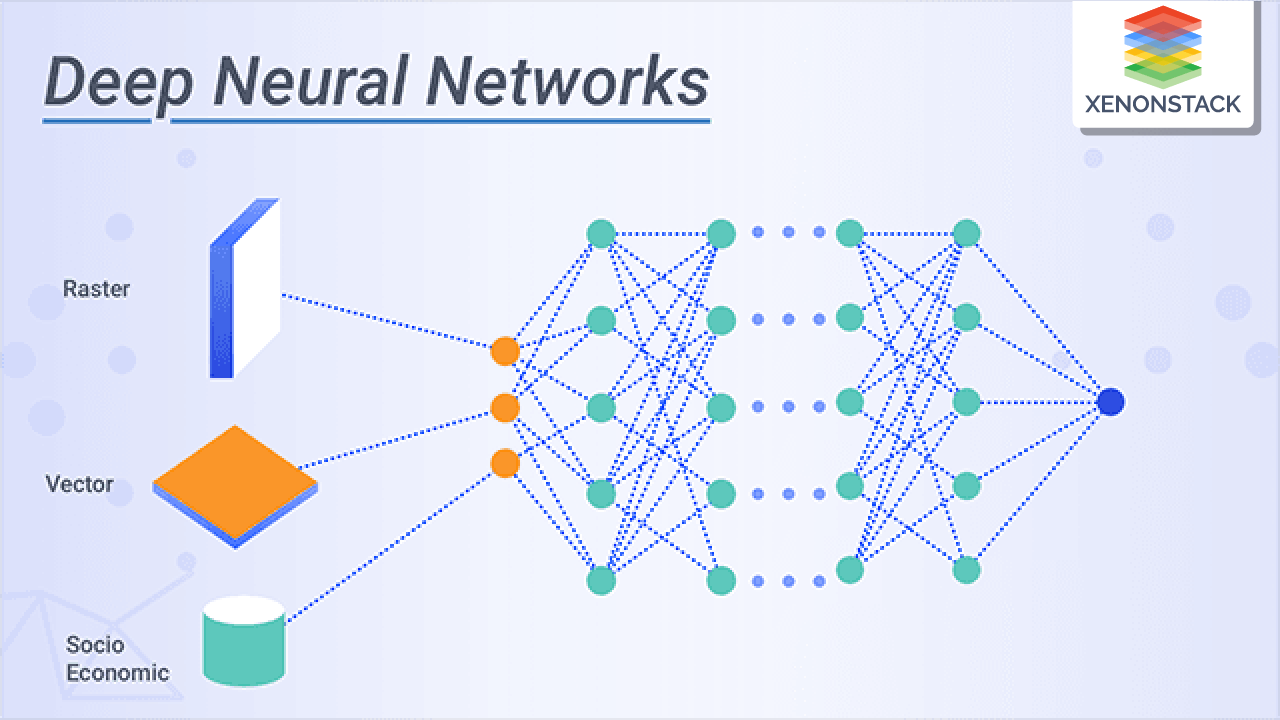


**Artificial neural network:-**

Artificial Neural Networks (ANN) are algorithms based on brain function and are used to model complicated patterns and forecast issues. The Artificial Neural Network (ANN) is a deep learning method that arose from the concept of the human brain Biological Neural Networks. The development of ANN was the result of an attempt to replicate the workings of the human brain. The workings of ANN are extremely similar to those of biological neural networks, although they are not identical. ANN algorithm accepts only numeric and structured data.



**Deep neural network:-**



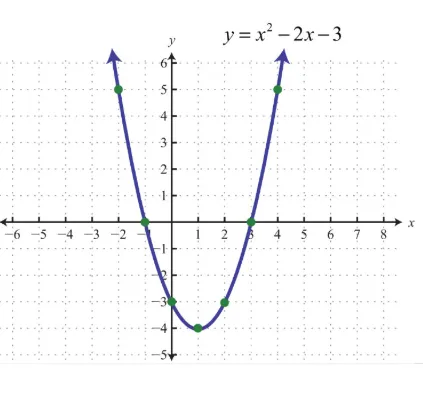
A Deep Neural Network (DNN) is a machine learning technique that allows a computer, by training it, to do tasks that would be very difficult to do using conventional programming techniques. Neural network algorithms were inspired by the human brain and its functions: like our human mind, it is designed to work not only by following a preset list of rules, but by predicting solutions and drawing conclusions based on previous iterations and experiences.

A deep neural network’s process is best understood by looking at an example. Imagine you had hundreds of thousands of images, some of which had dogs in them, and you decided you wanted to write a computer program to recognize dogs in pictures.

Deep neural networks can recognize voice commands, identify voices, recognize sounds and graphics and do much more than a neural network. Deep learning networks utilize "Big Data'' along with algorithms in order to solve a problem, and these deep neural networks can solve problems with limited or no human input.

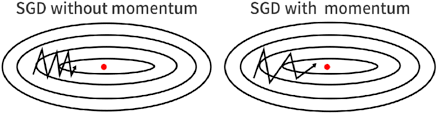
**Optimization algorithms-SGD,momentum,NAG,adagrad,adadelta,RMSprop,adam:-**

Gradient, in plain terms means slope or slant of a surface. So gradient descent literally means descending a slope to reach the lowest point on that surface. Let us imagine a two dimensional graph, such as a parabola in the figure below.



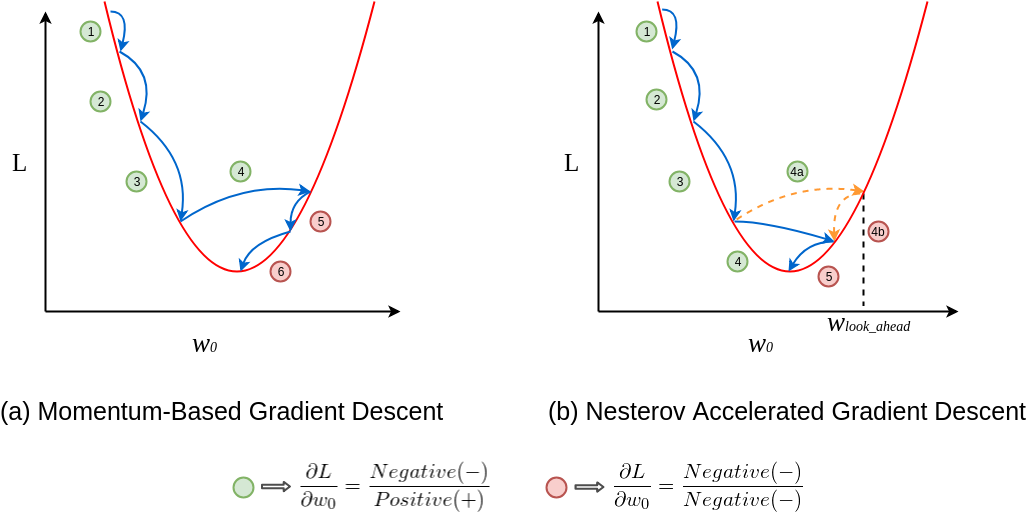
Momentum:-

**Momentum** is an extension to the gradient descent optimization algorithm that allows the search to build inertia in a direction in the search space and overcome the oscillations of noisy gradients and coast across flat spots of the search space.

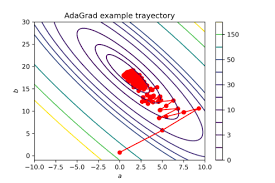


The intuition behind NAG can be summarized as 'look before you leap'. Let us try to understand this through an example. As can see, in the momentum-based gradient, the steps become larger and larger due to the accumulated momentum, and then we overshoot at the 4th step.

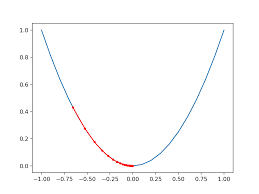
NAG resolves this problem by adding a look ahead term in our equation. The intuition behind NAG can be summarized as ‘look before you leap’. Let us try to understand this through an example.



**Adaptive Gradients**, or **AdaGrad** for short, is an extension of the gradient descent optimization algorithm that allows the step size in each dimension used by the optimization algorithm to be automatically adapted based on the gradients seen for the variable (partial derivatives) seen over the course of the search.



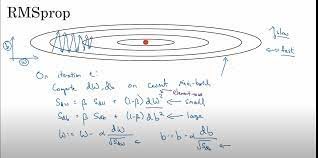
AdaDelta is a stochastic optimization technique that allows for per-dimension learning rate method for SGD. It is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate.



RProp, or we call Resilient Back Propagation, is the widely used algorithm for supervised learning with multi-layered feed-forward networks. The basic concept of the backpropagation learning algorithm is the repeated application of the chain rule to compute the influence of each weight in the network with respect to an arbitrary error.

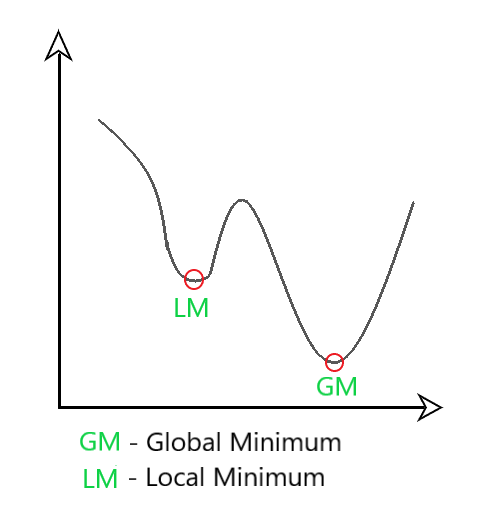
RProp algorithm does not work for mini-batches is because it violates the central idea behind stochastic gradient descent, when we have a small enough learning rate, it averages the gradients over successive mini-batches. To solve this issue, consider the weight, that gets the gradient 0.1 on nine mini-batches, and the gradient of -0.9 on tenths mini-batch, RMSProp did force those gradients to roughly cancel each other out, so that the stay approximately the same when computing.

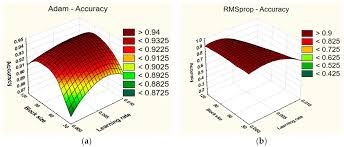
By using the sign of gradient from RProp algorithm, and the mini-batches efficiency, and averaging over mini-batches which allows combining gradients in the right way. RMSProp keep moving average of the squared gradients for each weight. And then we divide the gradient by square root the mean square.



Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data or parameters. It requires less memory and is efficient. Intuitively, it is a combination of the ‘gradient descent with momentum’ algorithm and the ‘RMSP’ algorithm.

Adam Optimizer inherits the strengths or the positive attributes of the above two methods and builds upon them to give a more optimized gradient descent.





**Batch normalization:-**

Batch normalization is a deep learning approach that has been shown to significantly improve the efficiency and reliability of neural network models. It is particularly useful for training very deep networks, as it can help to reduce the internal covariate shift that can occur during training.

Batch normalization is a supervised learning method for normalizing the interlayer outputs of a neural network. As a result, the next layer receives a “reset” of the output distribution from the preceding layer, allowing it to analyze the data more effectively.

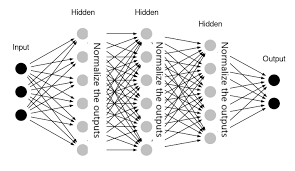
Advantages:-

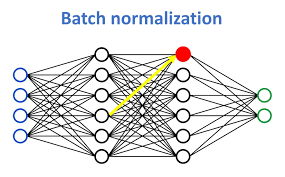
Stabilize the training process.

Improves generalization.

Reduces the need of carful initialization.

Allows for higher learning rates.



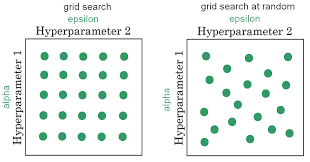


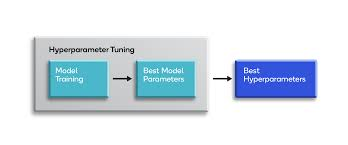
**Hyperparameter tuning:-**

Hyperparameter tuning is an essential part of controlling the behavior of a machine learning model. If we don’t correctly tune our hyperparameters, our estimated model parameters produce suboptimal results, as they don’t minimize the loss function. This means our model makes more errors. In practice, key indicators like the accuracy or the confusion matrix will be worse.

In machine learning, we need to differentiate between parameters and hyperparameters. A learning algorithm learns or estimates model parameters for the given data set, then continues updating these values as it continues to learn. After learning is complete, these parameters become part of the model. For example, each weight and bias in a neural network is a parameter.

Hyperparameters, on the other hand, are specific to the algorithm itself, so we can’t calculate their values from the data. We use hyperparameters to calculate the model parameters. Different hyperparameter values produce different model parameter values for a given data set.

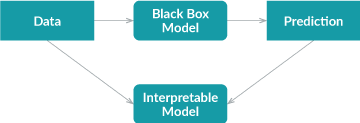


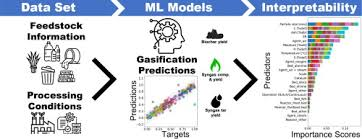


**Interpretability:-**

Model Interpretability of Deep Neural Networks (DNN) has always been a limiting factor for use cases requiring explanations of the features involved in modelling and such is the case for many industries such as Financial Services. Financial institution whether by regulation or by choice prefer structural models that are easy to interpret by humans that’s why deep learning models within these industries have had slow adoptions. An example of a critical use case would be risk models where usually banks prefer classic statistical methods such as Generalized Linear Models, Bayesian Models and traditional machine learning models such as Tree-based models that are easily explainable and interpret in terms of human intuition.

Interpretability since the beginning has been an important area of research since Deep Learning models can achieve high accuracy but at the expense of high abstraction (i.e. *accuracy vs interpretability problem*). This is important also because of *Trust* since a model that is not trusted is a model that will not be used (i.e. try selling to upper management a black box model).

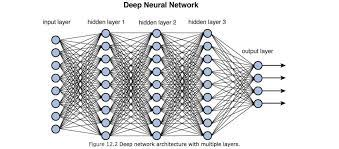


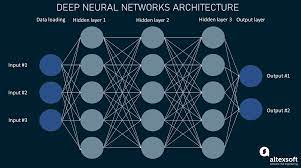


**Deep neural network:-**

At its simplest, a neural network with some level of complexity, usually at least two layers, qualifies as a deep neural network (DNN), or deep net for short. Deep nets process data in complex ways by employing sophisticated math modeling.

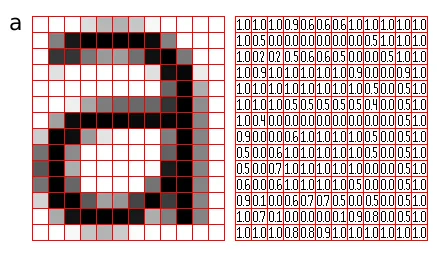
A Deep Neural Network (DNN) is a machine learning technique that allows a computer, by training it, to do tasks that would be very difficult to do using conventional programming techniques. Neural network algorithms were inspired by the human brain and its functions: like our human mind, it is designed to work not only by following a preset list of rules, but by predicting solutions and drawing conclusions based on previous iterations and experiences.

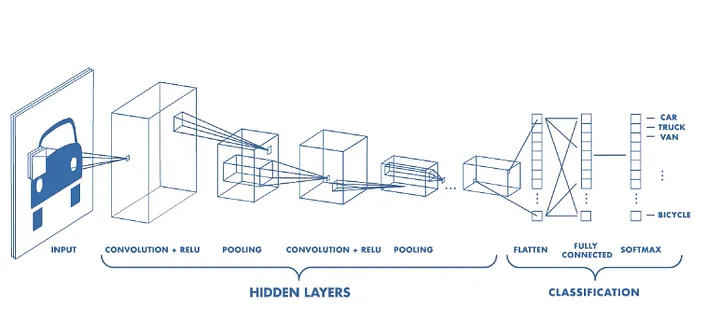




**Convolutional neural network & its layers:-**

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution.





4 layers of CNN

1.Convolutional layer.

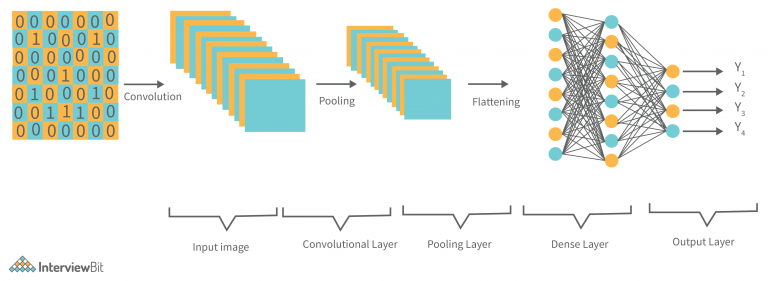
2.Pooling layer.

3.ReLU correction layer.

4.Fully connected layer.

**CNN architecture:-**

The ConvNet’s job is to compress the images into a format that is easier to process while preserving elements that are important for obtaining a decent prediction. This is critical for designing an architecture that is capable of learning features while also being scalable to large datasets.



Convolutional Layer (CONV): They are the foundation of CNN, and they are in charge of executing convolution operations. The Kernel/Filter is the component in this layer that performs the convolution operation (matrix).

Pooling Layer (POOL): This layer is in charge of reducing dimensionality. It aids in reducing the amount of computing power required to process the data. Pooling can be divided into two types: maximum pooling and average pooling.

Fully Connected Layer (FC): The fully connected layer (FC) works with a flattened input, which means that each input is coupled to every neuron. After that, the flattened vector is sent via a few additional FC layers, where the mathematical functional operations are normally performed.

Activation Function: The last fully connected layer’s activation function is frequently distinct from the others. Each activity necessitates the selection of an appropriate activation function.

Dropout Layers: The Dropout layer is a mask that nullifies some neurons’ contributions to the following layer while leaving all others unchanged. A Dropout layer can be applied to the input vector, nullifying some of its properties; however, it can also be applied to a hidden layer, nullifying some hidden neurons. Dropout layers are critical in CNN training because they prevent the training data from overfitting.

Different frameworks on deep learning(tensorflow, keras, pytorch & caffe):-

**Tensorflow:-**

In the context of machine learning, *tensor* refers to the multidimensional array used in the mathematical models that describe neural networks. In other words, a tensor is usually a higher-dimension generalization of a matrix or a vector.

Benefits:-

Eager execution.

Computational graph model.

Simple to user API.

Flexible architecture.

Distributed processing.

Performance.

**Keras:-**

Keras is a Python-based deep learning library that is different from other deep learning frameworks. Keras functions as a high-level API specification for neural networks. It can serve both as a user interface and to extend the capabilities of other deep learning framework back ends that it runs on.

Keras is an open source Python package released under the Massachusetts Institute of Technology (MIT) license, with François Chollet, Google, Microsoft, and other contributors holding some of the software's copyrights.

Benefits:-

Better user experience(UX) for deep learning applications.

Seamless python integration.

Large,portable body of work and strong knowledge base.

**Pytorch:-**

PyTorch is an open source Python package released under the modified Berkeley Software Distribution license. Facebook, the Idiap Research Institute, New York University (NYU), and NEC Labs America hold the copyrights for PyTorch. Although Python is the language of choice for data science, PyTorch is a relative newcomer to the deep learning arena.

This section provides an overview of the PyTorch system and gives more details about the following topics:

* Background of PyTorch
* Benefits of using PyTorch
* Typical PyTorch applications

Benefits:-

Dynamic computational graphs.

Lean back end.

Imperative programming style.

Highly extensible.

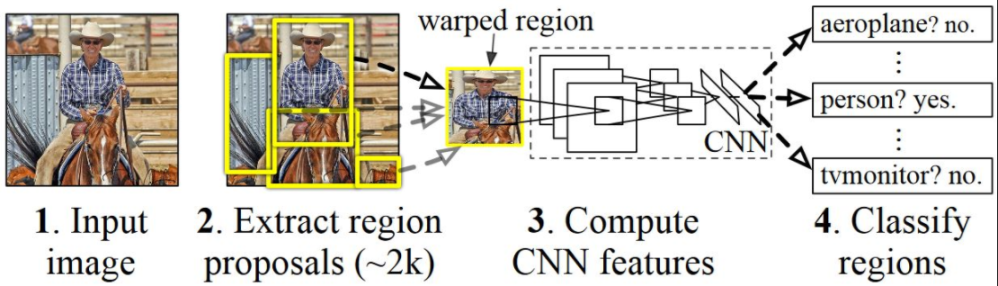
**Caffe:-**

Caffe has been used for image classification and other vision applications, and it supports GPU-based acceleration with the NVIDIA CUDA Deep Neural Network library. Caffe supports Open Multi-Processing (OpenMP) for parallelizing deep learning algorithms over a cluster of systems. Caffe and Caffe2 are written in C++ for performance and offer a Python and MATLAB interface for deep learning training and execution.

**Object recognition using pre trained model-caffe:-**

First of all, we are going to use a pretrained model that was trained using Cafee. Caffe is a Deep Learning Framework created by Facebook which allows us to create Deep Learning models. The model that was used is a Convolutional Neural Network.





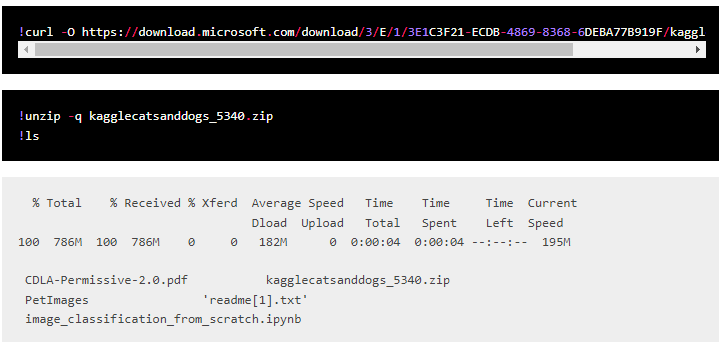
**Image classification using convolutional neural network from scratch-tensorflow & keras:-**

## **Setup**

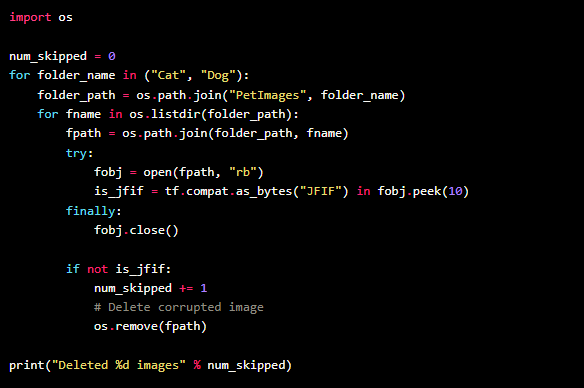
## 

## 

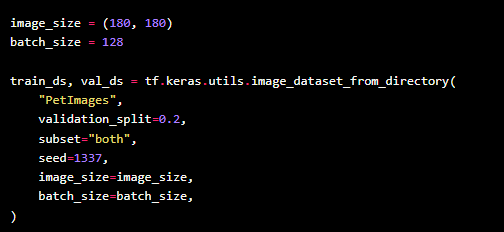
Load the data: the cats vs dogs dataset



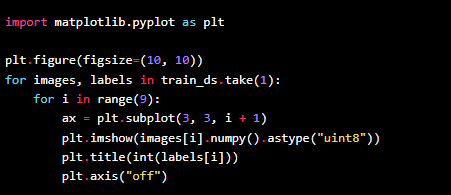
### **Filter out corrupted images**



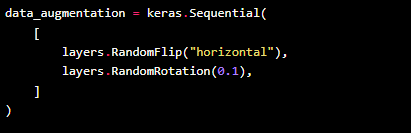
Generate a dataset

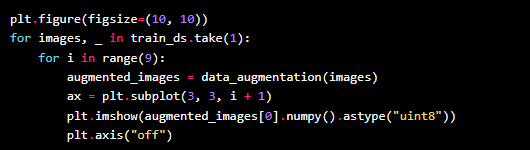


## **Visualize the data**



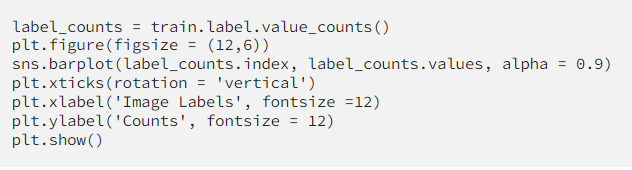
## **Using image data augmentation**



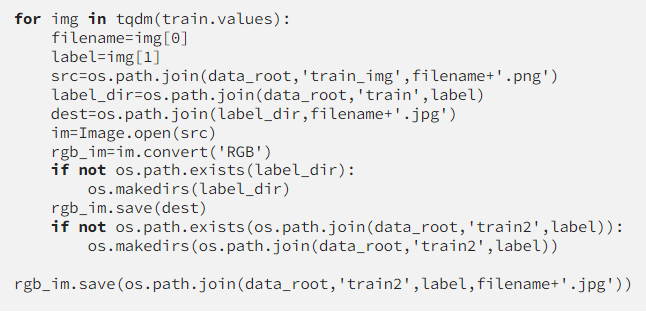


**Custom image classification using transfer learning:-**

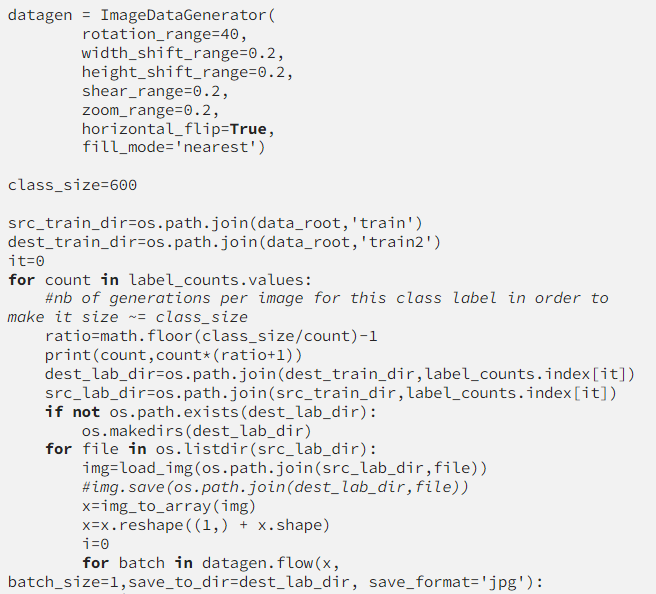
**Step 1: Preprocessing images**

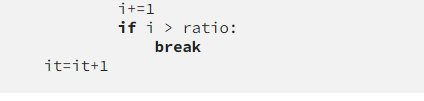


Preprocessing purpose:-



Training images:-

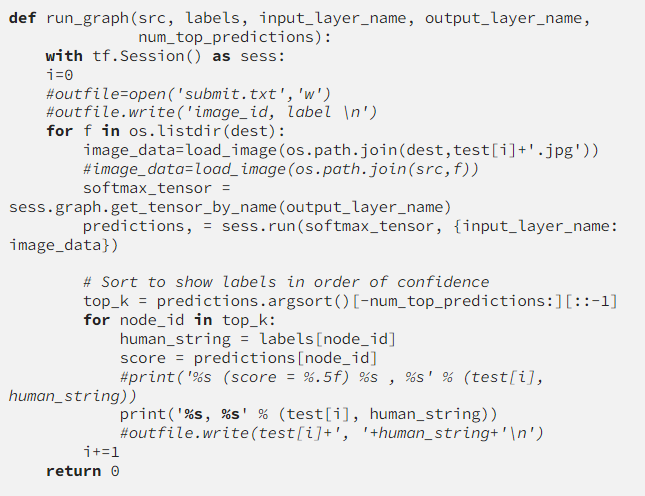




# **Step 2: retraining the bottleneck and fine-tuning the model**

* **image\_dir**: path to folder of labeled images. Fortunately, we properly set it up during preprocessing step.
* **output\_graph, intermediate\_output\_graphs\_dir, output\_labels, etc**.: where to save output files.
* **distortion feature:** my favorite. This feature alone deserves a whole paragraph. You may have noticed that images in our training set are perfect (clear, of high quality, unambiguous) but this is unfortunately not always the case in production. The algorithm may and will encounter after being deployed, fuzzy images, dimly lit images, etc.

# **Step 3: testing the model on unseen records**



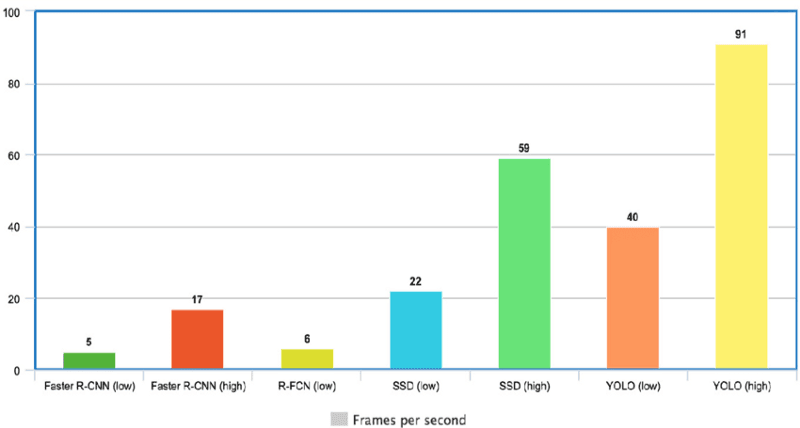
**YOLO object recognition:-**

Some of the reasons why YOLO is leading the competition include its:

* Speed
* Detection accuracy
* Good generalization
* Open-source

Speed:-

YOLO is extremely fast because it does not deal with complex pipelines. It can process images at 45 Frames Per Second (FPS). In addition, YOLO reaches more than twice the mean Average Precision (mAP) compared to other real-time systems, which makes it a great candidate for real-time processing.



High detection accuracy:-

YOLO is far beyond other state-of-the-art models in accuracy with very few background errors.

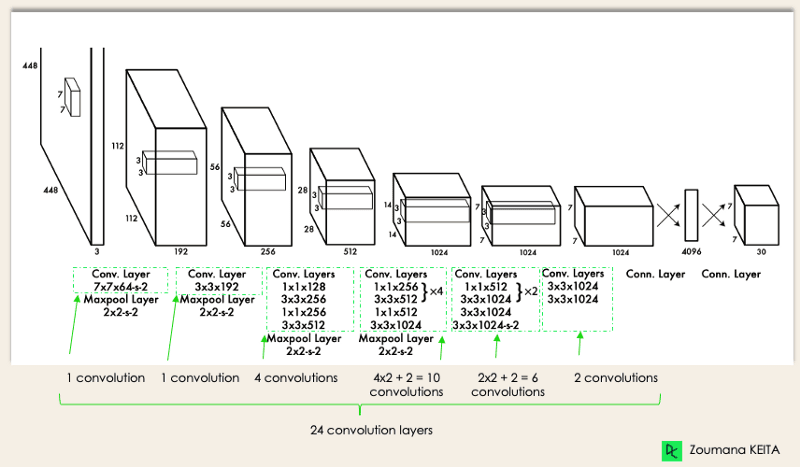
Better generalization:-

This is especially true for the new versions of YOLO, which will be discussed later in the article. With those advancements, YOLO pushed a little further by providing a better generalization for new domains, which makes it great for applications relying on fast and robust object detection.

Open source:-

Making YOLO open-source led the community to constantly improve the model. This is one of the reasons why YOLO has made so many improvements in such a limited time.

YOLO architecture:-



* Resizes the input image into 448x448 before going through the convolutional network.
* A 1x1 convolution is first applied to reduce the number of channels, which is then followed by a 3x3 convolution to generate a cuboidal output.
* The activation function under the hood is ReLU, except for the final layer, which uses a linear activation function.
* Some additional techniques, such as batch normalization and dropout, respectively regularize the model and prevent it from overfitting

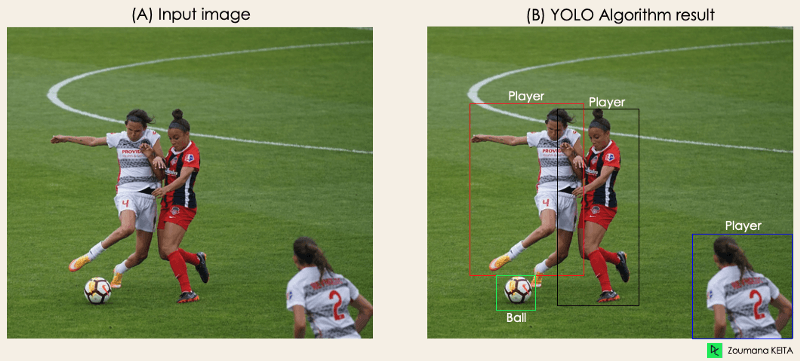
## **How Does YOLO Object Detection Work?**

Now that you understand the architecture, let’s have a high-level overview of how the YOLO algorithm performs object detection using a simple use case.

*“Imagine you built a YOLO application that detects players and soccer balls from a given image.*

*But how can you explain this process to someone, especially non-initiated people?*

*→ That is the whole point of this section. You will understand the whole process of how YOLO performs object detection; how to get image (B) from image (A)”*



The algorithm works based on the following four approaches:

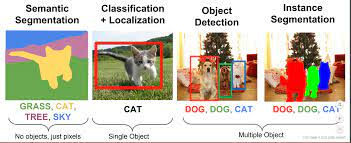
* Residual blocks
* Bounding box regression
* Intersection Over Unions or IOU for short
* Non-Maximum Suppression.

**Image segmentation:-**

Computer vision is a branch of AI that deals with processing and comprehending digital pictures. Computer vision is a branch of AI that deals with processing and comprehending digital pictures. Image segmentation deep learning is crucial in computer vision, with applications as diverse as self-driving automobiles and medical image analysis. Image segmentation is known as the technique of segmenting a digital image into many pieces.

It aims to group pixels based on their similarity, which can be measured in color, intensity, texture, or any other characteristic. Object recognition, medical image analysis, and a variety of other applications can all benefit from image semantic segmentation machine learning.

Image segmentation deep learning may be done using various techniques, including clustering, region expanding, and thresholding algorithms. Furthermore, because deep learning models offer a clear representation of what characteristics the model has learned, they are generally easier to comprehend than older techniques. Without human involvement, a deep learning machine learning model can learn to extract a face from an image.



**Project using MxNet:-**

MXNet is an open-source deep learning framework that is used to define, train and deploy neural networks. MXNet is short for mix-net because this framework was developed by combining various programming approaches into one.

Along with the aforementioned languages, trained MXNet models can be used for prediction in MATLAB and JavaScript. Regardless of the model-building language, MXNet calls optimized C++ as the back-end engine. Moreover, it is scalable and runs on systems ranging from mobile devices to distributed graphics processing unit (GPU) clusters. Not only does the MXNet framework enable fast model training, it scales automatically to the number of available GPUs across multiple hosts and multiple machines. MXNet also supports data synchronization over multiple devices with multiple users. MXNet research has been conducted at several universities, including Carnegie Mellon University, and Amazon uses it as its deep-learning framework due to its GPU capabilities and cloud computing integration.



Apache MXNet (MXNet) is an open source deep learning framework that allows you to define, train, and deploy deep neural networks on a wide array of platforms, from cloud infrastructure to mobile devices.

At its core, MXNet contains a dynamic dependency scheduler that automatically parallelizes both symbolic and imperative operations on the fly. A graph optimization layer on top of that makes symbolic execution fast and memory efficient. MXNet is portable and lightweight, scalable to many GPUs and machines.

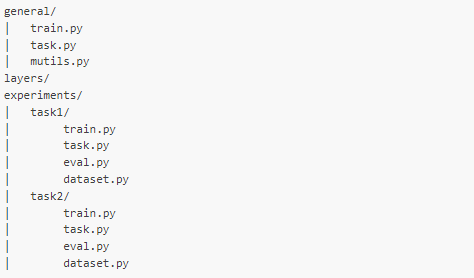
**Project using pytorch:-**

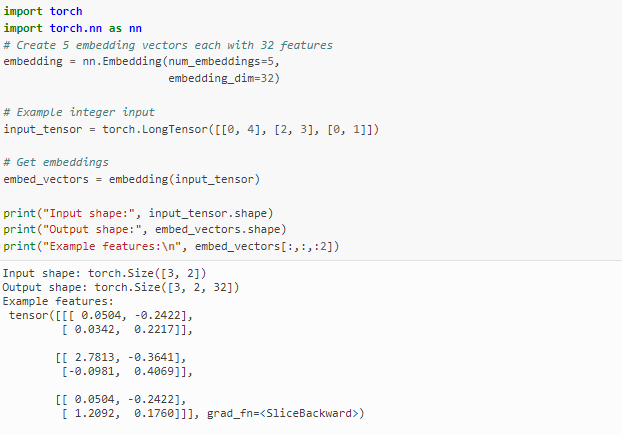
Steps:-

Setting up the dataset.

Creating the data loaders.

Creating the training, validation, and testing loops







**Social distancing detector:-**

Social-distancing is an important way to slow down the spread of infectious diseases. People are asked to limit their interactions with each other, reducing the chances of the disease being spread with physical or close contact.

* Step 1 — Installing the project
* Step 2 —Running the project
* Step 3 —Inputs and Outputs of the code
* Step 4— How does it work?
* Step 5— Suggested improvements

**Face mask detector:-**



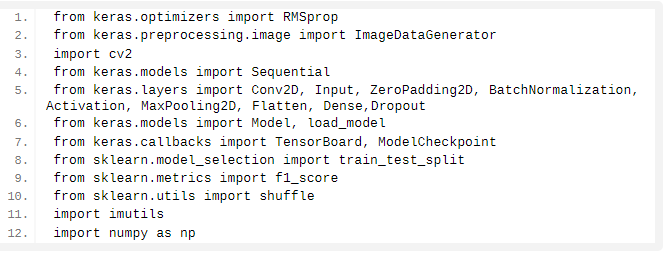
The wearing of the face masks appears as a solution for limiting the spread of COVID-19. In this context, efficient recognition systems are expected for checking that people's faces are masked in regulated areas.

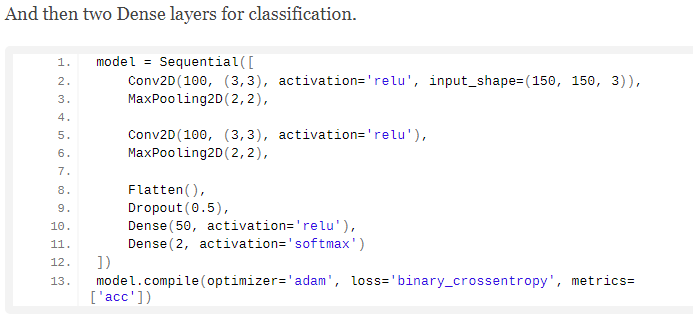
Steps:-

Download the dataset.

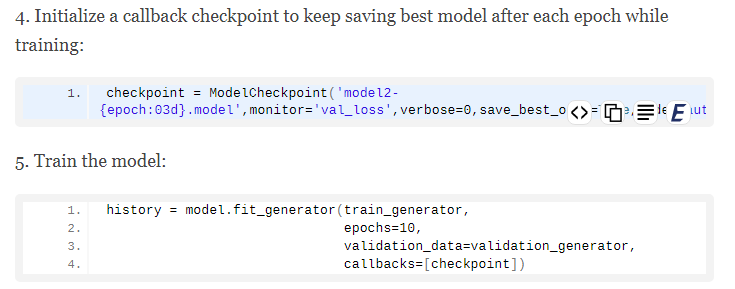
Download the project code.

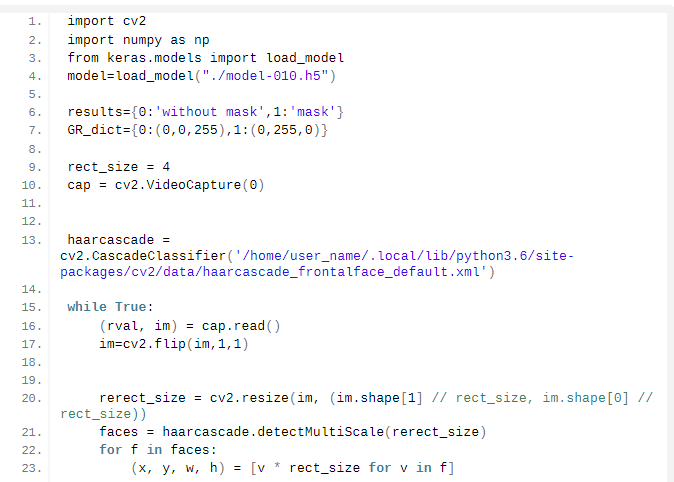
Install jupyter notebook.

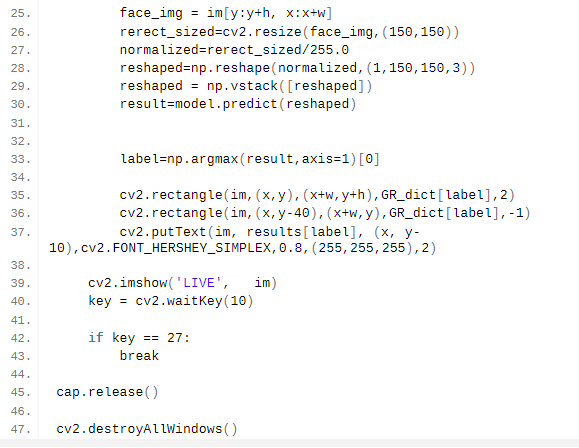






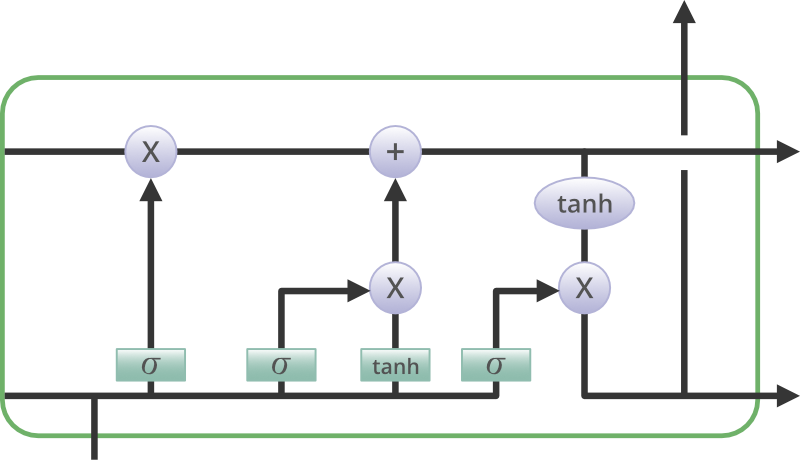






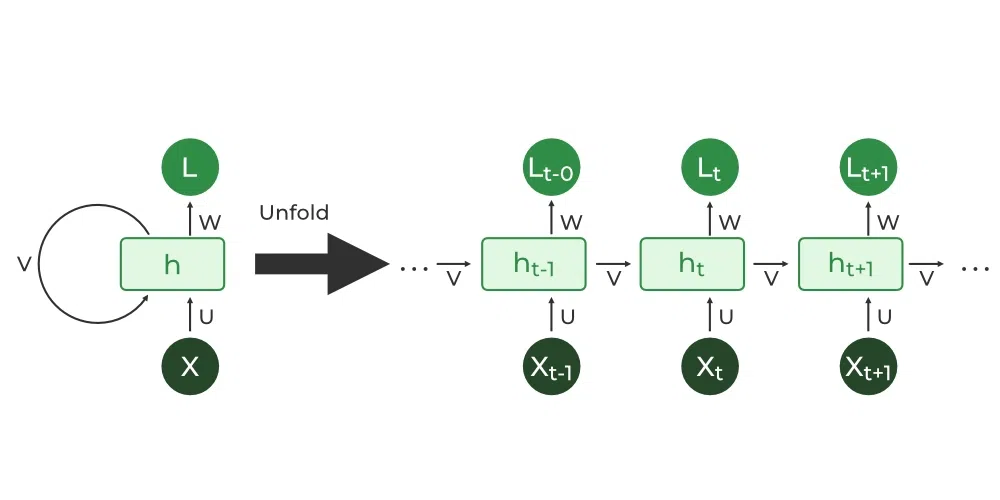
**Introduction to RNN and LSTM:-**

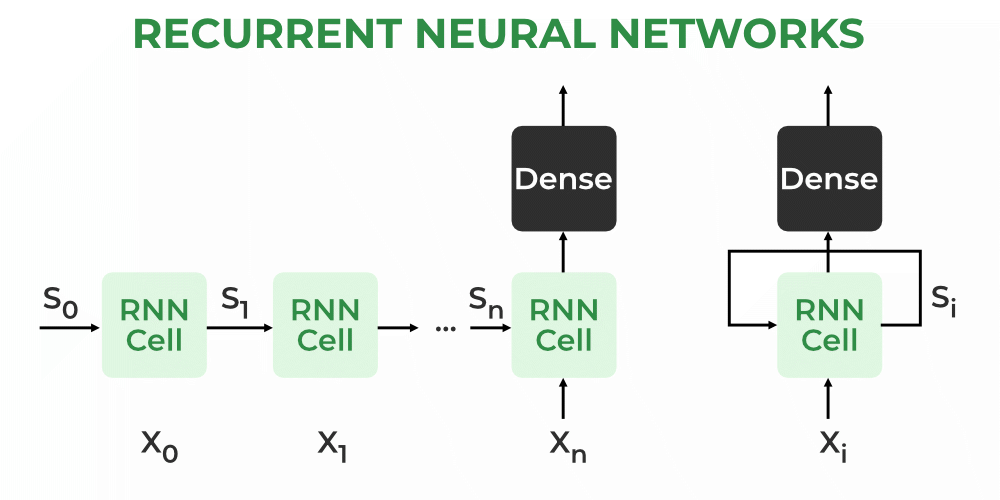
Long Short Term Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step. LSTM was designed by Hochreiter & Schmidhuber. It tackled the problem of long-term dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information. As the gap length increases RNN does not give an efficient performance. LSTM can by default retain the information for a long period of time. It is used for processing, predicting, and classifying on the basis of time-series data.



Recurrent neural network:-

Recurrent Neural Network(RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words.

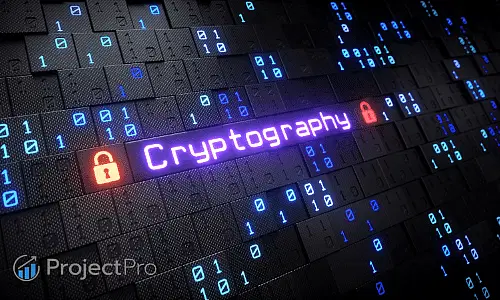




**Project using rnn:-**

### 1) Neural Network Projects in Cryptographic Applications

You can explore a wide range of applications in Cryptography using deep learning models such as neural networks. For instance, we have proposed a sequential machine using the Jordan network. Activation values of the output units in the case of a Jordan network are fed again in the input layer. The process is given shape using additional input units referred to as the state units. You can design such an application for encryption and decryption involving state diagrams and tables. State table will provide you with a training set and the input will comprise all the possible inputs and states. The output will include the encrypted/decrypted output along with the next state.



You can also use neural networks to develop cryptographic applications with the chaotic network. Chaotic neural networks are the ones that have the weights depending on a chaotic sequence. This sequence is dependent on the initial conditions and the associated parameters, x, and μ. The application will be exceptional from the security aspect as decryption of the data sets will be too complex without the details of x and μ.

* Review the basics of neural networks to understand their structure and functionality.
* Learn how to load and preprocess time series data for stock prices, ensuring data quality and consistency.
* Frame stock price prediction as a time series forecasting problem, considering the temporal order of the data and splitting it into training and testing sets.
* Learn the principles and architecture of recurrent neural networks (RNNs).
* Understand how RNNs handle sequential data and capture temporal dependencies.
* Evaluation Metrics for Model Performance.
* Dive into long short-term memory (LSTM) networks, a specialized type of RNN.
* Understand the advantages of LSTMs in capturing long-term dependencies and mitigating the vanishing gradient problem.
* Learn how to incorporate additional features or factors that may influence stock prices such as Relative Strength Index (RSI) and Exponential Moving Average (EMA) and formulate a multivariate input problem, to enhance prediction accuracy.
* Understand the inherent challenges and limitations of stock price prediction, including market volatility, unforeseen events, and the presence of noise in financial markets.

**Introduction CUDA toolkit and cudnn for deep learning:-**

Deep learning researchers and framework developers worldwide rely on cuDNN for high-performance GPU acceleration. It allows them to focus on training neural networks and developing software applications rather than spending time on low-level GPU performance tuning. cuDNN accelerates widely used deep learning frameworks, including Caffe2, Chainer, Keras, MATLAB, MxNet, PaddlePaddle, PyTorch, and TensorFlow. For access to NVIDIA optimized deep learning framework containers that have cuDNN integrated into frameworks, visit NVIDIA GPU CLOUD to learn more and get started.

Cudnn features:-

* Tensor Core acceleration for all popular convolutions including 2D, 3D, Grouped, Depth-wise separable, and Dilated with NHWC and NCHW inputs and outputs
* Optimized kernels for computer vision and speech models including ResNet, ResNext, EfficientNet, EfficientDet, SSD, MaskRCNN, Unet, VNet, BERT, GPT-2, Tacotron2 and WaveGlow
* Support for FP32, FP16, BF16 and TF32 floating point formats and INT8, and UINT8 integer formats
* Support for fusion of memory-limited operations like pointwise and reduction with math-limited operations like convolution and matmul
* Support for Windows and Linux with the latest NVIDIA data center and mobile GPUs.

Cudnn frameworks:-

1.Caffe.

2.caffee2.

3.chainer.

4.microsoft cognitive toolkit.

5.matlab.

6.mxnet.

7.paddlepaddle.

8.pytorch.

9.tensorflow.

10.torch.

11.wolfram language.

**Getting started with the intel movidius neural compute stick:-**

Intel sent me a Movidius Neural Compute Stick. It’s a USB stick a little larger than a thumb drive that is specifically designed to train and primarily run neural network graphs, which is particularly useful in running networks for deep learning where learning happened from media such as images and video. I’ll likely cover deep learning in a future post. From benchmarks, the Movidius neural compute stick promises to run models up to five times faster than a standard laptop.

### **Setting up a virtual machine**

The first step is getting a virtual machine(VM) up and running. Although there are a number of different VM software options, Virtual Box is a freely available one that’s simple to configure and use. Alternatives such as Parallels and VMWare may provide better performance if the VM is intended to be used as a primary workstation.

1. [Download Virtual Box](https://www.virtualbox.org/wiki/Downloads).
2. Install Virtual Box using the downloaded installer.
3. [Download Virtual Box Extension Pack](https://www.virtualbox.org/wiki/Downloads).
4. Install Virtual Box Extension Pack using the downloaded installer.
5. [Download Ubuntu 16.04 64 bit ISO image](http://releases.ubuntu.com/16.04/).
6. Create a new virtual machine.
7. Load Ubuntu 16.04 image as optical disk on the newly created virtual machine.
8. Start the virtual machine.
9. Follow the steps to install Ubuntu on the virtual machine.

My virtual machine specifications:

These are the configurations I used. Feel free to adjust the memory(RAM) and hard disk allocation as you see fit. Keep in mind that over-allocating resources will result in poor performance on the host operating system.

* Name: Ubuntu 16.04
* Type: Linux
* Memory Size: 3072 MB
* Virtual hard disk: 40 GB

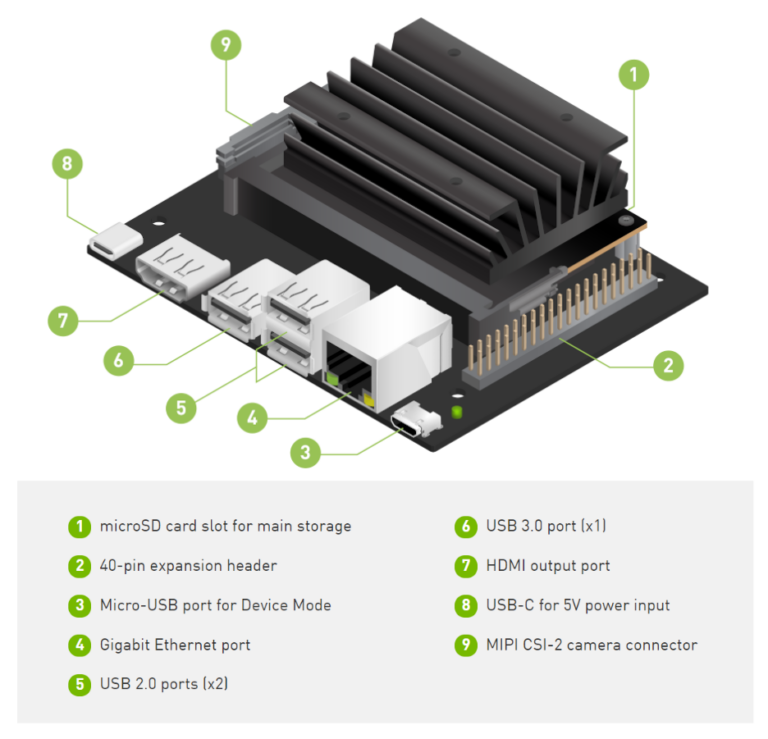
### **Install NCSDK**

The NCSDK is required to interact with the Movidius stick. The goal of the SDK is to provide an interface to neural compute hardware. This means that machine learning programs can be written to take advantage of the optimisation of purpose-specific hardware by using this SDK.

1. Clone the NCSDK (Neural Compute Software Development Kit) repository in Console: git clone <https://github.com/movidius/ncsdk.git>

**Custom object classification using nvidia jetson:-**

There were two reasons why **using our API was not an option**. First, the factory has **unstable internet connectivity**. Also, the entire solution needs to **run in real-time**. So we chose to experiment with embedded hardware that can be deployed in such an environment, and we are very glad that we found Nvidia Jetson Nano.



**Jetson Nano is an amazing small computer (embedded or edge device) built for AI.** It allows you to do machine learning in a very efficient way with low-power consumption (about 5 watts). It can be a part of IoT (Internet of Things) systems, running on Ubuntu & Linux, and is suitable for simple robotics or computer vision [projects in factories](https://www.ximilar.com/visual-ai-takes-quality-control-to-a-new-level/). **However,** **if you know that you will need to detect, recognize and track tens of different labels, choose the higher version of Jetson embedded hardware, such as** [**Xavier**](https://developer.nvidia.com/embedded-computing). It is a much faster device than Nano and can solve more complex problems.

* You need a **real-time analysis**
* Your problem can be solved with **one or two simple models**
* You need a **budget solution** & be cost-effective when running the system
* You want to **connect it to a static camera** – for example, monitoring an assembly line
* The system **cannot be connected to the internet** – for example, because your factory is in a remote place or for security reasons

## Image Recognition on Jetson Nano

For any [image categorization](https://www.ximilar.com/services/image-recognition/) problem, I would recommend using simple architecture as [MobileNetV2](https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html). You can select for example the depth multiplier for mobilenet of 0.35 and image resolution 128×128 pixels. In this way, you can **achieve great performance both in speed and precision**.

## Developing on Jetson Nano

The experience of programming challenging projects, exploring new gadgets, and helping our customers is something that deeply satisfies us. We are looking forward to trying other hardware for machine learning such as [**Coral**](https://coral.ai/) from Google, [**Raspberry Pi**](https://www.raspberrypi.org/), or [**Intel Movidius**](https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu.html#:~:text=Intel%C2%AE%20Movidius%E2%84%A2%20VPUs,edge%20AI%20workloads%20with%20efficiency.&text=VPU%20technology%20enables%20intelligent%20cameras,and%20safety%2C%20and%20industrial%20automation.) for Industry 4.0 projects.

Most of the time, we are developing a machine learning API for large [e-commerce](https://www.ximilar.com/how-it-works/e-commerce/) sites. We are really glad that our platform can also help us **build machine learning models on devices running in distant parts of the world** with no internet connectivity. I think that there are much more opportunities for similar projects in the future.

## 

## yers