Machine Learning

Project Title:

To identify ships in a satellite image

Submitted By:

P.K. Kartikeya Rao

Roll.no:221710308040

3rd year, CSE

GITAM Deemed to Be University

Hyderabad

Table of Contents

MACHINE LEARNING

1. INFORMATION ABOUT MACHINE LEARNING

Machine Learning (ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence (AI).

1.1 Importance of Machine Learning

Machine learning has several very practical applications that drive the kind of real business results – such as time and money savings – that have the potential to dramatically impact the future of your organization. At Interactions in particular, we see tremendous impact occurring within the customer care industry, whereby machine learning is allowing people to get things done more quickly and efficiently.

Through Virtual Assistant solutions, machine learning automates tasks that would otherwise need to be performed by a live agent – such as changing a password or checking an account balance. This frees up valuable agent time that can be used to focus on the kind of customer care that humans perform best: high touch, complicated decision-making that is not as easily handled by a machine.

At Interactions, we further improve the process by eliminating the decision of whether a request should be sent to a human or a machine: unique Adaptive Understanding technology, the machine learns to be aware of its limitations, and bail out to humans when it has a low confidence in providing the correct solution

Machine learning has made dramatic improvements in the past few years, but we are still very far from reaching human performance. Many times, the machine needs the assistance of human to complete its task. At Interactions, we have deployed Virtual Assistant solutions that seamlessly blend artificial with true human intelligence to deliver the highest level of accuracy and understanding.

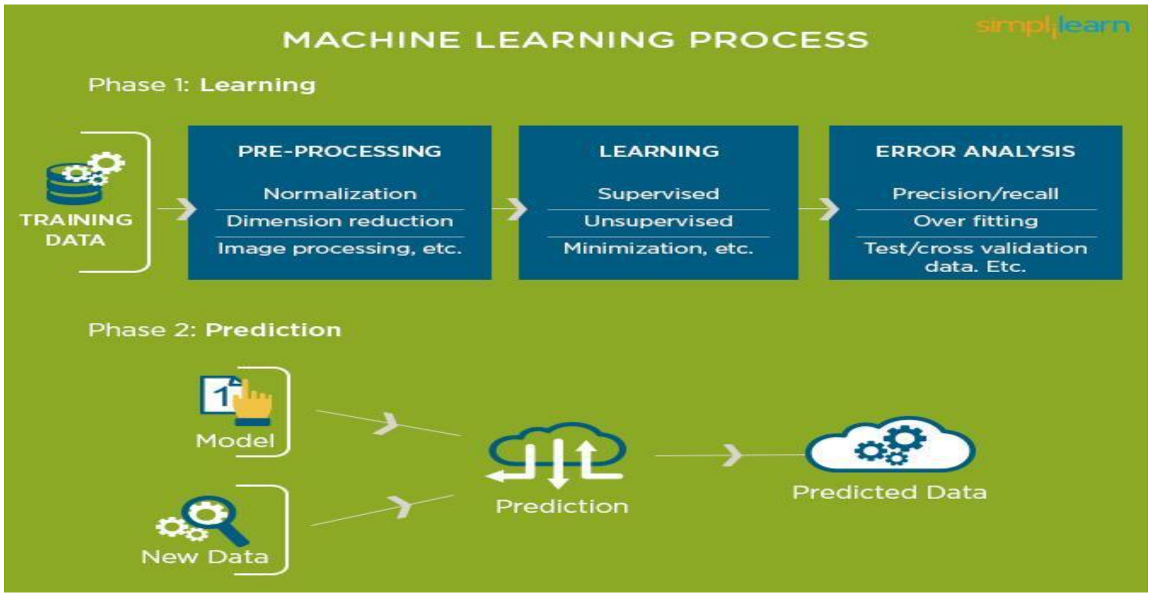


Fig: ML Process

1.2 Uses of Machine Learning

There are limitless applications of machine learning and there are a lot of machine learning algorithms are available to learn. They are available in every form from simple to highly complex. Top 10 Uses of machine learning are as follows:

- Image recognition
- Voice recognition
- Predictions
- Video Surveillance
- Social Media Platform
- Spam and Malware
- Customer Support
- Search Engine

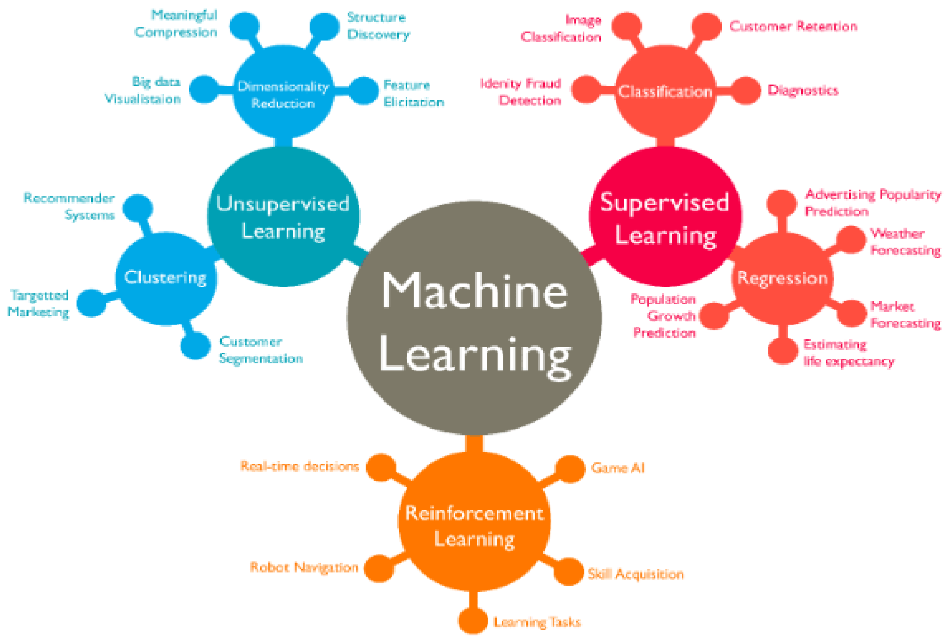


Fig: ML Uses

1.3 Types of Machine Learning

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

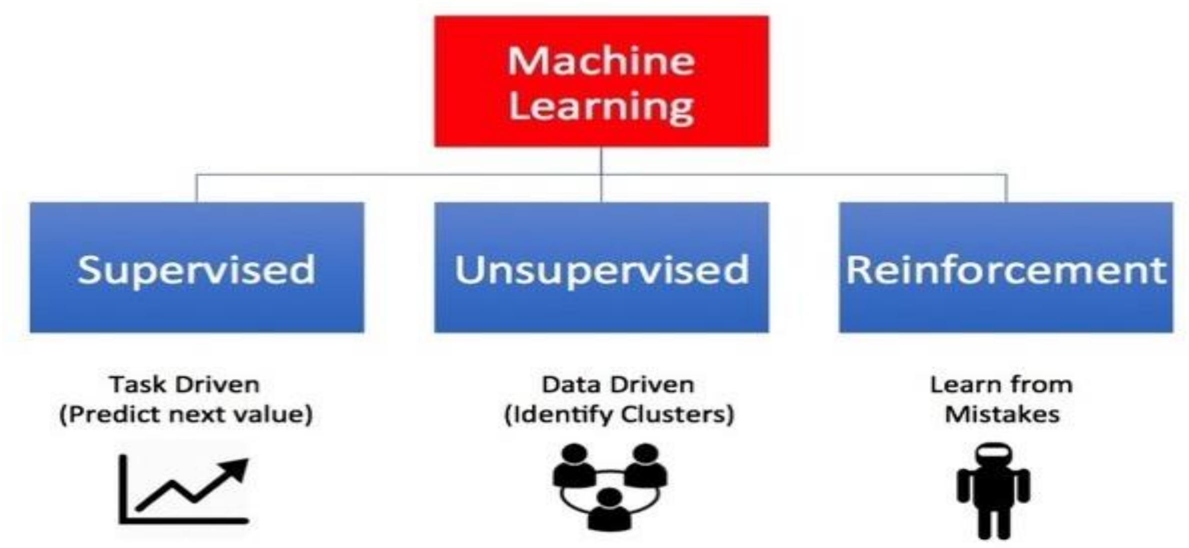


Fig: Types in Machine Learning

1.3.1 Supervised Learning

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning. Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data. Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign. Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

1.3.2 Unsupervised Learning

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms. Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

1.3.3 Semi Supervised Learning

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

DEEP LEARNING:

2.1.1 Importance of Deep Learning

The ability to process large numbers of features makes deep learning very powerful when dealing with unstructured data. However, deep learning algorithms can be overkill for less complex problems because they require access to a vast amount of data to be effective. For instance, ImageNet, the common benchmark for training deep learning models for comprehensive image recognition, has access to over 14 million images.

If the data is too simple or incomplete, it is very easy for a deep learning model to become overfitted and fail to generalize well to new data. As a result, deep learning models are not as effective as other techniques (such as boosted decision trees or linear models) for most practical business problems such as understanding customer churn, detecting fraudulent transactions and other cases with smaller datasets and fewer features. In certain cases, like multiclass classification, deep learning can work for smaller, structured datasets.

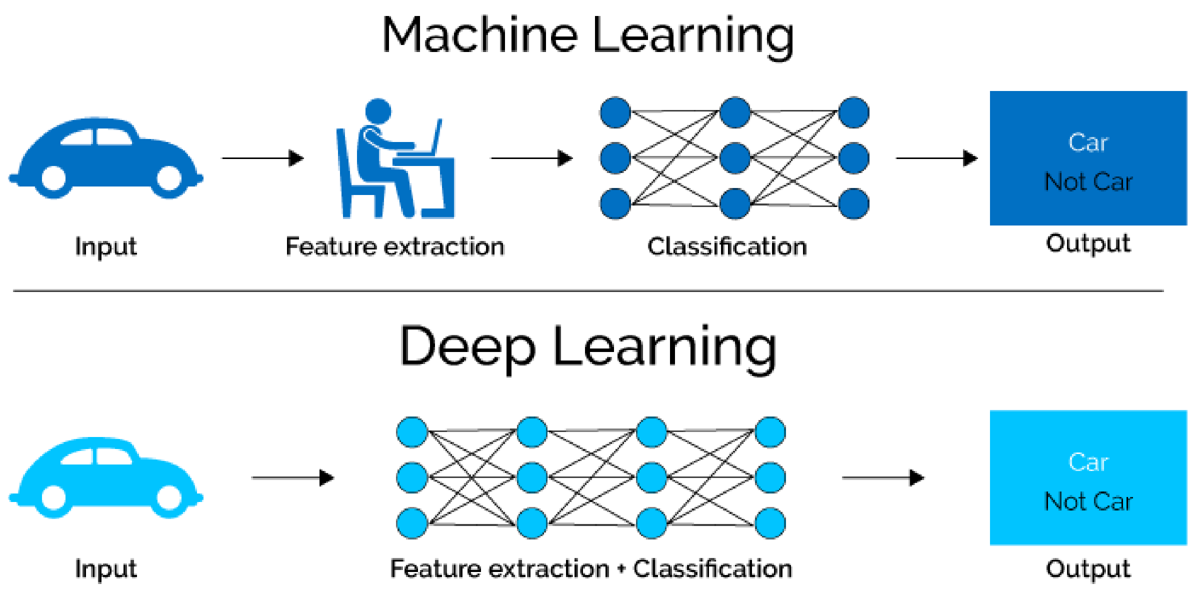


Fig: Working of ML And DL

2.1.2 Uses of Deep Learning

Deep learning applications are used in industries from automated driving to medical devices.

- Automated Driving
- Aerospace and Defense
- Medical Research
- Industrial Automation
- Electronics

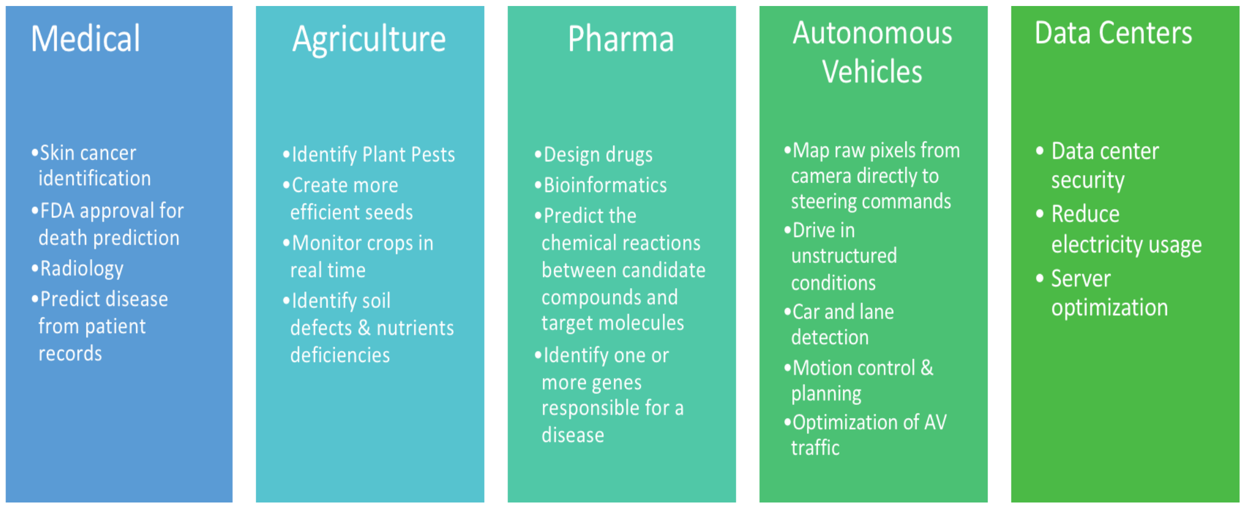

Medical
•Skin cancer identification
•FDA approval for death prediction
•Radiology
•Predict disease from patient records
Agriculture
•Identify Plant Pests
•Create more efficient seeds
•Monitor crops in real time
•Identify soil defects & nutrients deficiencies
Pharma
•Design drugs
•Bioinformatics
•Predict the chemical reactions between candidate compounds and target molecules
•Identify one or more genes responsible for a disease
Autonomous Vehicles
•Map raw pixels from camera directly to steering commands
•Drive in unstructured conditions
•Car and lane detection
•Motion control & planning
•Optimization of AV traffic
Data Centers
• Data center security
• Reduce electricity usage
• Server optimization

Fig: Uses of Deep Learning

2.1.3 Relation between Data Mining, Machine Learning and Deep Learning

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns.

Python

3.1 Introduction

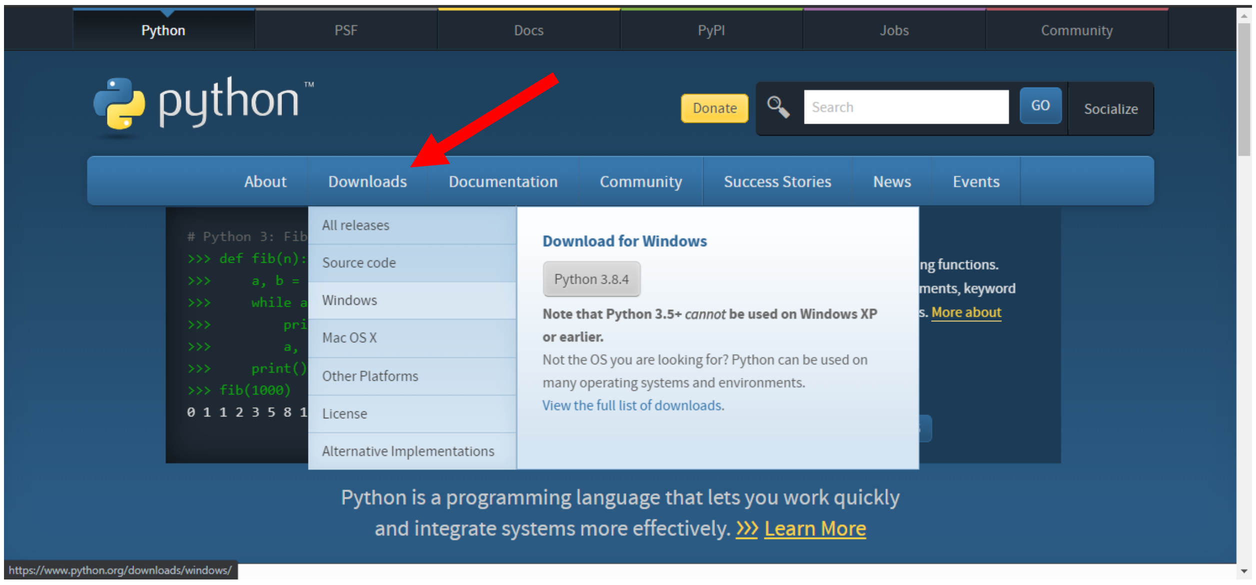
Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

3.2 Setup of Python

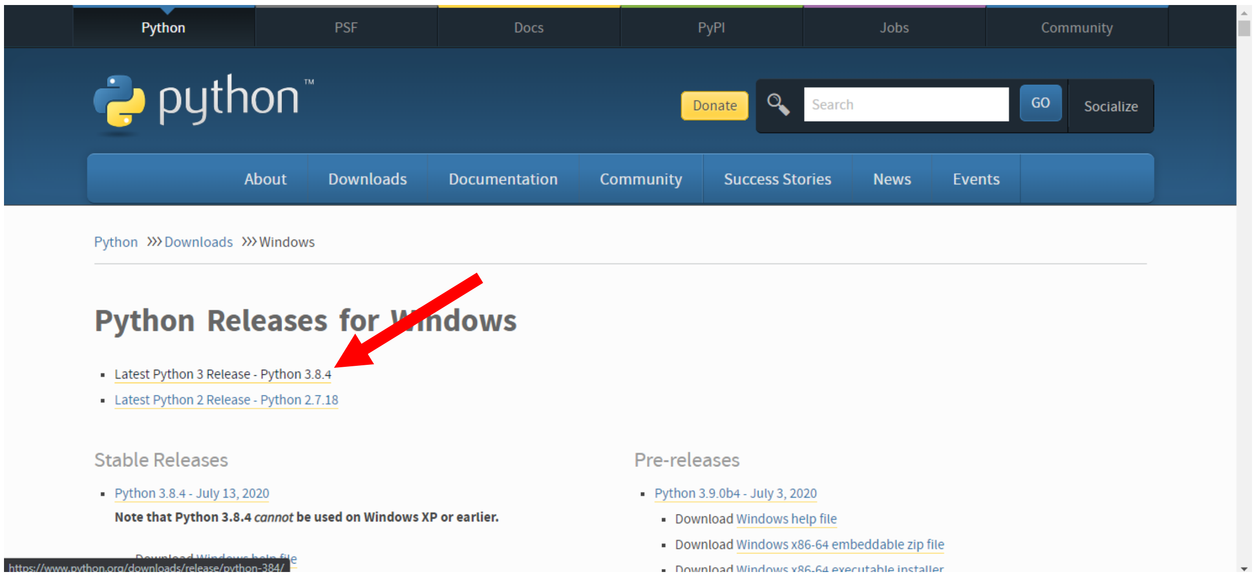
Step 1. Open https://www.python.org/ in your browser.
Step 2. Navigate to Downloads Section.
Step 3. Then Select Your preferred OS.
Step 4. Select desired version of python.
Step 5. After downloading the python file open and install the .exe file.
Step 6. To add the Python directory to the path for a particular session in Windows −At the command prompt − type path %path%;C:\Python and press Enter.



Note that Python 3.5+ *cannot* be used on Windows XP or earlier.
Not the OS you are looking for? Python can be used on many operating systems and environments.
View the full list of downloads.

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

https://www.python.org/downloads/windows/



Python >>> Downloads >>> Windows

Python Releases for Windows

- Latest Python 3 Release - Python 3.8.4
- Latest Python 2 Release - Python 2.7.18

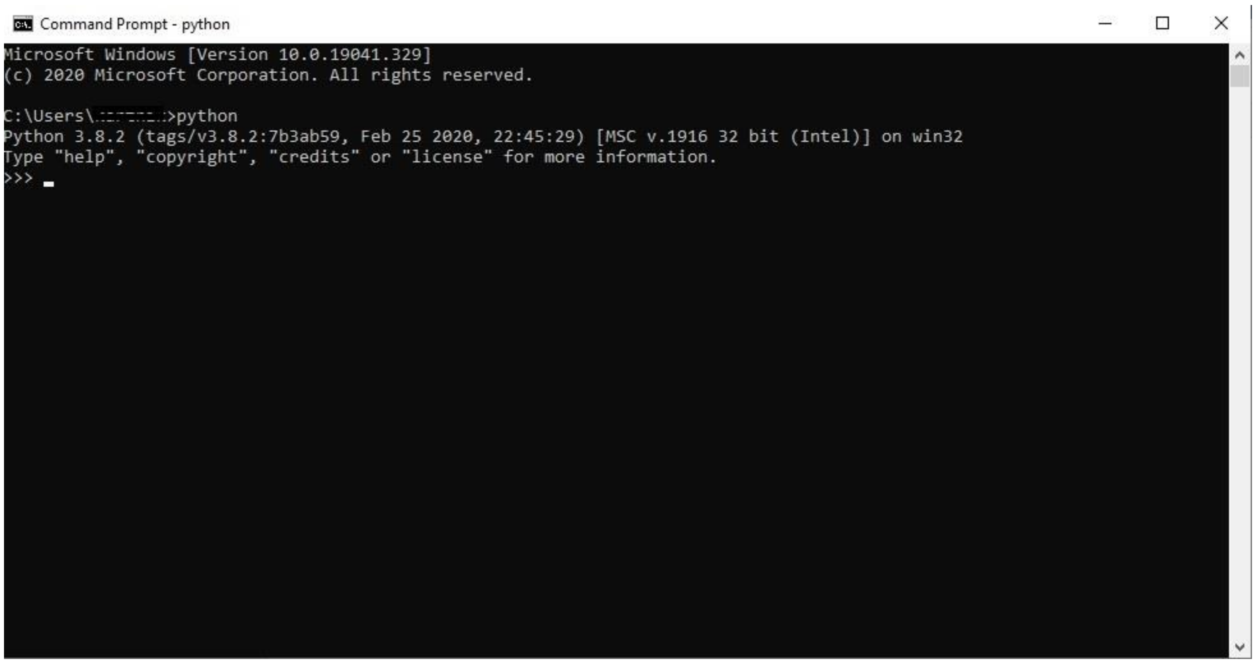
Stable Releases

- Python 3.8.4 - July 13, 2020
  Note that Python 3.8.4 *cannot* be used on Windows XP or earlier.

Pre-releases

- Python 3.9.0b4 - July 3, 2020
  - Download Windows help file
  - Download Windows x86-64 embeddable zip file
  - Download Windows x86-64 executable installer

https://www.python.org/downloads/release/python-384/

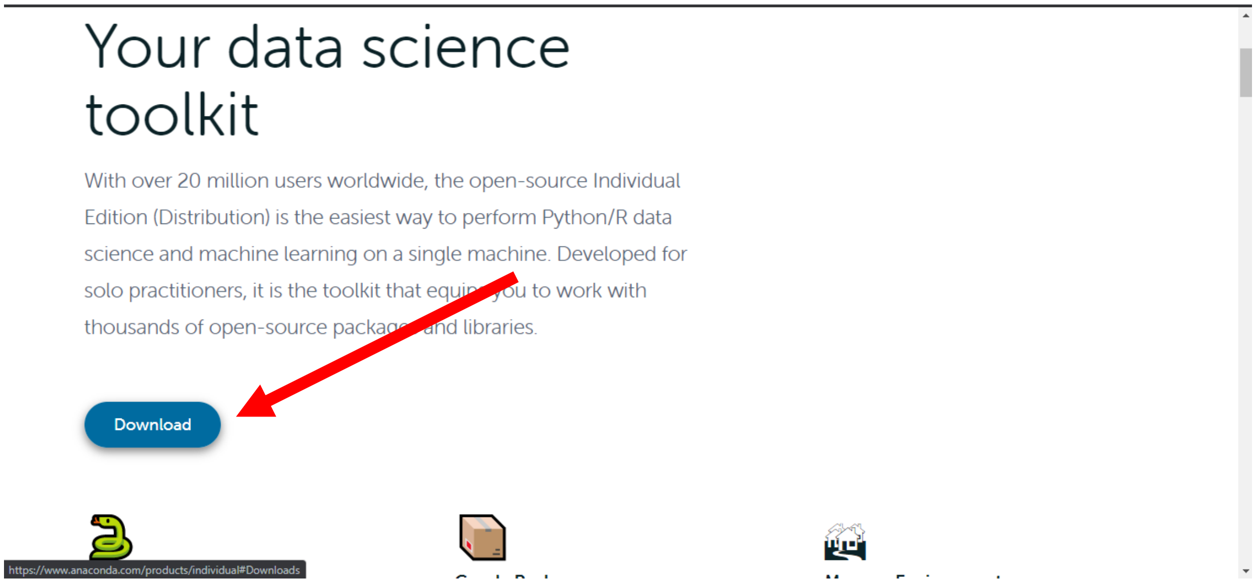


```
Microsoft Windows [Version 10.0.19041.329]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

3.3 Installing Anaconda (Notebook)

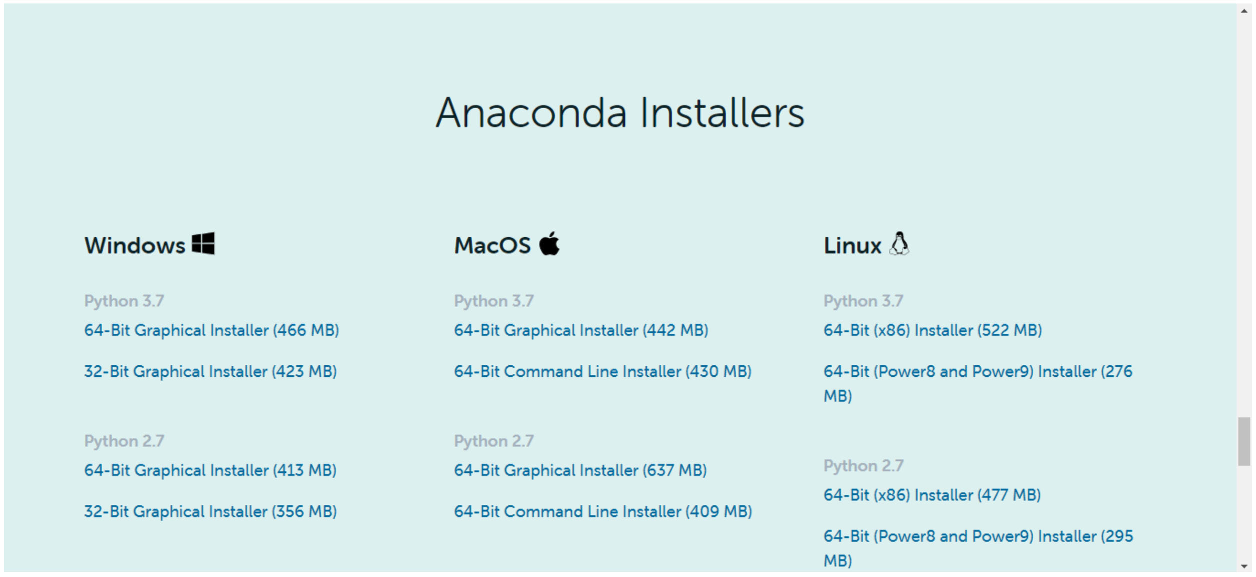
1. Navigate to https://www.anaconda.com/products/individual .
2. Click on the option Download.
3. You will be redirected to Anaconda Installers section.
4. Select your Operating System and the version of python.
5. A .exe file will be downloaded

6. Run the .exe file.
7. Select the path where Anaconda is to be installed
8. After successful installation, A entity will be created called Anaconda3.
9. Open the folder and select the jupyter Notebook.
10. A page will be opened where we can select our notebook location and start out programming.



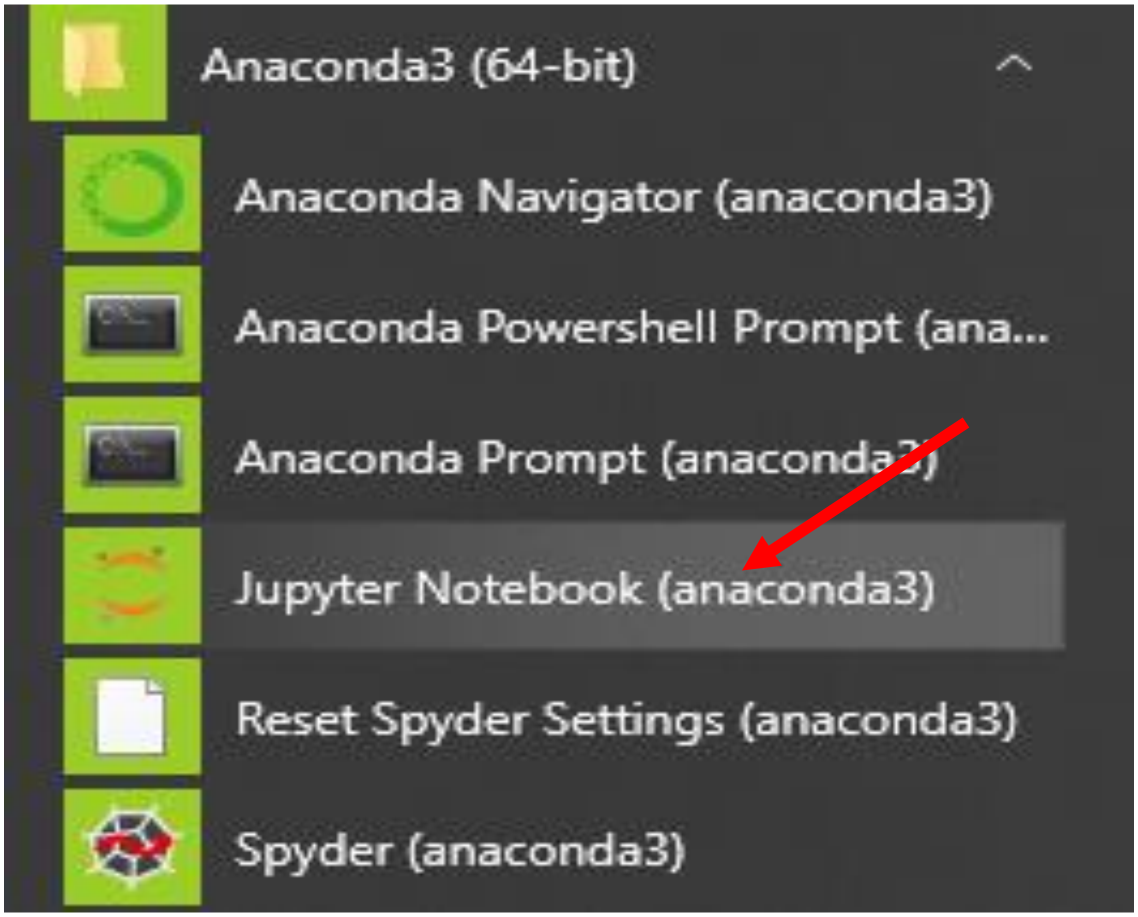
Your data science toolkit

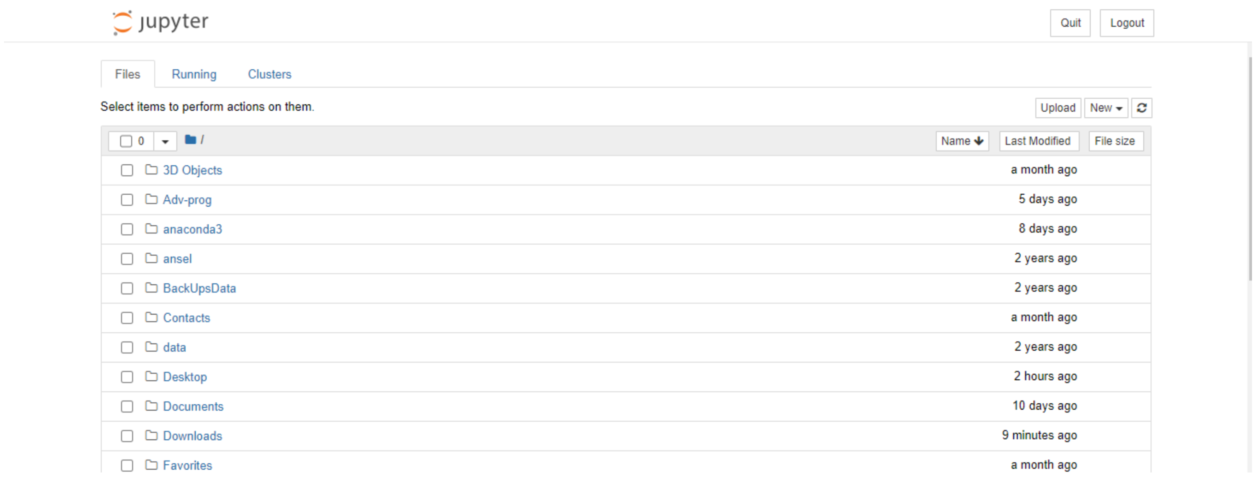
With over 20 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.



Anaconda Installers

Windows	MacOS	Linux
Python 3.7	Python 3.7	Python 3.7
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (442 MB)	64-Bit (x86) Installer (522 MB)
32-Bit Graphical Installer (423 MB)	64-Bit Command Line Installer (430 MB)	64-Bit (Power8 and Power9) Installer (276 MB)
Python 2.7	Python 2.7	Python 2.7
64-Bit Graphical Installer (413 MB)	64-Bit Graphical Installer (637 MB)	64-Bit (x86) Installer (477 MB)
32-Bit Graphical Installer (356 MB)	64-Bit Command Line Installer (409 MB)	64-Bit (Power8 and Power9) Installer (295 MB)


Anaconda3 (64-bit)
Anaconda Navigator (anaconda3)
Anaconda Powershell Prompt (ana...
Anaconda Prompt (anaconda3)
Jupyter Notebook (anaconda3)
Reset Spyder Settings (anaconda3)
Spyder (anaconda3)



Select items to perform actions on them.

3D Objects	a month ago
Adv-prog	5 days ago
anaconda3	8 days ago
ansel	2 years ago
BackUpsData	2 years ago
Contacts	a month ago
data	2 years ago
Desktop	2 hours ago
Documents	10 days ago
Downloads	9 minutes ago
Favorites	a month ago

3.4 Features

Python is a dynamic, high level, free open source and interpreted programming language. It supports object-oriented programming as well as procedural oriented programming. In Python, we don't need to declare the type of variable because it is a dynamically typed language.

- **Easy to code:**
  Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.
- **Free and Open Source:**
  Python language is freely available at the official website and Since it is open-source, this means that source code is also available to the public. So, you can download it as, use it as well as share it.
- **Object-Oriented Language:**
  One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.
- **GUI Programming Support:**
  Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python.
  PyQt5 is the most popular option for creating graphical apps with Python.
- **High-Level Language:**
  Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.
- **Extensible feature:**
  Python is an **Extensible** language. We can write us some Python code into C or C++ language and also, we can compile that code in C/C++ language.
- **Python is Portable language:**
  Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.
- **Large Standard Library**
  Python has a large standard library which provides a rich set of module and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc.
- **Dynamically Typed Language:**
  Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

3.5 Variable Types

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory. Variables are nothing but reserved memory locations to store values. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python Has 5 Standard Variables:

1.Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

2. Python Strings:

● Strings in Python are identified as a contiguous set of characters represented in the quotation marks.

● Python allows for either pairs of single or double quotes.

● Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes Starting at 0 in the beginning of the string and working their way from -1 at the end.

● The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

3. Python Lists:

● Lists are the most versatile of Python's compound data types.

● A list contains items separated by commas and enclosed within square brackets ([]).

● To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

● The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.

● The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

4.Python Tuples:

● A tuple is another sequence data type that is similar to the list.

● A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

● The main differences between lists and tuples are: Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated.

● Tuples can be thought of as read-only lists.

● For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

5.Python Dictionary:

● Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

● Dictionaries are enclosed by curly braces ({}) and values can be assigned and accessed using square braces ([]).

● You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.

● What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

3.5 Functions:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e. ()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses the code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

3.6 OOPs Concepts

1.Class

A class is a blueprint for the object.

We can think of class as a sketch of a parrot with labels. It contains all the details about the name, colors, size etc. Based on these descriptions, we can study about the parrot. Here, a parrot is an object.

The example for class of parrot can be:

```
class Parrot:
    pass
```

2.Object

An object (instance) is an instantiation of a class. When class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated.

The example for object of parrot class can be:

```
obj = Parrot ()
```

Here, obj is an object of class Parrot.

3.Methods

Methods are functions defined inside the body of a class. They are used to define the behaviors of an object.

4.Inheritance

Inheritance is a way of creating a new class for using details of an existing class without modifying it. The newly formed class is a derived class (or child class). Similarly, the existing class is a base class (or parent class).

5.Encapsulation

Using OOP in Python, we can restrict access to methods and variables. This prevents data from direct modification which is called encapsulation. In Python, we denote private attributes using underscore as the prefix i.e. single _ or double __.

6.Polymorphism

Polymorphism is an ability (in OOP) to use a common interface for multiple forms (data types).

Suppose, we need to color a shape, there are multiple shape options (rectangle, square, circle). However, we could use the same method to color any shape. This concept is called Polymorphism.

4. Ships in Satellite Imagery

4.1 Project Requirements

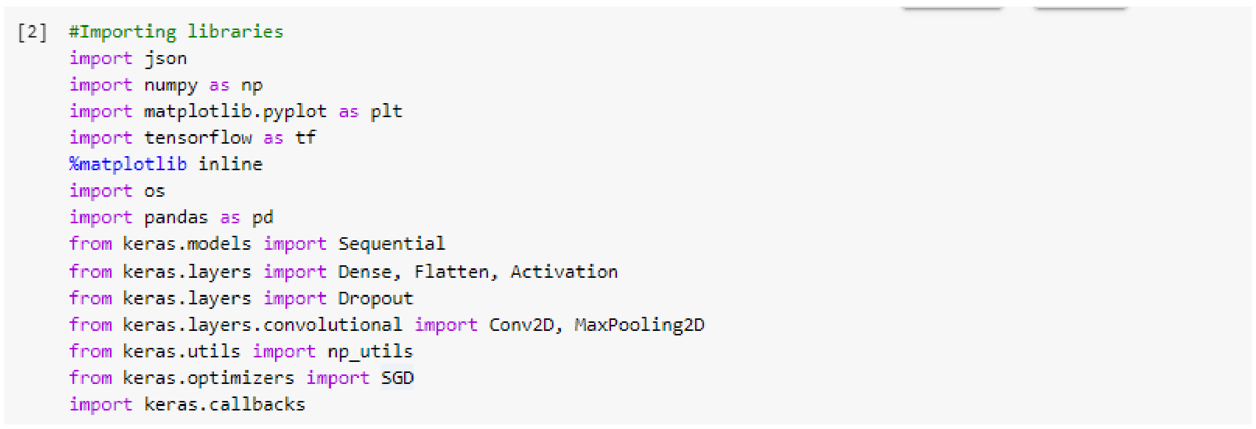
The dataset consists of image chips extracted from Planet satellite imagery collected over the San Francisco Bay and San Pedro Bay areas of California. It includes 4000 80x80 RGB images labeled with either a "ship" or "no-ship" classification. Image chips were derived from Planet Scope full-frame visual scene products, which are orthorectified to a 3-meter pixel size.

4.1.1 Packages used

Json	
Numpy	pandas
matplotlib.pyplot	keras models
tensorflow	keras layes
%matplotlib inline	keras utils

Os keras optimizers

keras callbacks



```
[2] #Importing libraries
    import json
    import numpy as np
    import matplotlib.pyplot as plt
    import tensorflow as tf
    %matplotlib inline
    import os
    import pandas as pd
    from keras.models import Sequential
    from keras.layers import Dense, Flatten, Activation
    from keras.layers import Dropout
    from keras.layers.convolutional import Conv2D, MaxPooling2D
    from keras.utils import np_utils
    from keras.optimizers import SGD
    import keras.callbacks
```

4.1.2 Versions of the packages

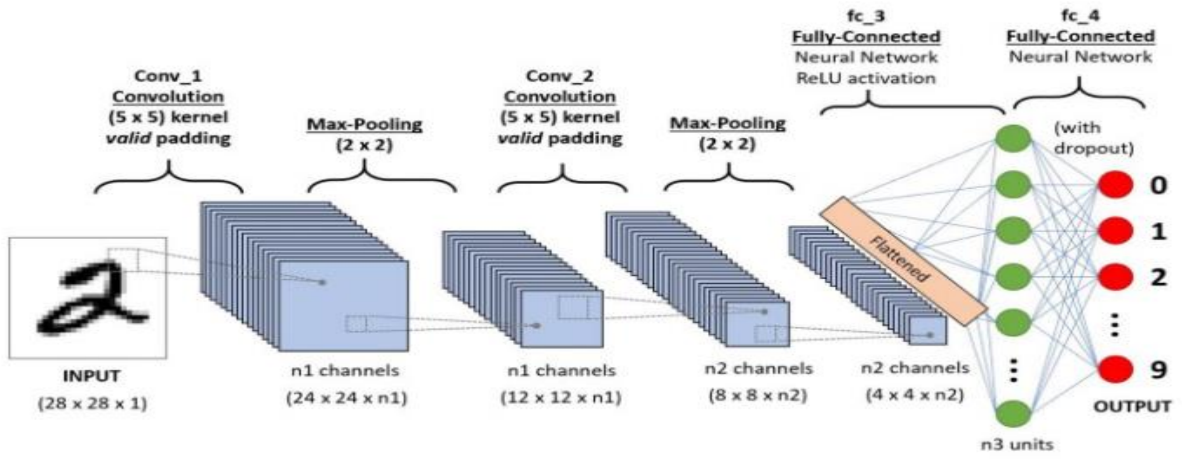
- `print(np.__version__) -→ 1.18.05`
- `print(tf.__version__) -→ 2.2.0`
- `print(pd.__version__)  -→ 1.0.5`
- `print(keras.__version__) -→ 2.3.1`

4.1.3 Algorithm used

Convolutional Neural Networks

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. Researchers and enthusiasts alike, work on numerous aspects of the field to make amazing things happen. One of many such areas is the domain of Computer Vision.

The agenda for this field is to enable machines to view the world as humans do, perceive it in a similar manner and even use the knowledge for a multitude of tasks such as Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems, Natural Language Processing, etc. The advancements in Computer Vision with Deep Learning has been constructed and perfected with time, primarily over one particular algorithm — a **Convolutional Neural Network**.



A CNN sequence to classify handwritten digits

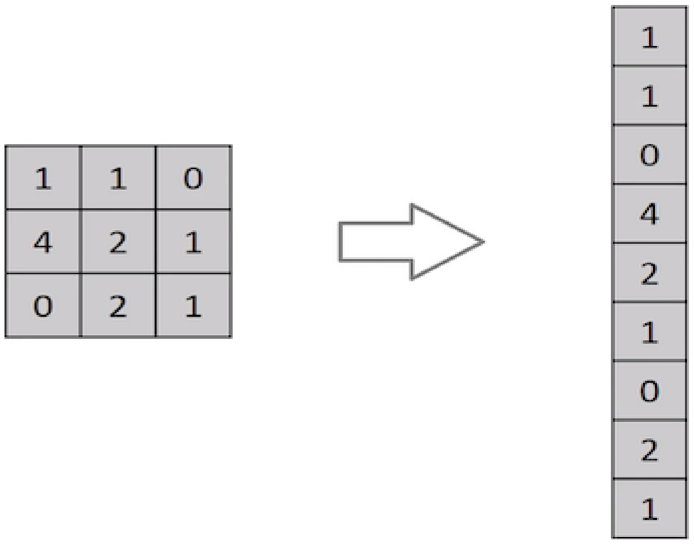
A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

Why ConvNets over Feed-Forward Neural Nets?

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes? Uh... not really.

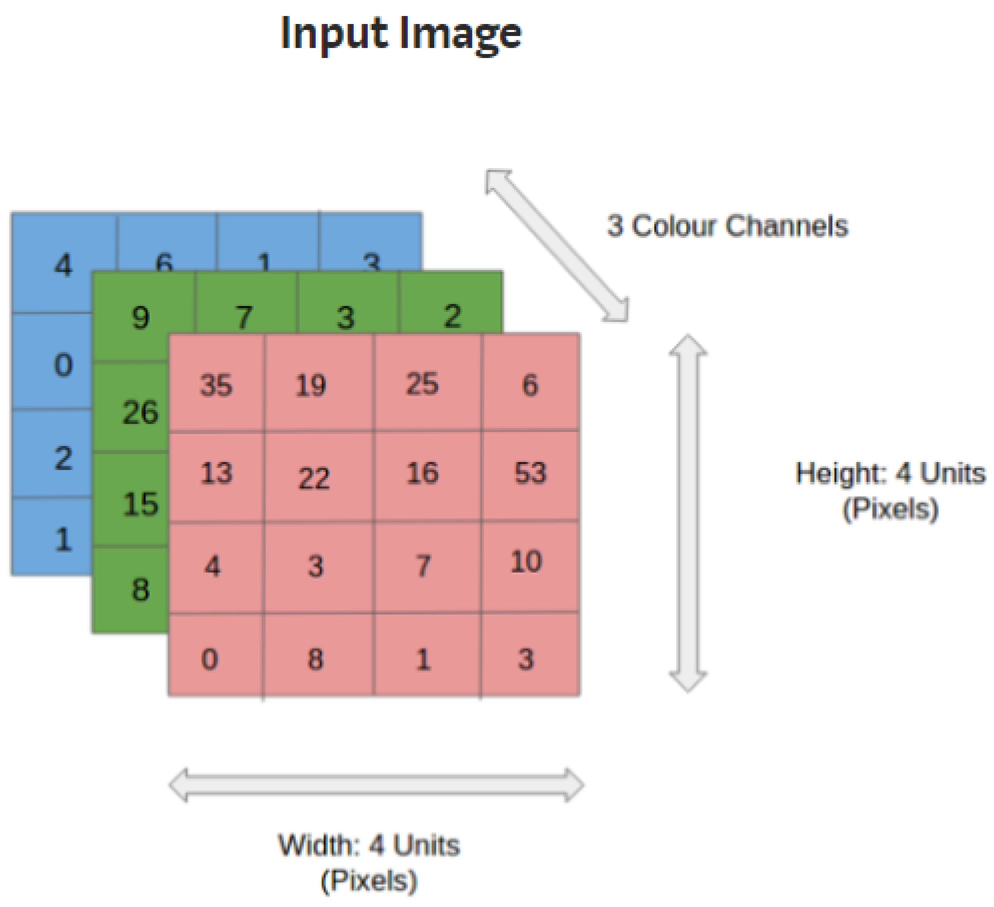
In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.



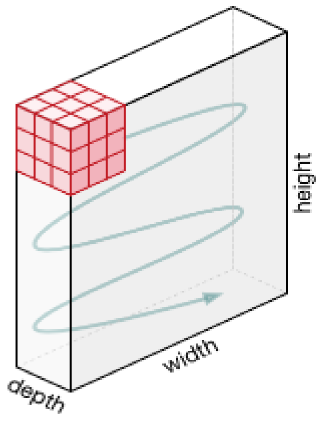
Flattening of a 3x3 image matrix into a 9x1 vector

You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.



4x4x3 RGB Image

The Kernel shifts 9 times because of **Stride Length = 1 (Non-Strided)**, every time performing a **matrix multiplication operation between K and the portion P of the image** over which the kernel is hovering.



Movement of the Kernel

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

Pooling Layer

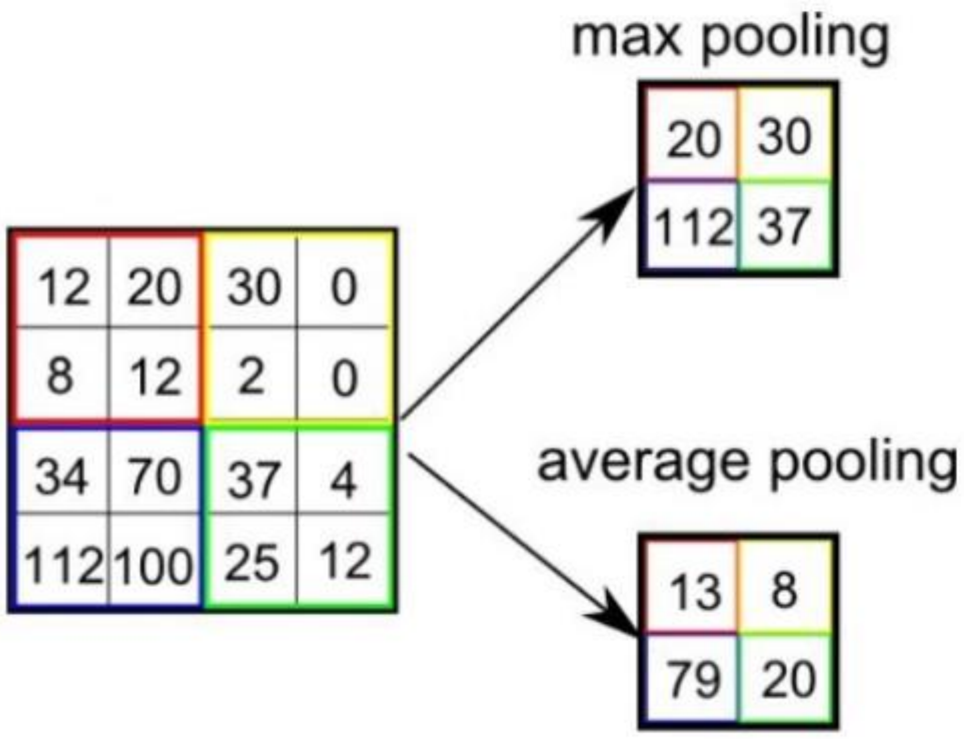
Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to **decrease the computational power required to process the data** through dimensionality reduction. Furthermore, it is useful for **extracting dominant features** which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling. **Max Pooling** returns the **maximum value** from the portion of the image covered by the Kernel. On the other hand, **Average Pooling** returns the **average of all the values** from the portion of the image covered by the Kernel.

Max Pooling also performs as a **Noise Suppressant**. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that **Max Pooling performs a lot better than Average Pooling**.

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.



Types of Pooling

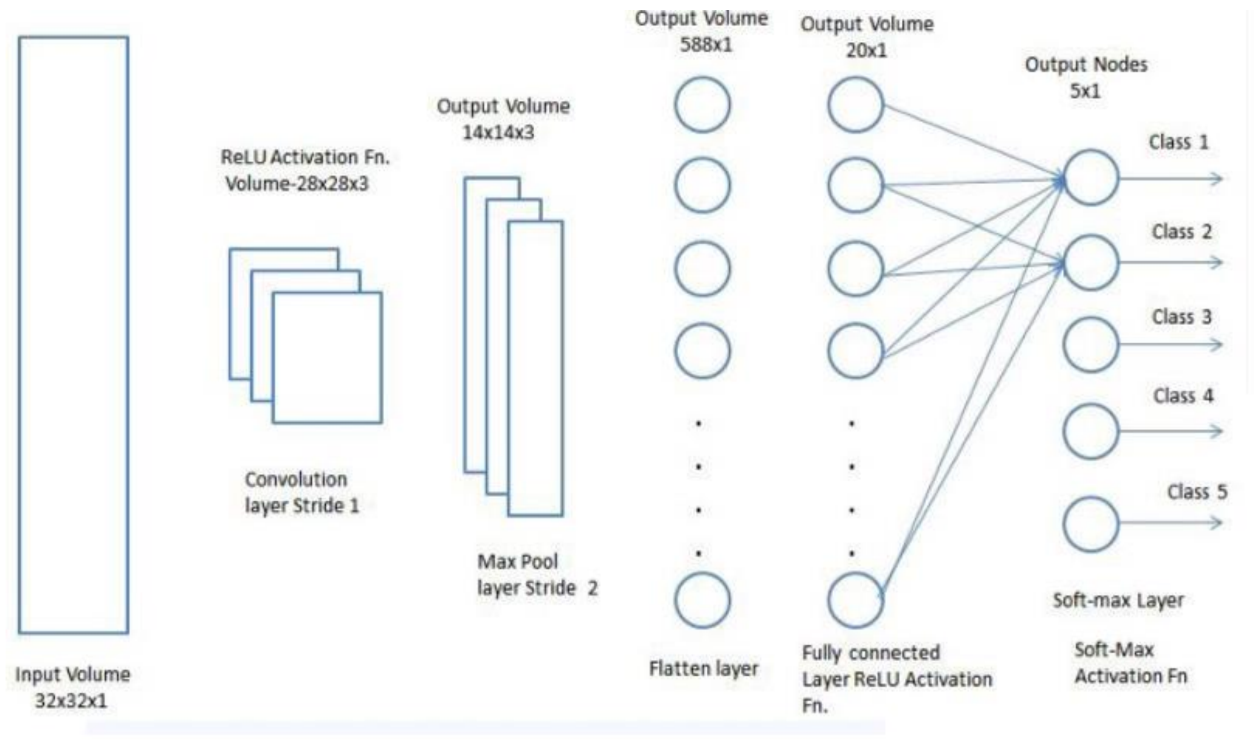
Classification — Fully Connected Layer (FC Layer)

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **Softmax Classification** technique.

There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. Some of them have been listed below:

1. LeNet
2. AlexNet
3. VGGNet
4. GoogLeNet
5. ResNet
6. ZFNet



4.2 Problem Statement

Satellite imagery provides unique insights into various markets, including agriculture, defense and intelligence, energy, and finance. New commercial imagery providers, such as Planet, are using constellations of small satellites to capture images of the entire Earth every day.

This flood of new imagery is outgrowing the ability for organizations to manually look at each image that gets captured, and there is a need for machine learning and computer vision algorithms to help automate the analysis process.

4.3 Dataset Description

The dataset consists of image chips extracted from Planet satellite imagery collected over the San Francisco Bay and San Pedro Bay areas of California. It includes 4000 80x80 RGB images labeled with either a "ship" or "no-ship" classification. Image chips were derived from Planet Scope full-frame visual scene products, which are orthorectified to a 3-meter pixel size.

Provided is a zipped directory shipsnet.zip that contains the entire dataset as .png image chips. Each individual image filename follows a specific format: {label} __ {scene id} __ {longitude} _ {latitude}.png

- label: Valued 1 or 0, representing the "ship" class and "no-ship" class, respectively.
- scene id: The unique identifier of the PlanetScope visual scene the image chip was extracted from. The scene id can be used with the Planet API to discover and download the entire scene.
- longitude_latitude: The longitude and latitude coordinates of the image center point, with values separated by a single underscore.

The dataset is also distributed as a JSON formatted text file shipsnet.json. The loaded object contains data, label, scene_ids, and location lists.

The pixel value data for each 80x80 RGB image is stored as a list of 19200 integers within the data list. The first 6400 entries contain the red channel values, the next 6400 the green, and the final 6400 the blue. The image is stored in row-major order, so that the first 80 entries of the array are the red channel values of the first row of the image.

The list values at index *i* in labels, scene_ids, and locations each correspond to the *i*-th image in the data list.

4.4 Objective of the Case Study

The aim of this dataset is to help address the difficult task of detecting the location of large ships in satellite images. Automating this process can be applied to many issues including monitoring port activity levels and supply chain analysis.

5. DATA PREPROCESSING/FEATURE ENGINEERING AND EDA

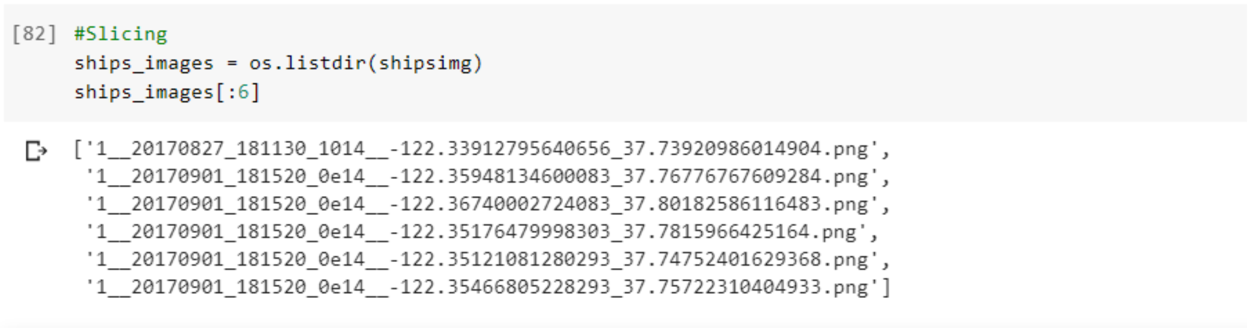
5.1 Statistical Analysis

Statistical analysis is the process of generating statistics from stored data and analyzing the results to deduce or infer meaning about the underlying dataset or the reality that it attempts to describe.

Statistics is defined as "…the study of the collection, analysis, interpretation, presentation, and organization of data." That's basically the same as the definition of data science, and in fact the term data science was initially coined in 2001 by Purdue statistician William S. Cleveland in the title to his paper "Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics.".

5.2 Visualizing the data

In this data set there are 2 folders namely 'scenes' and 'shipsnet', for visualizing the data we are using these 2 folders and selecting random images of ships, from the 'shipsnet' folder which has 4000 images taken from satellite. These images contain images of ships and other object present at a port. The 'scenes' folder contain 8 images which are the images of port captured using satellite.

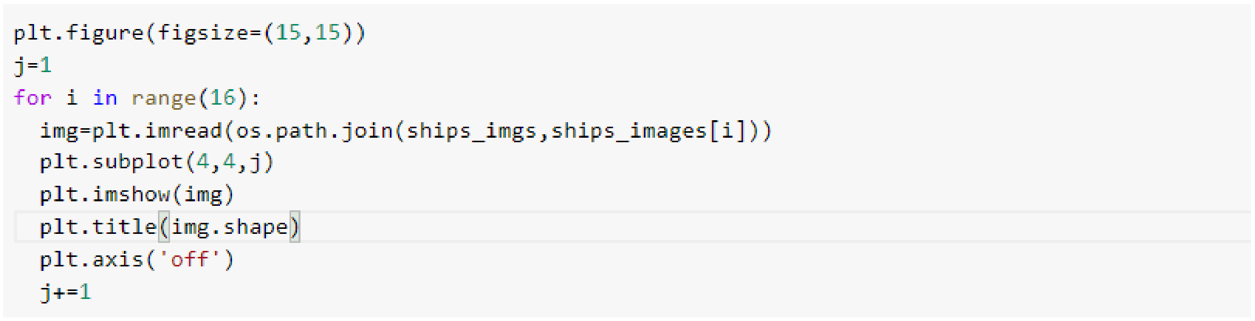


```
[82] #Slicing
     ships_images = os.listdir(shipsimg)
     ships_images[:6]
```

```
['1__20170827_181130_1014__-122.33912795640656_37.73920986014904.png',
 '1__20170901_181520_0e14__-122.35948134600083_37.76776767609284.png',
 '1__20170901_181520_0e14__-122.36740002724083_37.80182586116483.png',
 '1__20170901_181520_0e14__-122.35176479998303_37.7815966425164.png',
 '1__20170901_181520_0e14__-122.35121081280293_37.74752401629368.png',
 '1__20170901_181520_0e14__-122.35466805228293_37.75722310404933.png']
```

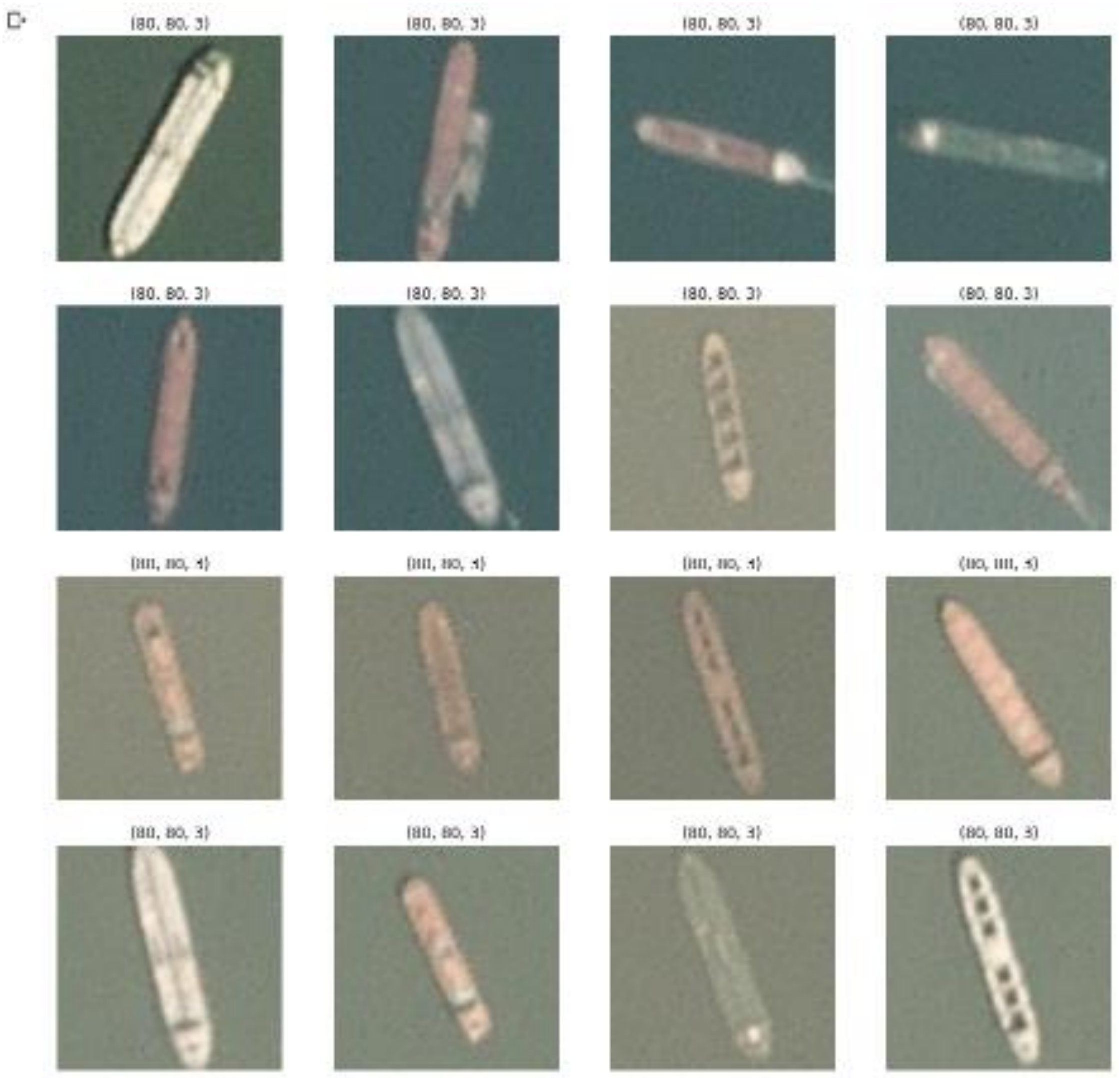
In the above code snippet, random files are being selected using the data slicing technique.

Ships_images [:6] implies that 6 random images of total 4000 images are being selected.

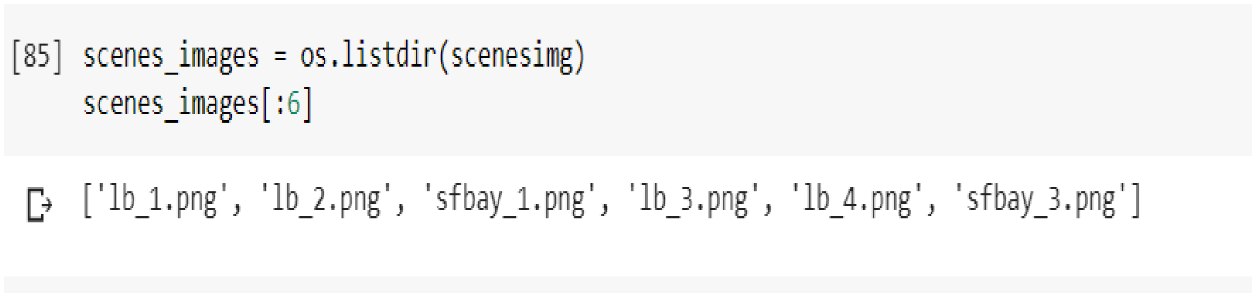


```
plt.figure(figsize=(15,15))
j=1
for i in range(16):
  img=plt.imread(os.path.join(ships_imgs,ships_images[i]))
  plt.subplot(4,4,j)
  plt.imshow(img)
  plt.title(img.shape)
  plt.axis('off')
  j+=1
```

Using the above code random 16 images of ships are selected from 4000 images and the shape of the images are being displayed as output.



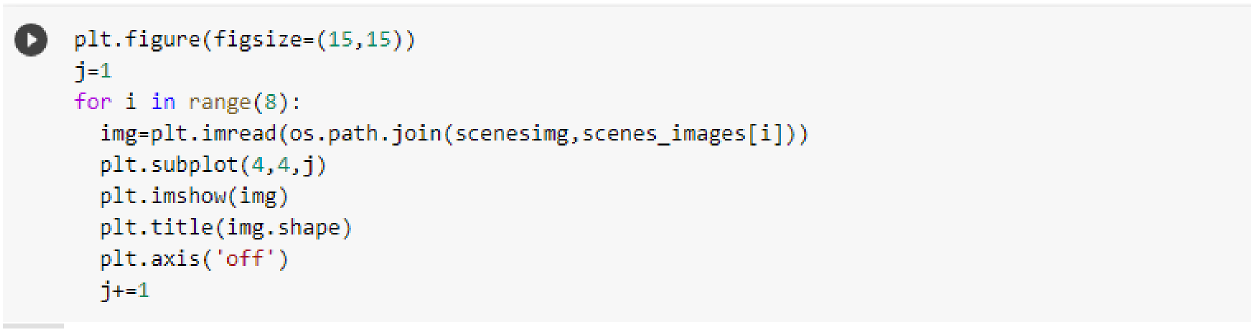
Similarly, for the scenes folder data slicing method is applied where randomly 6 images of port are being selected.



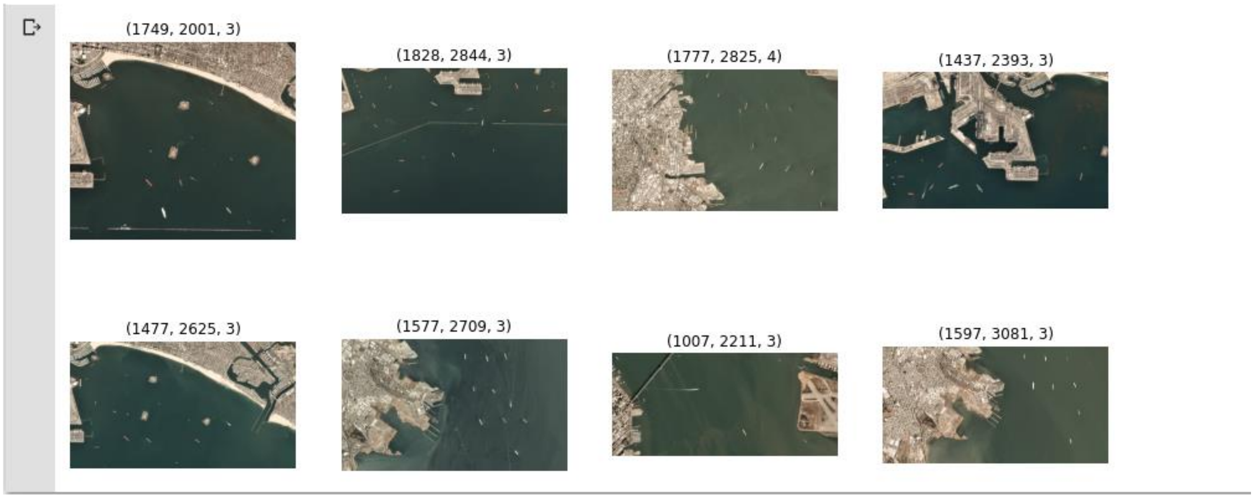
```
[85] scenes_images = os.listdir(scenesimg)
     scenes_images[:6]
```

```
['lb_1.png', 'lb_2.png', 'sfbay_1.png', 'lb_3.png', 'lb_4.png', 'sfbay_3.png']
```

After selecting random images, using the code below 8 images present in the data set are being displayed and the images shape.



```
plt.figure(figsize=(15,15))
j=1
for i in range(8):
  img=plt.imread(os.path.join(scenesimg,scenes_images[i]))
  plt.subplot(4,4,j)
  plt.imshow(img)
  plt.title(img.shape)
  plt.axis('off')
  j+=1
```



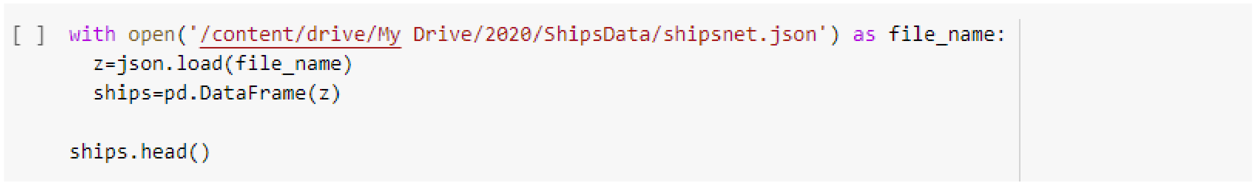
```
(1749, 2001, 3)    (1828, 2844, 3)    (1777, 2825, 4)    (1437, 2393, 3)
(1477, 2625, 3)    (1577, 2709, 3)    (1007, 2211, 3)    (1597, 3081, 3)
```

5.3 Data Type Conversions

The Data set contains 2 folders and a file of. json format which is ‘shipsnet. json’ so, **JSON** stands for JavaScript Object Notation. **JSON** is a lightweight format for storing and transporting **data**. **JSON** is often used when **data** is sent from a server to a web page. **JSON** is "self-describing" and easy to understand.

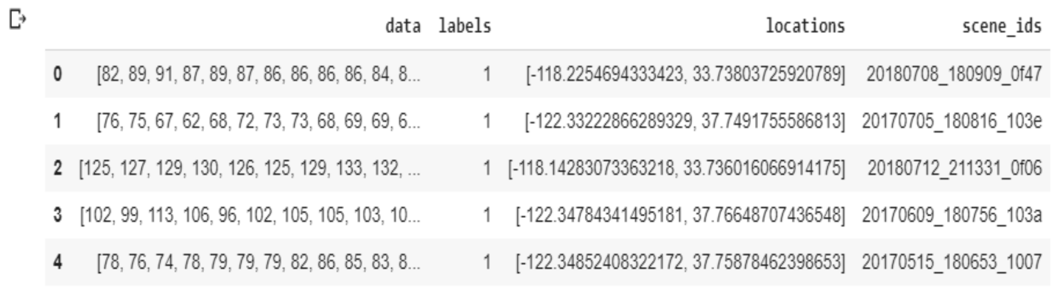
JSON format is **used** for serializing and transmitting structured data over network connection. It is primarily **used** to transmit data between a server and web applications. Web services and APIs use **JSON** format to provide public data.

Reading .JSON file:



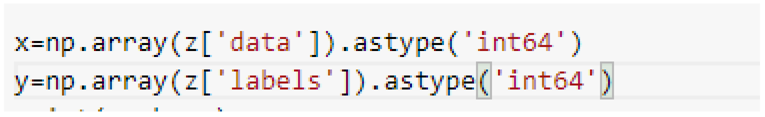
```
[ ]  with open('/content/drive/My Drive/2020/ShipsData/shipsnet.json') as file_name:
       z=json.load(file_name)
       ships=pd.DataFrame(z)

     ships.head()
```

Using the open('file location') command the .json file data is being opened and by using the load function the file is being loaded and data is formed as a data frame and stored in a variable using the .head() the top 5 values of the data frame are being extracted.



	data	labels	locations	scene_ids
0	[82, 89, 91, 87, 89, 87, 86, 86, 86, 86, 84, 8...	1	[-118.2254694333423, 33.73803725920789]	20180708_180909_0f47
1	[76, 75, 67, 62, 68, 72, 73, 73, 68, 69, 69, 6...	1	[-122.33222866289329, 37.7491755586813]	20170705_180816_103e
2	[125, 127, 129, 130, 126, 125, 129, 133, 132, ...	1	[-118.14283073363218, 33.736016066914175]	20180712_211331_0f06
3	[102, 99, 113, 106, 96, 102, 105, 105, 103, 10...	1	[-122.34784341495181, 37.76648707436548]	20170609_180756_103a
4	[78, 76, 74, 78, 79, 79, 79, 82, 86, 85, 83, 8...	1	[-122.34852408322172, 37.75878462398653]	20170515_180653_1007

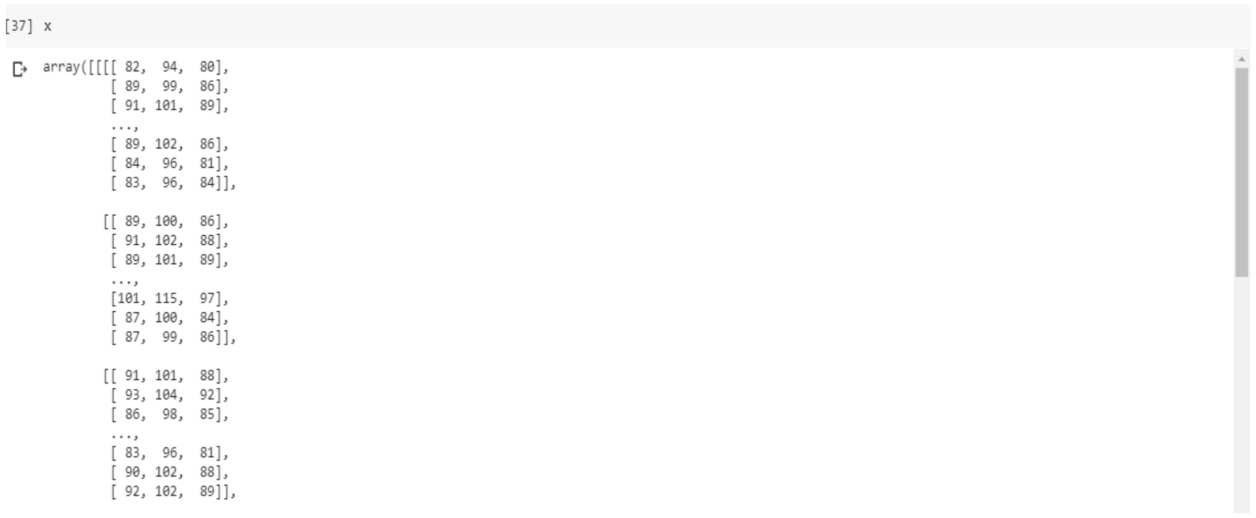
The data in json format is converted in the form of array using the np.array(z['data']).astype('int64') command. By this it would be simpler for the machine to read the data in form of array.



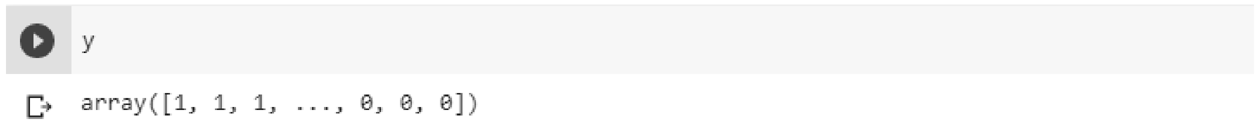
```
x=np.array(z['data']).astype('int64')
y=np.array(z['labels']).astype('int64')
```

Printing the x and y:

Now the x and values can be printed by just typing x and y on the command line.

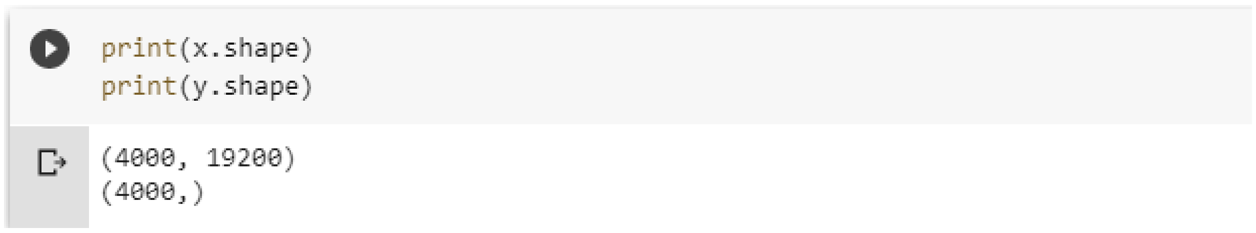


```
[37] x

array([[[[ 82,  94,  80],
         [ 89,  99,  86],
         [ 91, 101,  89],
         ...,
         [ 89, 102,  86],
         [ 84,  96,  81],
         [ 83,  96,  84]],

        [[ 89, 100,  86],
         [ 91, 102,  88],
         [ 89, 101,  89],
         ...,
         [101, 115,  97],
         [ 87, 100,  84],
         [ 87,  99,  86]],

        [[ 91, 101,  88],
         [ 93, 104,  92],
         [ 86,  98,  85],
         ...,
         [ 83,  96,  81],
         [ 90, 102,  88],
         [ 92, 102,  89]],
```



```
y

array([1, 1, 1, ..., 0, 0, 0])
```

Printing the shape:



```
print(x.shape)
print(y.shape)

(4000, 19200)
(4000,)
```

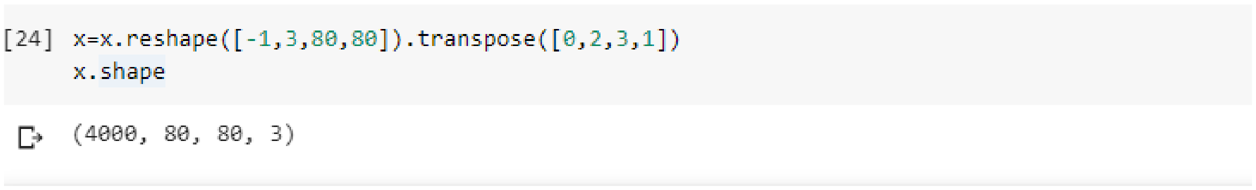
It would be difficult for a programmer to count the number of values present in a huge data set so by using ‘shape’ a user can get to know about the data set files.

5.4 Handling Missing Values

Since the data set consists of images, unlike other datasets like categorical datasets this dataset doesn’t have missing values, this dataset consists of image chips extracted from Planet satellite imagery collected over the San Francisco Bay and San Pedro Bay areas of California. It includes 4000 80x80 RGB images labeled with either a "ship" or "no-ship" classification. So, the Handing missing values isn’t used here.

5.5 Data Reshaping

The **reshape ()** function is used to give a new shape to an array without changing its data. Array to be **reshaped**. The new shape should be compatible with the original shape. If an integer, then the result will be a 1-D array of that length. The transpose method returns a transposed view of the passed multi-dimensional matrix.



```
[24] x=x.reshape([-1,3,80,80]).transpose([0,2,3,1])
     x.shape

     (4000, 80, 80, 3)
```

6.MODEL BUILDING AND EVALUATION

6.1 Brief about the algorithms used

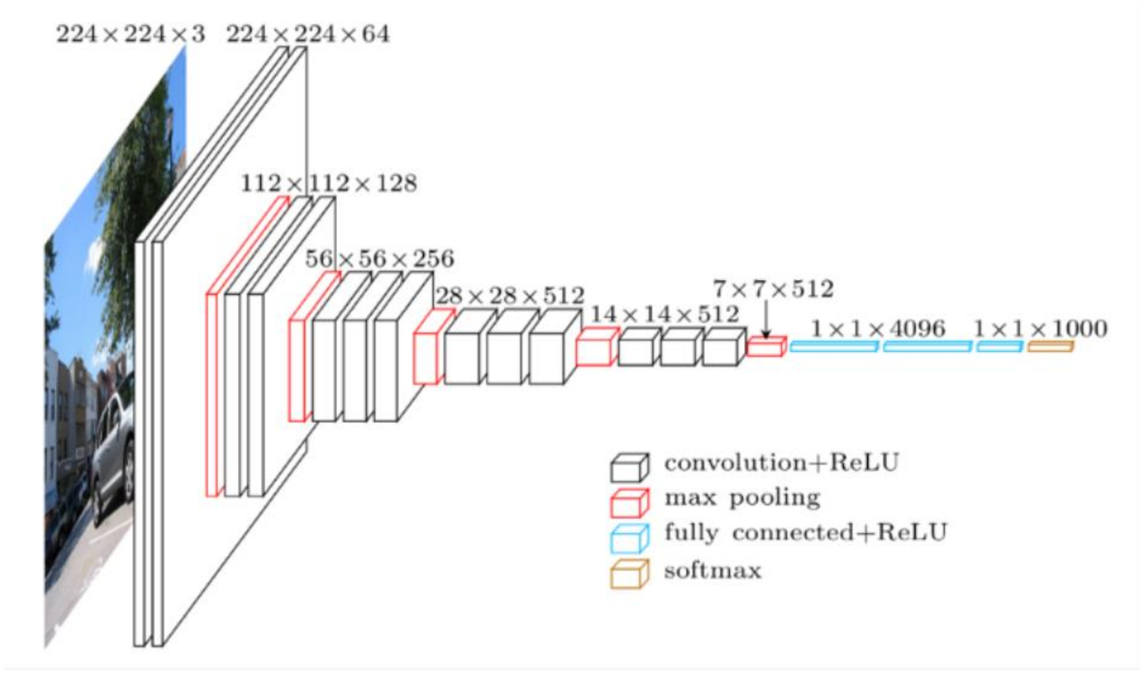
What is CNN?

Computer vision is evolving rapidly day-by-day. It's one of the reasons is deep learning. When we talk about computer vision, a term convolutional neural network (abbreviated as CNN) comes in our mind because CNN is heavily used here. Examples of CNN in computer vision are face recognition, image classification etc. It is similar to the basic neural network. CNN also have learnable parameter like neural network i.e., weights, biases etc.

Why should we use CNN?

Problem with Feedforward Neural Network

Suppose you are working with MNIST dataset, you know each image in MNIST is 28 x 28 x 1(black & white image contains only 1 channel). Total number of neurons in input layer will 28 x 28 = 784, this can be manageable. What if the size of image is 1000 x 1000 which means you need 10^6 neurons in input layer? This seems a huge number of neurons are required for operation. It is computationally ineffective right. So here comes Convolutional Neural Network or CNN. In simple word what CNN does is, it extracts the feature of image and convert it into lower dimension without losing its characteristics. In the following example you can see that initial the size of the image is 224 x 224 x 3. If you proceed without convolution then you need 224 x 224 x 3 = 100, 352 numbers of neurons in input layer but after applying convolution you input tensor dimension is reduced to 1 x 1 x 1000. It means you only need 1000 neurons in first layer of feedforward neural network.



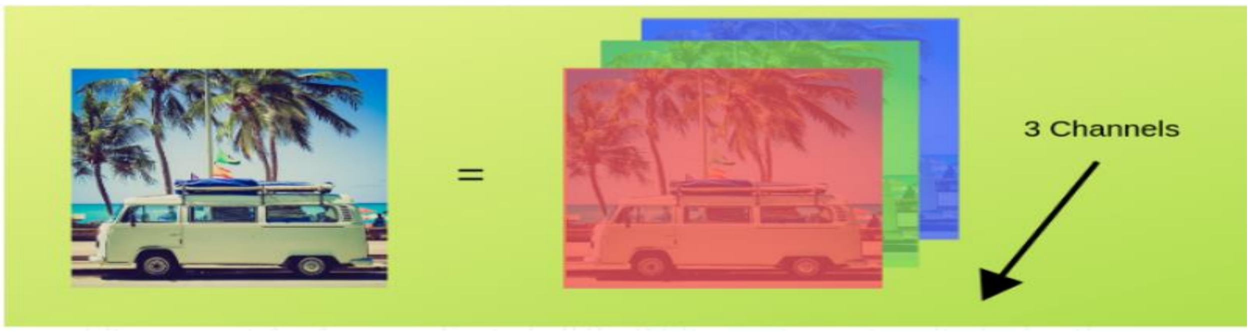
Few Definitions

There are few definitions you should know before understanding CNN

Image Representation

Thinking about images, it's easy to understand that it has a height and width, so it would make sense to represent the information contained in it with a two dimensional structure (a matrix) until you remember that images have colors, and to add information about the colors, we need another dimension, and that is when Tensors become particularly helpful.

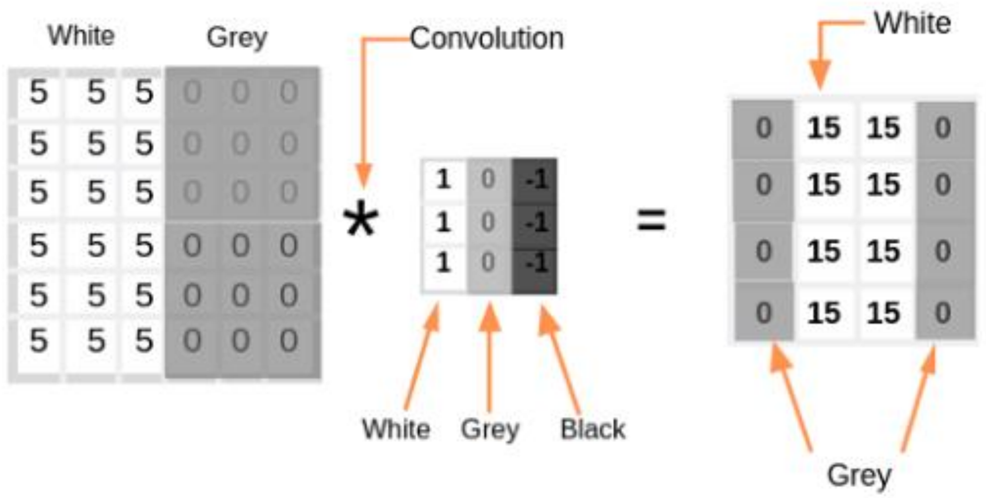
Images are encoded into color channels, the image data is represented into each color intensity in a color channel at a given point, the most common one being RGB, which means Red, Blue and Green. The information contained into an image is the intensity of each channel color into the width and height of the image, just like this



So, the intensity of the red channel at each point with width and height can be represented into a matrix, the same goes for the blue and green channels, so we end up having three matrices, and when these are combined, they form a tensor.

Edge Detection

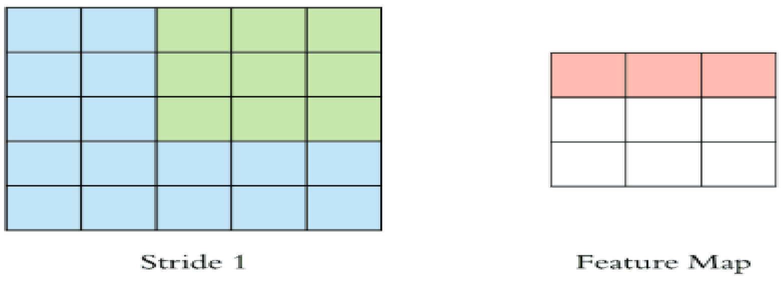
Every image has vertical and horizontal edges which actually combining to form a image. Convolution operation is used with some filters for detecting edges. Suppose you have gray scale image with dimension 6 x 6 and filter of dimension 3 x 3(say). When 6 x 6 grey scale image convolve with 3 x 3 filter, we get 4 x 4 image. First of all, 3 x 3 filter matrix get multiplied with first 3 x 3 size of our grey scale image, then we shift one column right up to end, after that we shift one row and so on.



Convolution operation

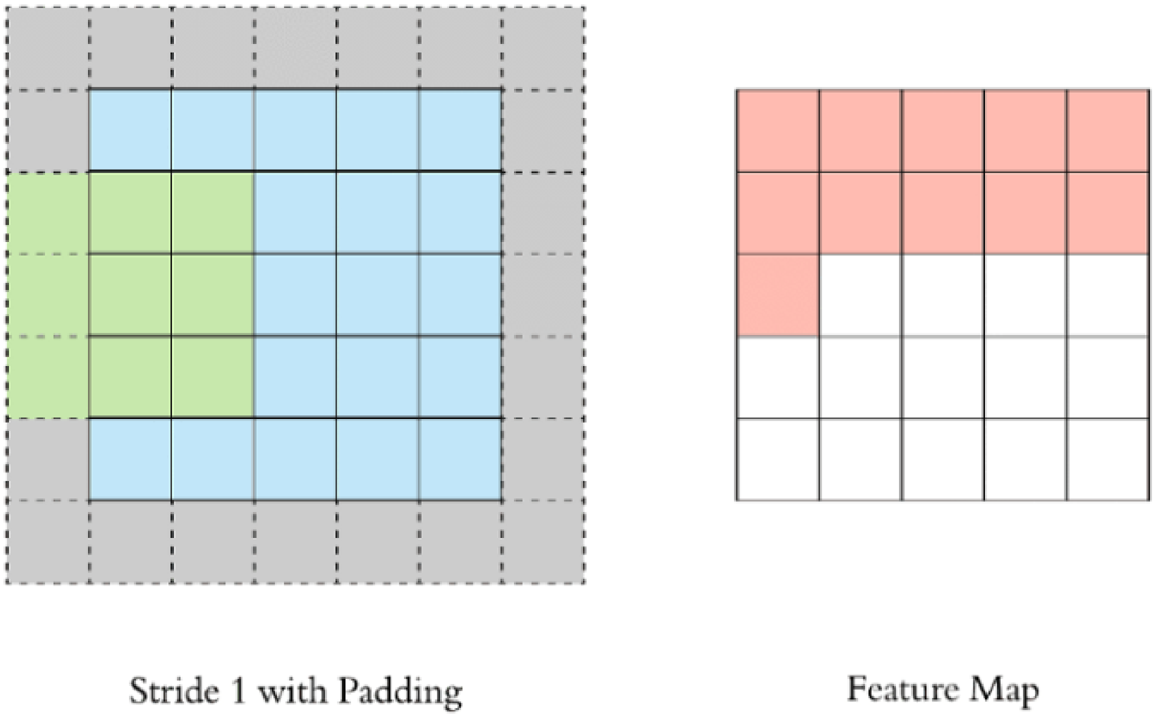
Stride and Padding

Stride denotes how many steps we are moving in each step-in convolution. By default, it is one



Convolution with Stride 1

We can observe that the size of output is smaller than input. To maintain the dimension of output as in input, we use padding. Padding is a process of adding zeros to the input matrix symmetrically. In the following example, the extra grey blocks denote the padding. It is used to make the dimension of output same as input

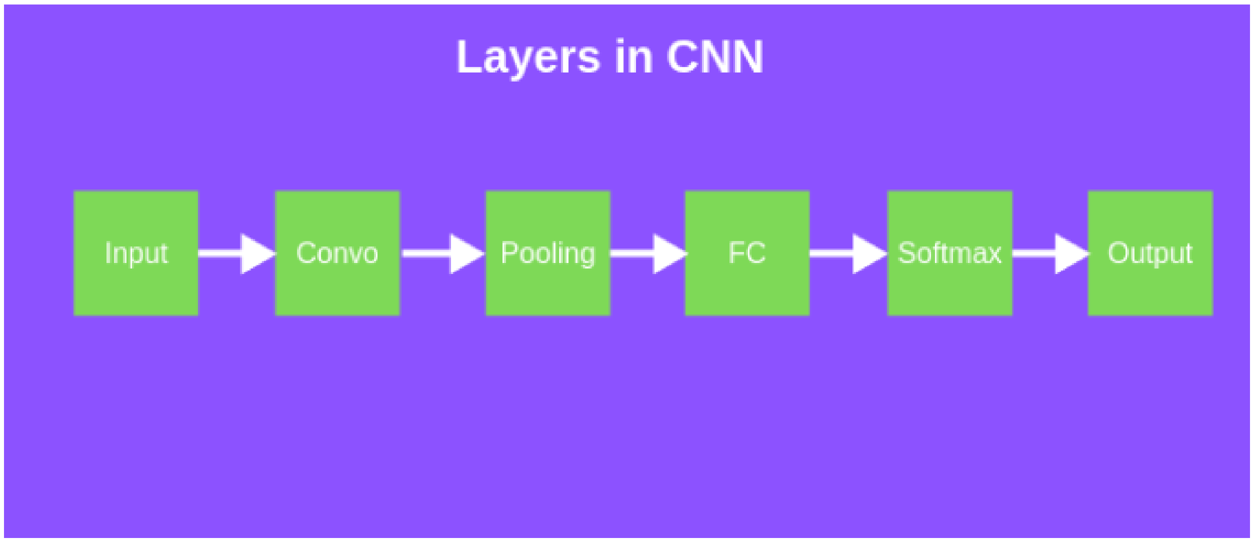


Stride 1 with Padding 1

Layers in CNN

There are five different layers in CNN

- Input layer
- Convo layer (Convo + ReLU)
- Pooling layer
- Fully connected (FC) layer
- Softmax/logistic layer
- Output layer



Different layers of CNN

Input Layer

Input layer in CNN should contain image data. Image data is represented by three-dimensional matrix as we saw earlier. You need to reshape it into a single column. Suppose you have image of dimension 28 x 28 =784, you need to convert it into 784 x 1 before feeding into input. If you have "m" training examples then dimension of input will be (784, m).

Convo Layer

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then we slide the filter over the next receptive field of the same input image by a Stride and do the same operation again. We will repeat the same process again and again until we go through the whole image. The output will be the input for the next layer.

Convo layer also contains ReLU activation to make all negative value to zero.

Pooling Layer

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layers. If we apply FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive and we don't want it. So, the max pooling is only way to reduce the spatial volume of input image. In the above example, we have applied max pooling in single depth slice with Stride of 2. You can observe the 4 x 4 dimension input is reduce to 2 x 2 dimension.

There is no parameter in pooling layer but it has two hyperparameters — Filter(F) and Stride(S).

In general, if we have input dimension W1 x H1 x D1, then

$W2 = (W1-F)/S+1$

$H2 = (H1-F)/S+1$

$D2 = D1$

Where W2, H2 and D2 are the width, height and depth of output.

Fully Connected Layer (FC)

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different category by training.

Softmax / Logistic Layer

Softmax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and Softmax is for multi-classification.

Output Layer

Output layer contains the label which is in the form of one-hot encoded.

Now you have a good understanding of CNN. Let's implement a CNN in Keras.

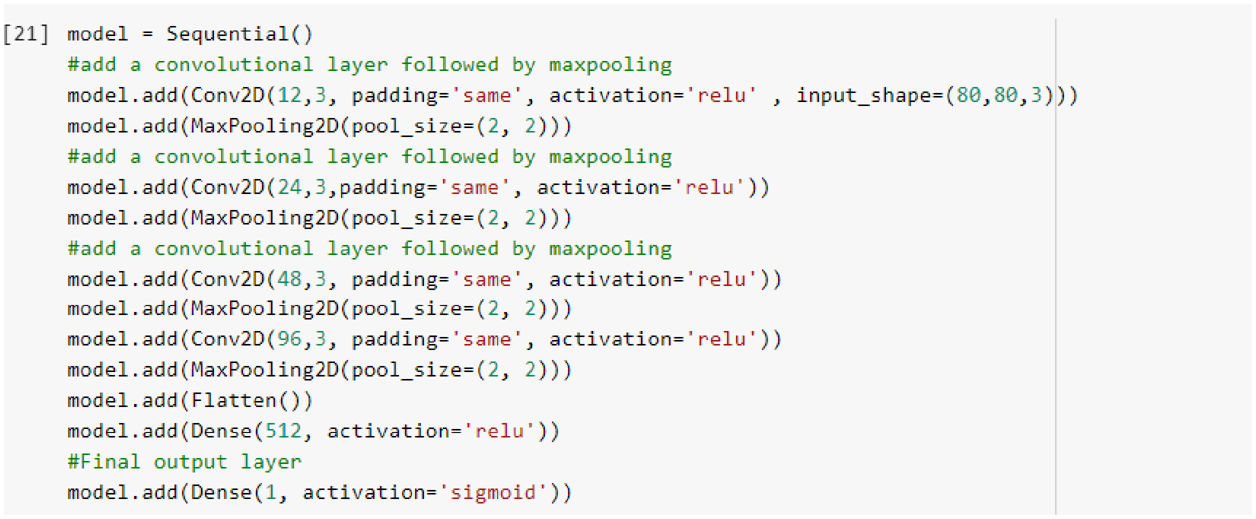
6.2 Building the Model

There is nothing better than deploying the model in a real-time environment. It helps us to gain analytical insights into the decision-making procedure. You constantly need to update the model with additional features for customer satisfaction.

To predict business decisions, plan market strategies, and create personalized customer interests, we integrate the machine learning model into the existing production domain.

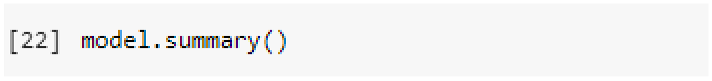
For example, when you go through the Amazon website and notice the product recommendations completely based on your curiosities. Same as Amazon, Netflix gives you the movie's suggestion based on your watching history and several interests. You can experience the increase in the involvement of the customers utilizing these services. That's how a deployed model changes the mindset of the customer and convince him to purchase the product.

Building Model for ships dataset

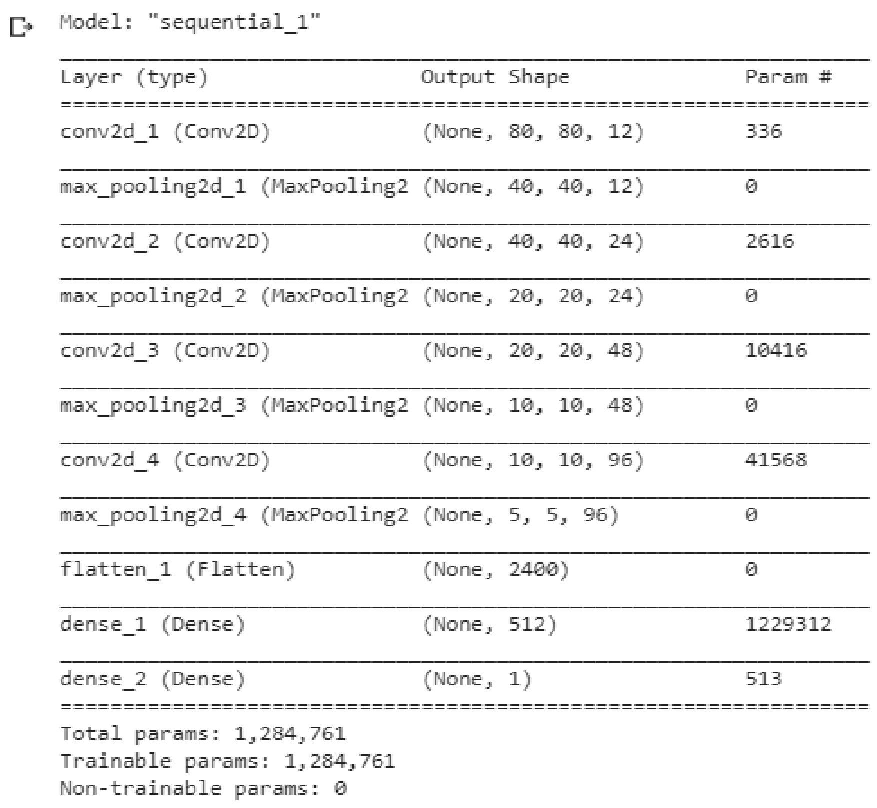


```
[21] model = Sequential()
     #add a convolutional layer followed by maxpooling
     model.add(Conv2D(12,3, padding='same', activation='relu' , input_shape=(80,80,3)))
     model.add(MaxPooling2D(pool_size=(2, 2)))
     #add a convolutional layer followed by maxpooling
     model.add(Conv2D(24,3,padding='same', activation='relu'))
     model.add(MaxPooling2D(pool_size=(2, 2)))
     #add a convolutional layer followed by maxpooling
     model.add(Conv2D(48,3, padding='same', activation='relu'))
     model.add(MaxPooling2D(pool_size=(2, 2)))
     model.add(Conv2D(96,3, padding='same', activation='relu'))
     model.add(MaxPooling2D(pool_size=(2, 2)))
     model.add(Flatten())
     model.add(Dense(512, activation='relu'))
     #Final output layer
     model.add(Dense(1, activation='sigmoid'))
```

By using the summary () function we can get the data model summary.



```
[22] model.summary()
```



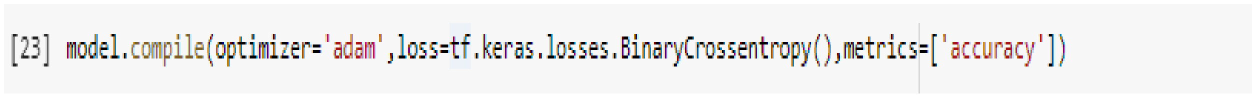
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 80, 80, 12)        336       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 40, 40, 12)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 40, 40, 24)        2616      
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 20, 20, 24)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 20, 20, 48)        10416     
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 10, 10, 48)        0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 10, 10, 96)        41568     
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 96)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 2400)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1229312   
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 513       
=================================================================
Total params: 1,284,761
Trainable params: 1,284,761
Non-trainable params: 0
```

A total of 1,284,761 parameters are present in which 1,284,761 are trainable and 0 are non-trainable.

6.3 Compiling the Model:

To compile the model, you need to specify the optimizer and loss function to use. In the video, Dan mentioned that the Adam optimizer is an excellent choice. You can read more about it as well as other keras optimizers here, and if you are really curious to learn more, you can read the original paper that introduced the Adam optimizer.

For the model designed using the dataset we are compiling the model using ‘adam’ as optimizer and ‘BinaryCrossentropy ()’ as loss value and the metrics is set to ‘accuracy’.



```
[23] model.compile(optimizer='adam',loss=tf.keras.losses.BinaryCrossentropy(),metrics=['accuracy'])
```

Model groups layers into an object with training and inference features.

Arguments

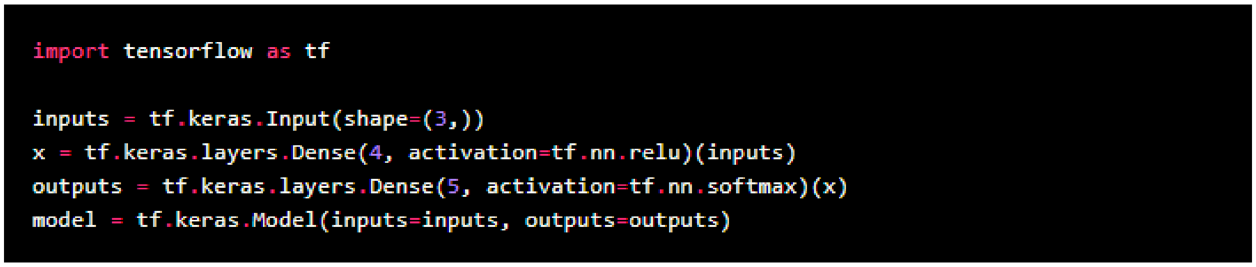
inputs: The input(s) of the model: a keras.Input object or list of keras.Input objects.

outputs: The output(s) of the model. See Functional API example below.

name: String, the name of the model.

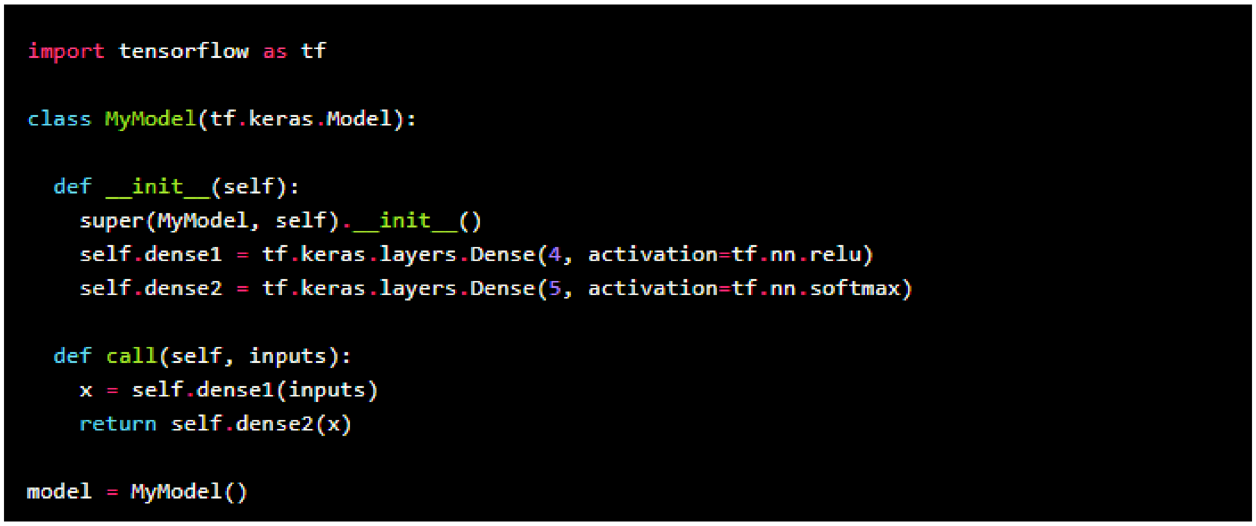
There are two ways to instantiate a Model:

1 - With the "Functional API", where you start from Input, you chain layer calls to specify the model's forward pass, and finally you create your model from inputs and outputs.



```
import tensorflow as tf

inputs = tf.keras.Input(shape=(3,))
x = tf.keras.layers.Dense(4, activation=tf.nn.relu)(inputs)
outputs = tf.keras.layers.Dense(5, activation=tf.nn.softmax)(x)
model = tf.keras.Model(inputs=inputs, outputs=outputs)
```

1 - By subclassing the Model class: in that case, you should define your layers in __init__ and you should implement the model's forward pass in call.



```
import tensorflow as tf

class MyModel(tf.keras.Model):

  def __init__(self):
    super(MyModel, self).__init__()
    self.dense1 = tf.keras.layers.Dense(4, activation=tf.nn.relu)
    self.dense2 = tf.keras.layers.Dense(5, activation=tf.nn.softmax)

  def call(self, inputs):
    x = self.dense1(inputs)
    return self.dense2(x)

model = MyModel()
```

Once the model is created, you can config the model with losses and metrics with **model. compile ()**, train the model with **model.fit (),** or use the model to do prediction with **model. predict ().**

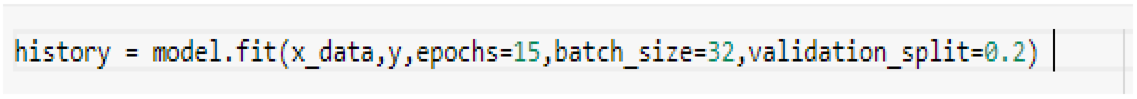
6.4 Fitting the data

Model fitting is a procedure that takes three steps:

First you need a function that takes in a set of parameters and returns a predicted data set.

Second you need an 'error function' that provides a number representing the difference between your data and the model's prediction for any given set of model parameters. This is usually either the sums of squared error (SSE) or maximum likelihood.

Third you need to find the parameters that minimize this difference. Once you set things up properly, this third step is easy thanks to the nerds at MathWorks.

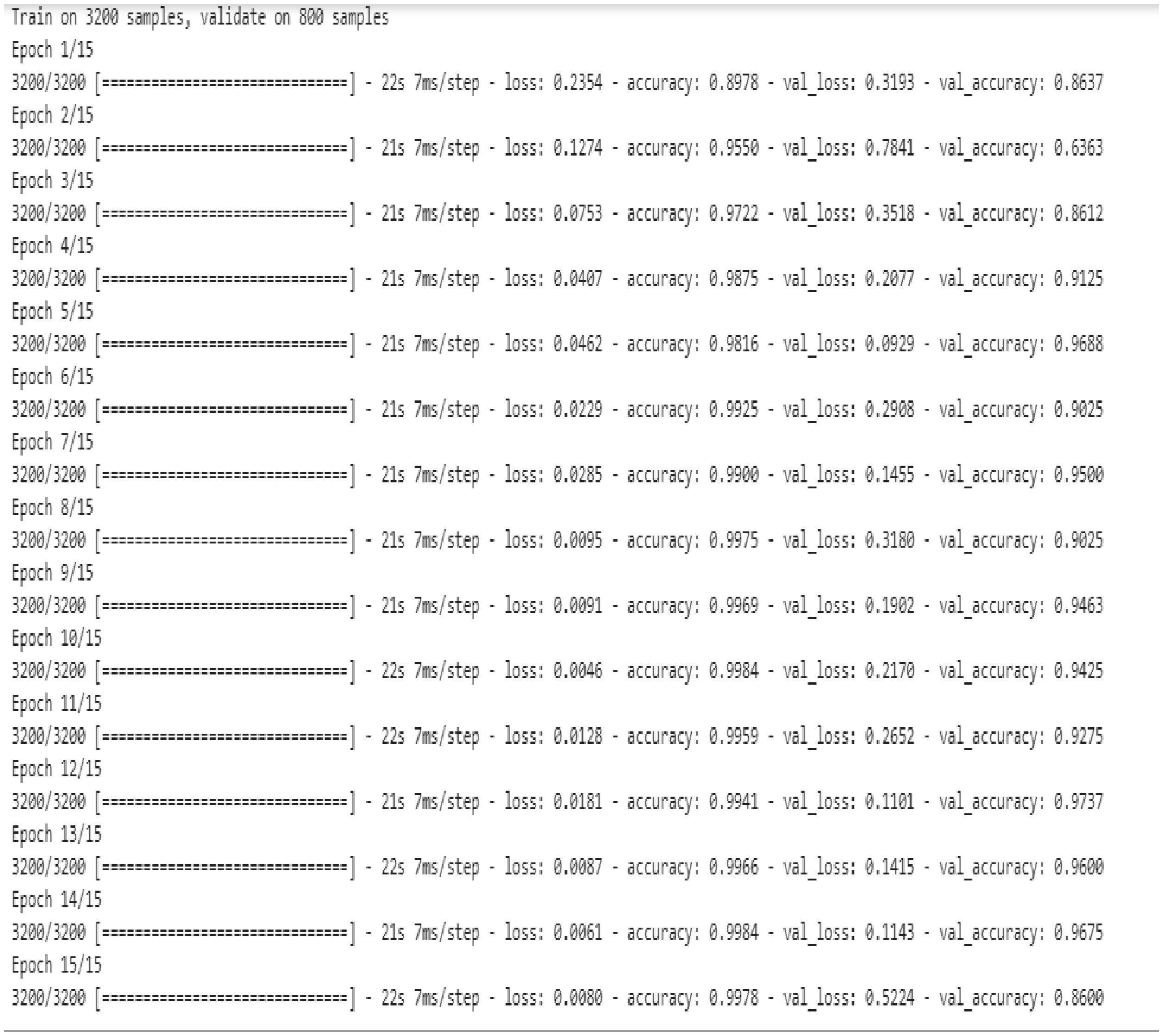


```
history = model.fit(x_data,y,epochs=15,batch_size=32,validation_split=0.2)
```

Here the model is being fitted using the scaled data x_data and y, the epochs, an epoch refers to one cycle through the full training dataset. Usually, training a neural network takes more than a few epochs.

Batch_size: batch_size denotes the subset size of your training sample (e.g. 100 out of 1000) which is going to be used in order to train the network during its learning process. Each batch trains network in a successive order, taking into account the updated weights coming from the appliance of the previous batch.

Validation_split: Where it refers to the percent of data to be divided as training and testing data.



```
Train on 3200 samples, validate on 800 samples
Epoch 1/15
3200/3200 [==============================] - 22s 7ms/step - loss: 0.2354 - accuracy: 0.8978 - val_loss: 0.3193 - val_accuracy: 0.8637
Epoch 2/15
3200/3200 [==============================] - 21s 7ms/step - loss: 0.1274 - accuracy: 0.9550 - val_loss: 0.7841 - val_accuracy: 0.6363
Epoch 3/15
3200/3200 [==============================] - 21s 7ms/step - loss: 0.0753 - accuracy: 0.9722 - val_loss: 0.3518 - val_accuracy: 0.8612
Epoch 4/15
3200/3200 [==============================] - 21s 7ms/step - loss: 0.0407 - accuracy: 0.9875 - val_loss: 0.2077 - val_accuracy: 0.9125
Epoch 5/15
3200/3200 [==============================] - 21s 7ms/step - loss: 0.0462 - accuracy: 0.9816 - val_loss: 0.0929 - val_accuracy: 0.9688
Epoch 6/15
3200/3200 [==============================] - 21s 7ms/step - loss: 0.0229 - accuracy: 0.9925 - val_loss: 0.2908 - val_accuracy: 0.9025
Epoch 7/15
3200/3200 [==============================] - 21s 7ms/step - loss: 0.0285 - accuracy: 0.9900 - val_loss: 0.1455 - val_accuracy: 0.9500
Epoch 8/15
3200/3200 [==============================] - 21s 7ms/step - loss: 0.0095 - accuracy: 0.9975 - val_loss: 0.3180 - val_accuracy: 0.9025
Epoch 9/15
3200/3200 [==============================] - 21s 7ms/step - loss: 0.0091 - accuracy: 0.9969 - val_loss: 0.1902 - val_accuracy: 0.9463
Epoch 10/15
3200/3200 [==============================] - 22s 7ms/step - loss: 0.0046 - accuracy: 0.9984 - val_loss: 0.2170 - val_accuracy: 0.9425
Epoch 11/15
3200/3200 [==============================] - 22s 7ms/step - loss: 0.0128 - accuracy: 0.9959 - val_loss: 0.2652 - val_accuracy: 0.9275
Epoch 12/15
3200/3200 [==============================] - 21s 7ms/step - loss: 0.0181 - accuracy: 0.9941 - val_loss: 0.1101 - val_accuracy: 0.9737
Epoch 13/15
3200/3200 [==============================] - 22s 7ms/step - loss: 0.0087 - accuracy: 0.9966 - val_loss: 0.1415 - val_accuracy: 0.9600
Epoch 14/15
3200/3200 [==============================] - 21s 7ms/step - loss: 0.0061 - accuracy: 0.9984 - val_loss: 0.1143 - val_accuracy: 0.9675
Epoch 15/15
3200/3200 [==============================] - 22s 7ms/step - loss: 0.0080 - accuracy: 0.9978 - val_loss: 0.5224 - val_accuracy: 0.8600
```

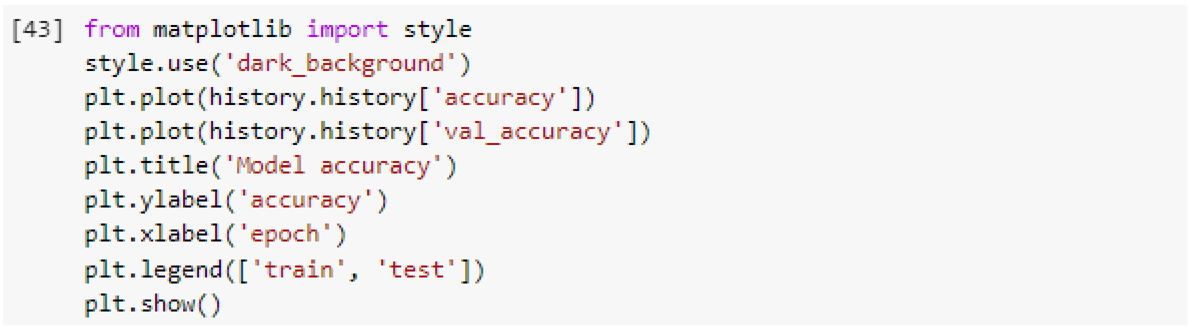
6.5 Model Accuracy

What does Machine Learning Model Accuracy Mean?

Machine learning model accuracy is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data. The better a model can generalize to ‘unseen’ data, the better predictions and insights it can produce, which in turn deliver more business value.

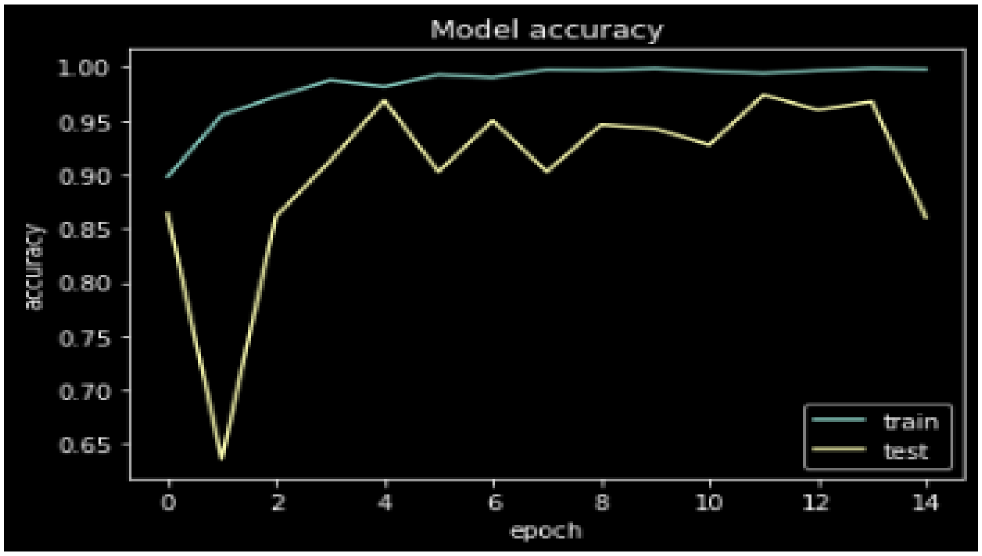
Why is Model Accuracy Important?

Machine learning models are used to make practical business decisions, and more accurate model outcomes result in better decisions. The cost of errors can be huge, but optimizing model accuracy mitigates that cost. There is, of course, a point of diminishing returns when the value of developing a more accurate model won't result in a corresponding profit increase, but often it is beneficial across the board. A false positive cancer diagnosis, for example, costs both the hospital and the patient. The benefits of improving model accuracy help avoid considerable time, money, and undue stress.

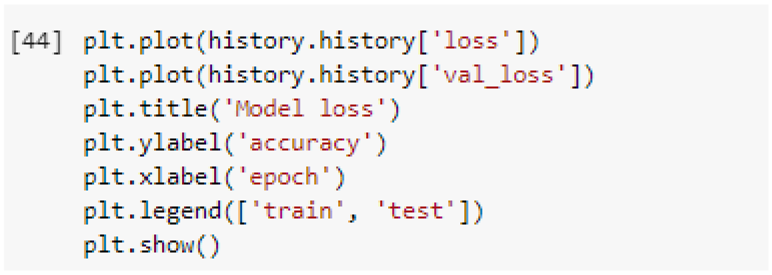


```
[43] from matplotlib import style
     style.use('dark_background')
     plt.plot(history.history['accuracy'])
     plt.plot(history.history['val_accuracy'])
     plt.title('Model accuracy')
     plt.ylabel('accuracy')
     plt.xlabel('epoch')
     plt.legend(['train', 'test'])
     plt.show()
```

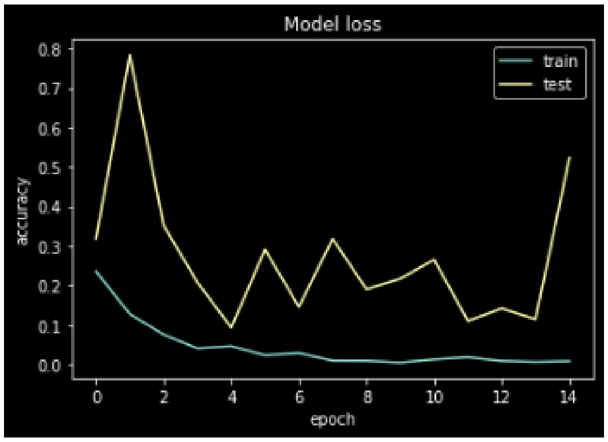
The Accuracy of the model is being calculated during the fitting of the data and using the plt.plot function a graph is plotted between the training and testing values obtained.



Another graph is also plotted to see the model loss value and the graph is plotted for the training and testing data loss values.

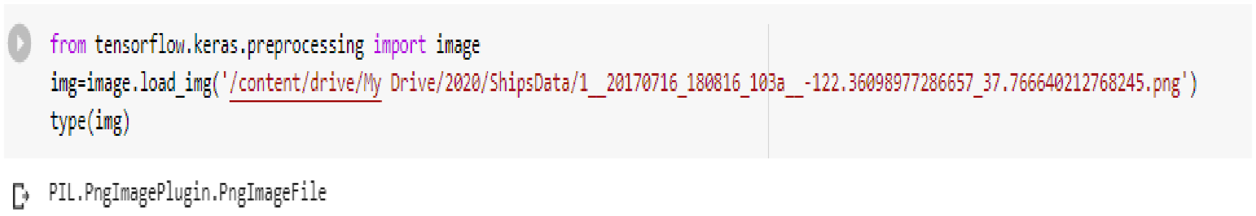


```
[44] plt.plot(history.history['loss'])
     plt.plot(history.history['val_loss'])
     plt.title('Model loss')
     plt.ylabel('accuracy')
     plt.xlabel('epoch')
     plt.legend(['train', 'test'])
     plt.show()
```



6.6 Image Prediction

The final step of the Project is to predict the images given by user weather the image is ship or not if the given image is of ship then model will return value near to 1 or 1 else it will return value closer or equal to 0.

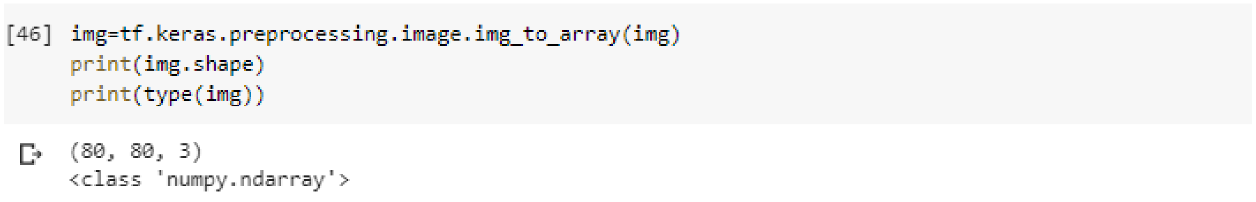


```
from tensorflow.keras.preprocessing import image
img=image.load_img('/content/drive/My Drive/2020/ShipsData/1__20170716_180816_103a__-122.36098977286657_37.766640212768245.png')
type(img)
```

```
PIL.PngImagePlugin.PngImageFile
```

Importing the image and printing the image type

The Image Given is of the type PIL.PngImagePlugin.PngImageFile or simply .png format.



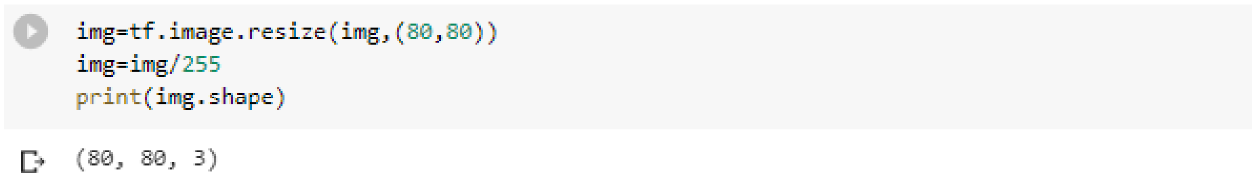
```
[46] img=tf.keras.preprocessing.image.img_to_array(img)
     print(img.shape)
     print(type(img))
```

```
(80, 80, 3)
<class 'numpy.ndarray'>
```

Now the image is being processed to an numpy.ndarray using the function tf.keras.preprocessing.image.img_to_array(img) and the image type and shape are being printed.

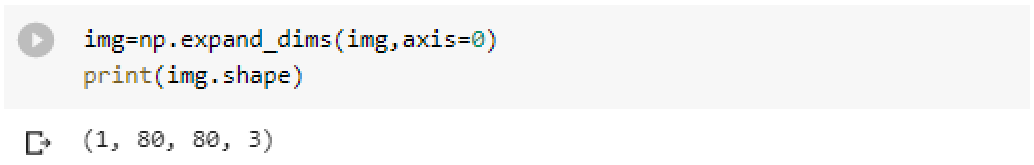
The image is of type `<class 'numpy.ndarray'>` and it is having shape of (80,80,3).

Now the image is being resized and scaled for the prediction.



```
img=tf.image.resize(img,(80,80))
img=img/255
print(img.shape)
```

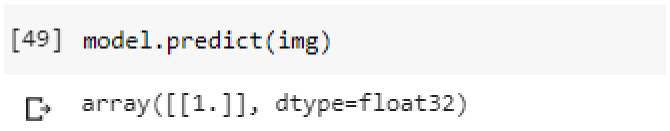
```
(80, 80, 3)
```



```
img=np.expand_dims(img,axis=0)
print(img.shape)
```

```
(1, 80, 80, 3)
```

After all the image processing steps the images is being predicted by calling the `predict`.



```
[49] model.predict(img)
```

```
array([[1.]], dtype=float32)
```

Clearly the output is 1 which implies that the given image is of a ship.

Image Used:

Conclusion

By this it is concluded that dataset which is being used is having images of ships, where a model is designed using the CNN algorithm and the model is able to predict weather the image given by the user is a ship or not a ship.

References

https://www.educba.com/uses-of-machine-learning/

https://www.datarobot.com/wiki/deep-learning/#:~:text=Why%20is%20Deep%20Learning%20Important,of%20data%20to%20be%20effective.

https://www.w3schools.com/python/python_intro.asp

https://www.geeksforgeeks.org/python-features/

https://www.programiz.com/python-programming/object-oriented-programming

https://keras.io/api/models/model/

https://scipy-cookbook.readthedocs.io/items/FittingData.html

https://deepai.org/machine-learning-glossary-and-terms

https://machinelearningmastery.com/how-to-connect-model-input-data-with-predictions-for-machine-learning/