

An Industrial Internship-I report submitted in third semester to CUSAT in partial fulfillment of  
the requirements for the award of degree

**BACHELOR OF TECHNOLOGY  
IN  
COMPUTER SCIENCE & ENGINEERING**

Submitted by

**RAVISANKAR S(20223565)**



**Division of Computer Science & Engineering**

**COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**COCHIN UNIVERSITY COLLEGE OF ENGINEERING KUTTANADU  
ALAPPUZHA**

**NOVEMBER 2024**

**COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY**  
**COCHIN UNIVERSITY COLLEGE OF ENGINEERING KUTTANADU**  
Pulincunnoo - 688504

**Division of Computer Science & Engineering**



**BONAFIDE CERTIFICATE**

This is to certify that the industrial internship certified is a bonafide report of industrial internship done by **RAVISANKAR S (20223565)** towards Industrial Internship-I report to be submitted in third semester of **B.Tech in Computer Science and Engineering** of **Cochin University of Science And Technology**.

**Mrs. Hafeesa M Habeeb**  
Assistant Professor  
Division Of CSE

**Dr. Preetha Mathew K**  
Head of Department  
Division Of CSE

# CERTIFICATE

Kerala State Electronics Development Corporation Ltd.  
(A Government of Kerala Undertaking)



**ITBG - Knowledge Services Group**

GSTIN : 32AABCK1319E4Z5

Phone: 0471-2724765 || Helpline No: +91 9188665545 || E-mail: iteg@keltron.org || Website Link : ksg.keltron.in

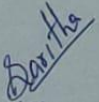
KSG/42/INTP/24-06/L0487

03.06.2024

## CERTIFICATE

Certified that the Student has completed the **Internship Program** conducted at KELTRON as per the particulars given below:

Name of the Student	Mr. RAVISANKAR S
Name & Address of the College/Institution at which the Student is Studying	Cochin University College of Engineering Kuttanad, Kidangara Mankombu Road, Kannady, Pullinkunnu, Alappuzha, Kerala - 688504
University/Board, which the Institution is Affiliated to	Cochin University of Science and Technology
Branch/Discipline of Study	B. Tech in Computer Science & Engineering
Student's Registration No.	20223565
Semester / Year of Study	2 <sup>nd</sup> Semester
Name & Address of the Institution/Centre where the Internship Program was conducted	Keltron Knowledge Centre (42), 3 <sup>rd</sup> Floor, MES Cultural Complex, Near Reserve Bank of India, Kaloor, Ernakulam - 682017, Kerala
Online Portal Registration No.	KI041978
Roll No./Register No.	K2442I120070
Program Category	INTERNSHIP
Period & Duration of the Program	09.05.2024 to 22.05.2024 (2 Weeks)
Area of Exposure	Data Science & AI: Python Fundamentals, Introduction to Machine Learning, Supervised Learning & Unsupervised Learning, Introduction to Artificial intelligence, Types of AI, Applications of AI, Introduction to NN, Introduction to NLP, Deep Learning, Introduction to Data Science.
Status	COMPLETED SUCCESSFULLY

  
Authorized Signatory  
(Knowledge Services)



Regd. Office : Keltron House, Vellayambalam, Piruvambathapuram, Kerala State, India. Pin.695 033, Tel: 0471-4094444

Doc. No: KSG/LH/12-23/018 (1500)

Visit us at our website : <http://www.keltron.org>

Temp.No.: KSG/INTP/Type- 1/2-Rev1.06.08.2023

## TABLE OF CONTENTS

Chapter No	Title	Page No
1	About Keltron	2
2	Internship Goals	3
3	Technologies learned	
	3.1 - Python	4
	3.2 – Data Science & AI	8
4	Project - I	15
5	Project - II	19
6	Conclusion	25

## **CHAPTER - 1**

### **ABOUT KELTRON**

Kerala State Electronics Development Corporation Limited is a company incorporated under the Companies Act, 1956. The company is fully owned by the Government of Kerala. KELTRON Group comprises the holding company, Kerala State Electronics Development Corporation Limited and ten subsidiary and associate companies, turning out more than hundred different types of products. These products are marketed through a strong sales and distribution network spanning the entire country, through the seven branch offices located at Ahmedabad, Bangalore, Calcutta, Chennai, Delhi, Hyderabad, and Mumbai, to ensure that the products are backed by dedicated support and service. Over the years KELTRON has accomplished much more than what is set out to achieve; to industrialize a virgin territory and to provide employment opportunities to the teeming millions. It pioneered new concepts in industry and management, and created many products for the first time in the country. KELTRON has not only changed the industrial profile of the State but has even transformed the lifestyles of its people. Trivandrum, the corporate head-quarters of KELTRON, once a placid town, is now a sophisticated city humming along with the highest in Quality of Life Index. More than offering employment opportunities to the people, Keltron has trained and nurtured many high caliber managers and technical hands who have in turn enlarged the horizons of a developmental process that it had initiated years ago.

Today, KELTRON products remain as icons of technology brought to benefit the people. As diverse as its products and their uses are, they are unified by its pivotal strength as a solutions provider. It is this philosophy that has made KELTRON a vital contributor to the changing needs of the world and the community to which it belongs, during the last quarter century. From within the four walls of your drawing room to the vast expanse of an international airport, from the deep oceans to the realms of outer space, there is a KELTRON product in the form of an innovative solution. Continuing in its quest to bring the benefits of frontier technology to its customers, KELTRON has forged strategic alliances with world leaders in the trade. Its focus today is in adapting technology to fulfill the needs of its customers with a renewal mission to emerge as a provider of better solutions for the future. Keltron has started Knowledge Centers as an initiative to support youth to develop skills in fields like IT, ITeS and Computer Education.



## **CHAPTER - 2**

### **INTERNSHIP GOALS**

- ❖ Gain a solid understanding of the fundamental concepts of Python.
- ❖ Become acquainted with employing Python to develop machine learning projects.
- ❖ Gain a foundational understanding of machine learning, AI and its algorithms.
- ❖ To develop expertise in deep learning and CNNs through practical experience with image classification using the MNIST dataset. We aim to become proficient in TensorFlow/Keras for building and training deep learning models.
- ❖ To implement a model to develop a spam detection model using advanced machine learning techniques. The goal is to build a real-time spam filter. Through this project, we aim to gain practical experience in NLP and model development.

## **CHAPTER - 3**

### **TECHNOLOGIES LEARNED**

#### **3.1 – PYTHON:**

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems.

#### **Object Oriented Programming Language:**

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A distinguishing feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OO programming, computer programs are designed by making them out of objects that interact with one another. There is significant diversity in object oriented programming, but most popular languages are class-based, meaning that objects are instances of classes, which typically also determines their type.

#### **Data Type:**

Data types determine whether an object can do something, or whether it just would not make sense. Other programming languages often determine whether an operation makes sense for an object by making sure the object can never be stored somewhere where the operation will be performed on the object (this type system is called static typing). Python does not do that. Instead it stores the type of an object with the object, and checks when the operation is performed whether that operation makes sense for that object (this is called dynamic typing).

Different types of data types are:

- ❖ Booleans are either True or False.
- ❖ Numbers can be integers (1 and 2), floats (1.1 and 1.2), fractions (1/2 and 2/3), or even complex numbers.
- ❖ Strings are sequences of Unicode characters, e.g. an HTML document.
- ❖ Bytes and byte arrays, e.g. a JPEG image file.
- ❖ Lists are ordered sequences of values.
- ❖ Tuples are ordered, immutable sequences of values.
- ❖ Sets are unordered bags of values.

## Variables:

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Ex: counter = 100 # An integer

## String:

In programming terms, we usually call text a string. When you think of a string as a collection of letters, the term makes sense.

Ex: point name = "John" # A string

- "hello"+"world"      "helloworld"      # concatenation
- "hello"\*3      "hellohellohello"      # repetition
- "hello"[0]      "h"      # indexing
- "hello"[-1]      "o"      # (from end)
- "hello"[1:4]      "ell"      # slicing
- len("hello")      5      # size
- "hello" < "jello"      1      # comparison

## Operators:

Operator	Meaning	Example
+	Add two operands or unary plus	x + y +2
-	Subtract right operand from the left or unary minus	x - y -2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)



## Tuples:

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses. To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index.

For example – `tup1 = ('physics', 'chemistry', 1997, 2000); tup2 = (1, 2, 3, 4, 5, 6, 7); print "tup1[0]: ", tup1[0] print "tup2[1:5]: ", tup2[1:5]`

When the above code is executed, it produces the following result – `tup1[0]: physics tup2[1:5]: [2, 3, 4, 5]`

Tuples respond to the + and \* operators much like strings; they mean concatenation and repetition here too, except that the result is a new tuple, not a string.

## List:

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type. Creating a list is as simple as putting different comma-separated values between square brackets.

For example – `list1 = ['physics', 'chemistry', 1997, 2000]; list2 = [1, 2, 3, 4, 5 ]; list3 = ["a", "b", "c", "d"];`

Python Expression	Results	Description
<code>len([1, 2, 3])</code>	3	Length
<code>[1, 2, 3] + [4, 5, 6]</code>	<code>[1, 2, 3, 4, 5, 6]</code>	Concatenation
<code>['Hi!'] * 4</code>	<code>['Hi!', 'Hi!', 'Hi!', 'Hi!']</code>	Repetition
<code>3 in [1, 2, 3]</code>	True	Membership
<code>for x in [1, 2, 3]: print x,</code>	1 2 3	Iteration

## Loop:

Programming languages provide various control structures that allow for more complicated execution paths. A loop statement allows us to execute a statement or group of statements multiple times. Python programming language provides following types of loops to handle looping requirements:

- ❖ while loop: Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.

- ❖ for loop: Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
- ❖ nested loops: You can use one or more loop inside any another while, for or do..while loop.

## Conditional Statements:

Decision making is anticipating of conditions occurring while execution of the program and specifying actions taken according to the conditions. Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE otherwise. Python programming language provides following types of decision making statements.

- ❖ If statements: An if statement consists of a boolean expression followed by one or more statements.
- ❖ If...else statements: An if statement can be followed by an optional else statement, which executes when the boolean expression is FALSE.
- ❖ Nested if statements: You can use one if or else if statement inside another if or else if statement(s).

## Function:

Function blocks begin with the keyword `def` followed by the function name and parentheses ( ( ) ). Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses. The first statement of a function can be an optional statement - the documentation string of the function. The code block within every function starts with a colon (:) and is indented. The statement `return [expression]` exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as `return None`.

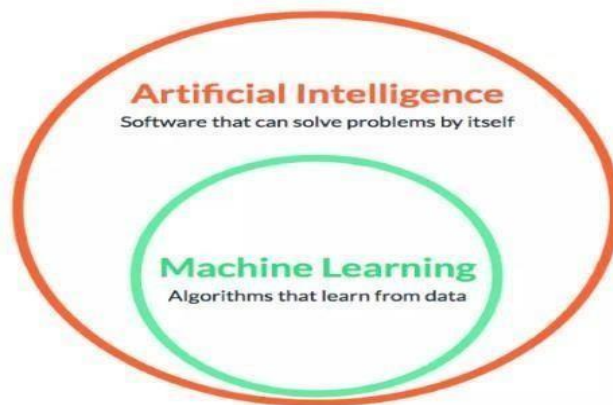
Syntax:

```
Def functionname(parameters):  
    "function_docstring"  
    Function_suite  
    Return[expression]
```

### 3.2 Data Science and Artificial Intelligence:

Artificial intelligence refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem- solving.

Relationship between AI and ML:



Machine Learning is an approach or subset of Artificial Intelligence that is based on the idea that machines can be given access to data along with the ability to learn from it.

#### Machine Learning:

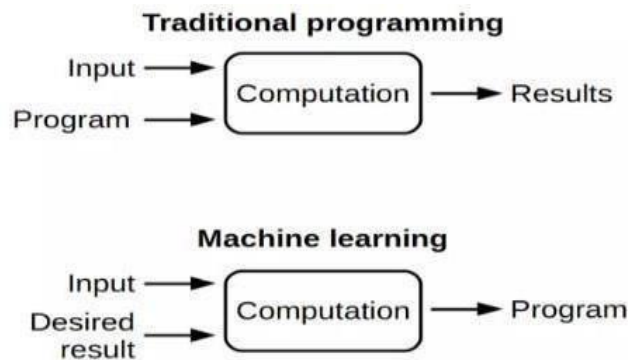
Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

Features of Machine Learning:

- ❖ Machine Learning is computing-intensive and generally requires a large amount of training data.
- ❖ It involves repetitive training to improve the learning and decision making of algorithms.
- ❖ As more data gets added, Machine Learning training can be automated for learning new data patterns and adapting its algorithm.

#### Traditional Programming Vs. Machine Learning Approach:

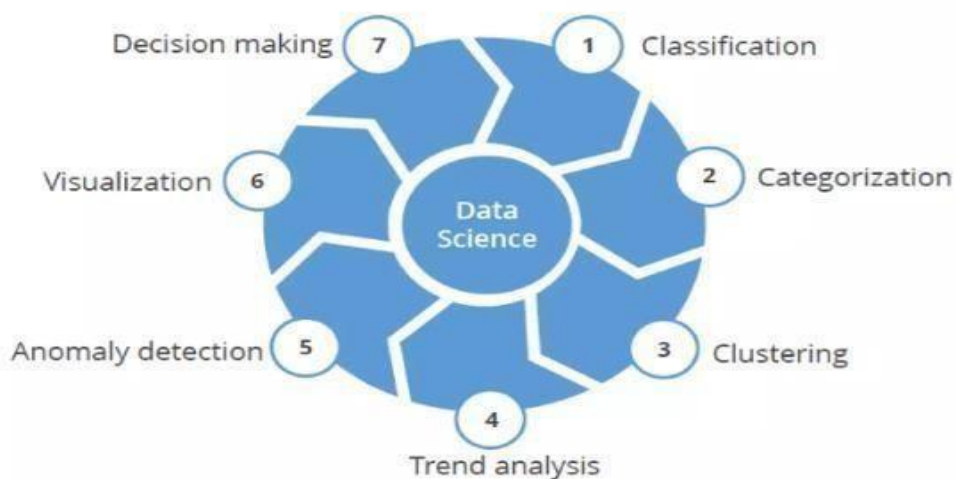
Traditional programming relies on hard-coded rules.



Machine Learning relies on learning patterns based on sample data.

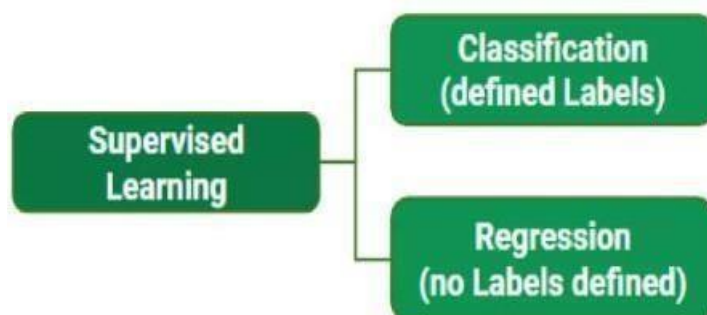
### Machine Learning Techniques:

Machine Learning uses a number of theories and techniques from Data Science.



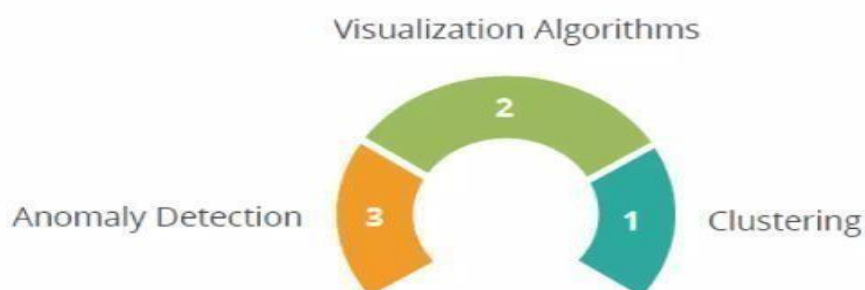
### Supervised Learning:

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples.



**Unsupervised Learning:**

Unsupervised learning is the training of machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data.

**Regression:**

In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modeling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors').

**1. Linear Regression:**

Linear regression is a linear approach for modeling the relationship between a scalar dependent variable  $y$  and an independent variable  $x$ , where  $x$ ,  $y$ ,  $w$  are vectors of real numbers and  $w$  is a vector of weight parameters. The equation is also written as:  $y = wx + b$ , where  $b$  is the bias or the value of output for zero input.

**2. Multiple Linear Regression:**

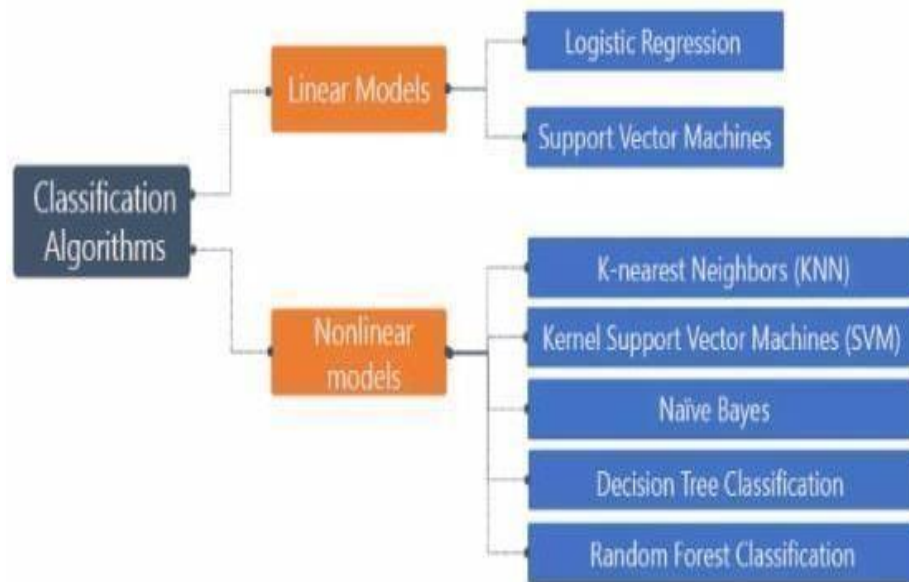
It is a statistical technique used to predict the outcome of a response variable through several explanatory variables and model the relationships between them.

**3. Polynomial Regression:**

Polynomial regression is applied when data is not formed in a straight line. It is used to fit a linear model to non-linear data by creating new features from powers of non-linear features

**Classification:**

Classification is a machine learning task that involves assigning input data elements to specific classes or categories. The primary objective of classification is to predict the class to which a given input variable belongs. This type of supervised learning is particularly well-suited for scenarios where the output consists of finite and discrete values. Classification can be broadly categorized into two types: binomial and multi-class.



### Linear Models:

#### 1. Logistic Regression:

This method is widely used for binary classification problems. It can also be extended to multi class classification problems. A binary dependent variable can have only two values, like 0 or 1, win or lose, pass or fail, healthy or sick, etc. The probability in the logistic regression is often represented by the Sigmoid function (also called the logistic function or the S-curve).

#### 2. Support Vector machines:

SVMs are very versatile and are also capable of performing linear or nonlinear classification, regression, and outlier detection. They involve detecting hyperplanes which segregate data into classes. The optimization objective is to find “maximum margin hyperplane” that is farthest from the closest points in the two classes (these points are called support vectors).

### Nonlinear Models:

#### 1. K-Nearest Neighbors (KNN):

K-nearest Neighbors algorithm is used to assign a data point to clusters based on similarity measurement. A new input point is classified in the category such that it has the greatest number of neighbors from that category.

#### 2. Naïve Bayes:

According to Bayes model, the conditional probability  $P(Y|X)$  can be calculated as:

$$P(Y|X) = P(X|Y) P(Y)$$

$$P(X)$$

This means you have to estimate a very large number of  $P(X|Y)$  probabilities for a relatively small vector space  $X$ .

#### 3. Decision Tree Classification:

The advantage of decision trees is that they require very little data preparation. They do not require feature scaling or centering at all. They are also the fundamental components of Random Forests, one of the most powerful ML algorithms. Start at the tree root and split the data on the feature using the decision algorithm, resulting in the largest information gain (IG).

## Clustering:

Clustering is a Machine Learning technique that involves the grouping of data points.

K-Means Clustering:

- ❖ K-means, a Prototype-based method, is the most popular method for clustering that involves:
- ❖ Training data that gets assigned to matching cluster based on similarity.
- ❖ Iterative process to get data points in the best clusters possible.

## Neural Networks:

A neural network is a machine learning program, or model, that makes decisions in a manner similar to the human brain, by using processes that mimic the way biological neurons work together to identify phenomena, weigh options and arrive at conclusions.

Every neural network consists of layers of nodes, or artificial neurons—an input layer, one or more hidden layers, and an output layer. Each node connects to others, and has its own associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

Neural networks rely on training data to learn and improve their accuracy over time. Once they are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity. Tasks in speech recognition or image recognition can take minutes versus hours when compared to the manual identification by human experts. One of the best-known examples of a neural network is Google's search algorithm.

Neural networks are sometimes called artificial neural networks (ANNs) or simulated neural networks (SNNs). They are a subset of machine learning, and at the heart of deep learning models

## Convolutional Neural Network (CNN):

A Convolutional Neural Network (CNN), also known as ConvNet, is a specialized type of deep learning algorithm mainly designed for tasks that necessitate object recognition, including image classification, detection, and segmentation.

### Key Components of a CNN

The convolutional neural network is made of four main parts.

But how do CNNs Learn with those parts?

They help the CNNs mimic how the human brain operates to recognize patterns and features in images:

- Convolutional layers
- Rectified Linear Unit (ReLU for short)
- Pooling layers
- Fully connected layers

**1. Convolution layers:**

This is the first building block of a CNN. As the name suggests, the main mathematical task performed is called convolution, which is the application of a sliding window function to a matrix of pixels representing an image. The sliding function applied to the matrix is called kernel or filter, and both can be used interchangeably.

In the convolution layer, several filters of equal size are applied, and each filter is used to recognize a specific pattern from the image, such as the curving of the digits, the edges, the whole shape of the digits, and more.

Put simply, in the convolution layer, we use small grids (called filters or kernels) that move over the image. Each small grid is like a mini magnifying glass that looks for specific patterns in the photo, like lines, curves, or shapes. As it moves across the photo, it creates a new grid that highlights where it found these patterns.

For example, one filter might be good at finding straight lines, another might find curves, and so on. By using several different filters, the CNN can get a good idea of all the different patterns that make up the image.

**2. Activation function:**

A ReLU activation function is applied after each convolution operation. This function helps the network learn non-linear relationships between the features in the image, hence making the network more robust for identifying different patterns. It also helps to mitigate the vanishing gradient problems.

**3. Pooling layer**

The goal of the pooling layer is to pull the most significant features from the convoluted matrix. This is done by applying some aggregation operations, which reduce the dimension of the feature map (convoluted matrix) reducing the memory used while training the network. Pooling is also relevant for mitigating overfitting.

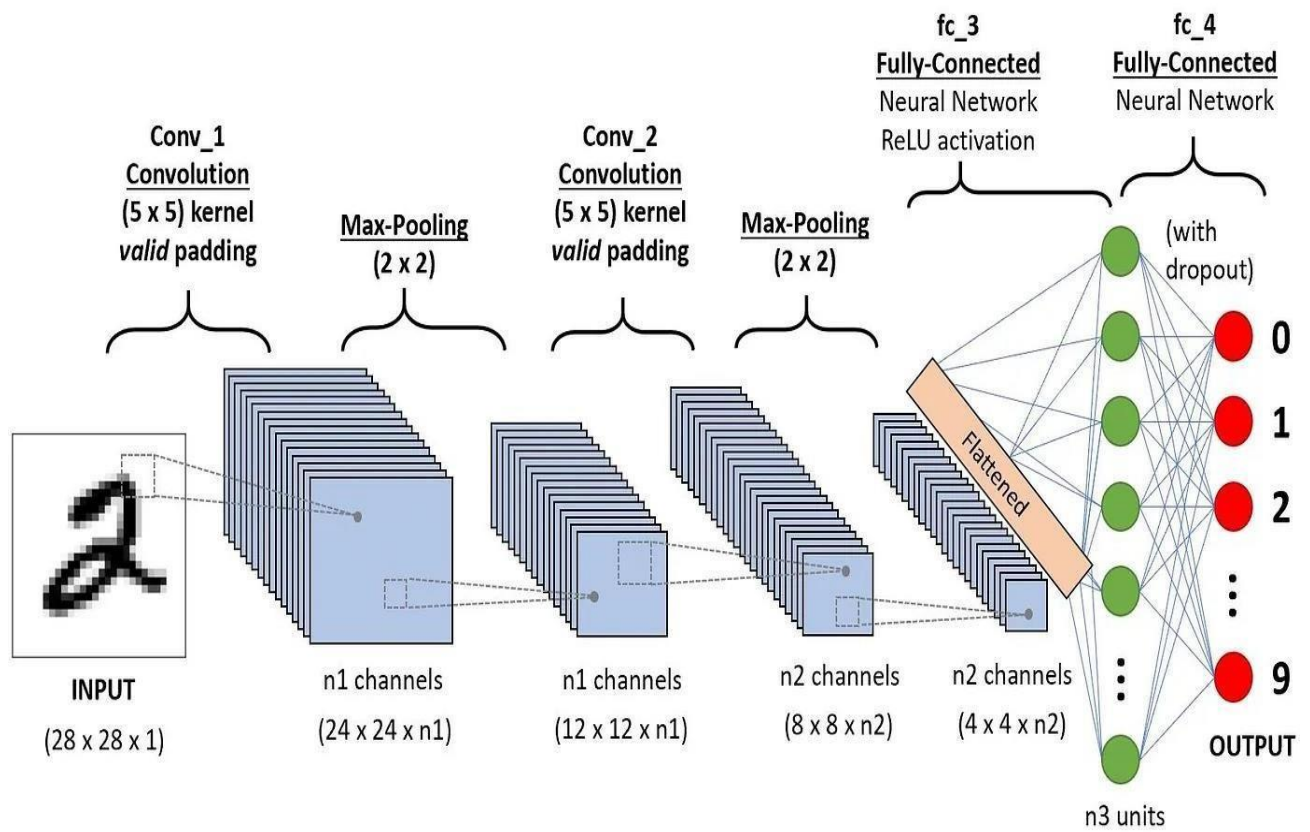
The most common aggregation functions that can be applied are:

- Max pooling, which is the maximum value of the feature map
- Sum pooling corresponds to the sum of all the values of the feature map
- Average pooling is the average of all the values.

**4. Fully connected layers:**

These layers are in the last layer of the convolutional neural network, and their inputs correspond to the flattened one-dimensional matrix generated by the last pooling layer. ReLU activations functions are applied to them for non-linearity. Finally, a softmax prediction layer is used to generate probability values for each of the possible output labels, and the final label predicted is the one with the highest probability score.





## CHAPTER - 4

### PROJECT - 1

## NUMBER RECOGNITION

### Objective :

The primary objective of the number recognition project is to develop a machine learning-based system capable of accurately identifying and classifying handwritten or printed numeric characters from images.

### Technologies Used:

#### 1. Python Libraries:

1. NumPy & Pandas: For data handling and preprocessing.
2. Matplotlib & Seaborn: For visualizing dataset and results.
3. Scikit-learn: For training and evaluating machine learning models.

#### 2. Deep Learning Frameworks:

1. TensorFlow/Keras: To create and train neural networks for image classification.

#### 3. Computer Vision Libraries:

1. OpenCV: For preprocessing images (e.g., resizing, grayscale conversion, and noise reduction).

#### 1. Datasets:

1. MNIST Dataset: A well-known dataset of handwritten digits used for training and testing.

#### 2. Google Colab:

1. Cloud-based platform to train and evaluate models with GPU acceleration.

### Workflow Explanation:

1. Data Collection : Use datasets like MNIST, containing images of handwritten digits (0–9).
2. Preprocessing: Normalize pixel values, resize images, and split into training, validation, and test sets.
3. Model Design: Build a Convolutional Neural Network (CNN) for feature extraction and classification.
4. Training: Train the model using the training set with a suitable optimizer (e.g., Adam) and loss function.
5. Evaluation: Validate the model using metrics like accuracy and a confusion matrix.
6. Testing: Test the model on unseen data to ensure robustness.
7. Deployment: Save the model and integrate it into applications for real-world usage.

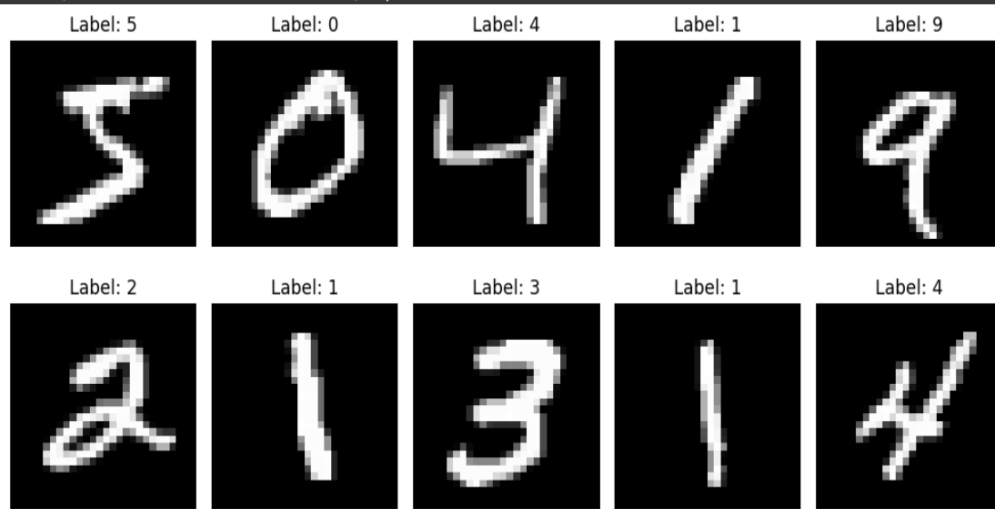
This concise workflow ensures efficient development and deployment of the number recognition system.

```
# Importing necessary libraries for data handling, visualization, and building the model
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist # Built-in dataset for handwritten digits
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical
```

```
[ ] # Loading the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Visualizing some samples from the dataset
plt.figure(figsize=(10, 5))
for i in range(10):
    plt.subplot(2, 5, i + 1)
    plt.imshow(x_train[i], cmap='gray')
    plt.title(f"Label: {y_train[i]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>  
11490434/11490434 0s 0us/step



```
# Normalizing the images (scaling pixel values to the range 0-1)
x_train = x_train / 255.0
x_test = x_test / 255.0

# Reshaping the data to add a channel dimension (for grayscale images)
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

# One-hot encoding the labels (e.g., 2 -> [0, 0, 1, 0, 0, 0, 0, 0, 0, 0])
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

```
[ ] # Creating a sequential model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)), # Convolution layer
    MaxPooling2D((2, 2)), # Max pooling layer
    Conv2D(64, (3, 3), activation='relu'), # Additional convolution layer
    MaxPooling2D((2, 2)), # Additional max pooling layer
    Flatten(), # Flattening layer to convert 2D data into 1D
    Dense(128, activation='relu'), # Fully connected dense layer
    Dense(10, activation='softmax') # Output layer with 10 neurons (for digits 0-9)
])

# Compiling the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base\_conv.py:107: UserWarning: Do not pass an `input\_shape`/`input\_dim` arg  
super().\_\_init\_\_(activity\_regularizer=activity\_regularizer, \*\*kwargs)

```
[ ] # Training the model on the training data
history = model.fit(x_train, y_train, epochs=5, batch_size=32, validation_split=0.1)
```

```
Epoch 1/5
1688/1688 ————— 53s 30ms/step - accuracy: 0.9012 - loss: 0.3109 - val_accuracy: 0.9837 - val_loss: 0.0585
Epoch 2/5
1688/1688 ————— 82s 31ms/step - accuracy: 0.9851 - loss: 0.0475 - val_accuracy: 0.9868 - val_loss: 0.0433
Epoch 3/5
1688/1688 ————— 47s 28ms/step - accuracy: 0.9902 - loss: 0.0304 - val_accuracy: 0.9900 - val_loss: 0.0368
Epoch 4/5
1688/1688 ————— 83s 29ms/step - accuracy: 0.9935 - loss: 0.0209 - val_accuracy: 0.9913 - val_loss: 0.0356
Epoch 5/5
1688/1688 ————— 81s 28ms/step - accuracy: 0.9948 - loss: 0.0159 - val_accuracy: 0.9898 - val_loss: 0.0389
```

```
[ ] # Evaluating the model on the test data
test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=0)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
```

```
Test Accuracy: 99.16%
```

```
from google.colab import files
from PIL import Image

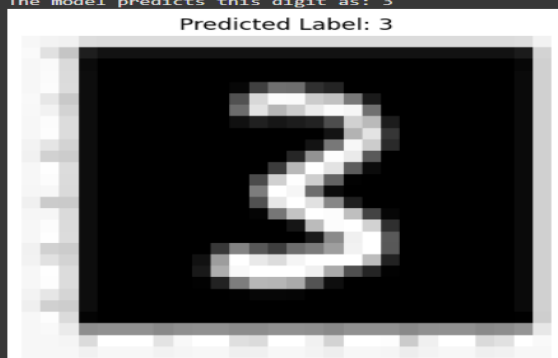
# Upload a handwritten digit image (28x28 grayscale recommended)
uploaded = files.upload() # Select an image to upload
img_path = list(uploaded.keys())[0]

# Preprocess the image for prediction
img = Image.open(img_path).convert('L') # Convert to grayscale
img = img.resize((28, 28)) # Resize to 28x28 pixels
img_array = np.array(img) / 255.0 # Normalize pixel values
img_array = img_array.reshape(1, 28, 28, 1) # Reshape for the model

# Predict the digit
prediction = model.predict(img_array)
predicted_label = np.argmax(prediction)
print(f"The model predicts this digit as: {predicted_label}")

# Show the uploaded image
plt.imshow(img, cmap='gray')
plt.title(f"Predicted Label: {predicted_label}")
plt.axis('off')
plt.show()
```

```
Choose Files 3.jpg
• 3.jpg(image/jpeg) - 17222 bytes, last modified: 16/11/2024 - 100% done
Saving 3.jpg to 3.jpg
1/1 ————— 0s 21ms/step
The model predicts this digit as: 3
```



```
[ ] # Save the trained model
model.save('digit_recognition_model.h5')
print("Model saved successfully!")
```

## **CONCLUSION :**

The number recognition project successfully demonstrated the application of machine learning techniques to accurately identify handwritten digits. By utilizing a dataset like MNIST and training a model, we were able to develop a system capable of recognizing digits with a high level of accuracy. This project highlighted the effectiveness of Convolutional Neural Networks (CNNs) in image processing tasks, especially for pattern recognition and classification. The model achieved promising results, which could be further improved with additional training, data augmentation, or the use of more advanced architectures. This project not only enhanced my understanding of machine learning algorithms but also provided hands-on experience with data preprocessing, model training, and evaluation techniques, laying a solid foundation for future AI and machine learning projects.

## CHAPTER - 5

### PROJECT - 2

### SPAM OR HAM DETECTION

#### Objective

The objective of this project is to build a machine learning model that can accurately classify emails as either spam or ham (not spam). This is achieved by training a Naive Bayes classifier on a dataset of labeled emails.

#### Technologies Used:

- **Python:** The core programming language used for the project.
- **Pandas:** Used for data manipulation and analysis, particularly for reading and processing the email dataset.
- **NumPy:** Provides support for numerical operations and array processing.
- **Scikit-learn:** A machine learning library used for model training, evaluation, and feature extraction.
- **CountVectorizer (from Scikit-learn):** Converts text data into numerical feature vectors for machine learning.
- **MultinomialNB (from Scikit-learn):** The Naive Bayes algorithm used for spam classification.
- **Google Colab:** A cloud-based platform providing a Jupyter Notebook environment for running the code.
- **Joblib:** Used for saving and loading the trained model and vectorizer.

#### Workflow Explanation:

- **Prepare Data:** Load and clean the labeled email dataset, converting labels to numerical form.
- **Extract Features:** Transform email text into numerical features using CountVectorizer.
- **Train Model:** Train a Naive Bayes model on the training data to learn spam/ham patterns.
- **Evaluate Model:** Assess model performance on the testing data using metrics like accuracy.
- **Deploy Model:** Save the trained model and vectorizer for later use.
- **Predict:** Classify new emails as spam or ham using the loaded model.

✓ 9s

```
# Importing libraries for data processing and machine learning
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

✓ 4m

```
from google.colab import files

# Upload the dataset
uploaded = files.upload() # Choose the SMS Spam Collection dataset
```

Choose Files mail\_data.csv

- **mail\_data.csv**(text/csv) - 485702 bytes, last modified: 11/16/2024 - 100% done

Saving mail\_data.csv to mail\_data.csv

✓ 0s

```
# Replace 'spam.csv' with the name of your uploaded file
data = pd.read_csv('mail_data.csv', encoding='latin-1')

# Check the first few rows of the dataset
print(data.head())

# Extract relevant columns (assumes first column is labels and second is messages)
data = data[['Category', 'Message']] # Use the correct column names from your dataset
data.columns = ['Label', 'Message'] # Rename columns to 'Label' and 'Message'

# Display dataset statistics
print("Dataset size:", data.shape)
print(data['Label'].value_counts())
```

Category Message

0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Dataset size: (5572, 2)

```
Label
ham      4825
spam      747
Name: count, dtype: int64
```

```
[ ] # Map 'ham' to 0 and 'spam' to 1
    data['Label'] = data['Label'].map({'ham': 0, 'spam': 1})

    # Check for missing data
    print(data.isnull().sum())
```

```
Label      0
Message     0
dtype: int64
```

```
0s # Splitting into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(data['Message'], data['Label'], test_size=0.2, random_state=42)

print(f"Training size: {len(X_train)}, Testing size: {len(X_test)}")
```

```
Training size: 4457, Testing size: 1115
```

```
0s # Convert text to numerical feature vectors
vectorizer = CountVectorizer(stop_words='english', max_features=5000)
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)

print("Feature vector shape (training):", X_train_vectorized.shape)
```

```
Feature vector shape (training): (4457, 5000)
```

```
0s # Initialize and train the Naive Bayes model
model = MultinomialNB()
model.fit(X_train_vectorized, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test_vectorized)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")
```

```
Model Accuracy: 99.01%
```



```

# Classification report
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=['Ham', 'Spam']))

# Confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

```

Classification Report:

	precision	recall	f1-score	support
Ham	0.99	1.00	0.99	966
Spam	0.97	0.95	0.96	149
accuracy			0.99	1115
macro avg	0.98	0.97	0.98	1115
weighted avg	0.99	0.99	0.99	1115

Confusion Matrix:

```

[[962  4]
 [ 7 142]]

```

✓  
0s

```

# Test the model with your own input
test_message = ["hi "]
test_vectorized = vectorizer.transform(test_message)

# Predict spam or ham
prediction = model.predict(test_vectorized)
print("Prediction:", "Spam" if prediction[0] == 1 else "Ham")

```

Prediction: Ham

OR

✓  
0s

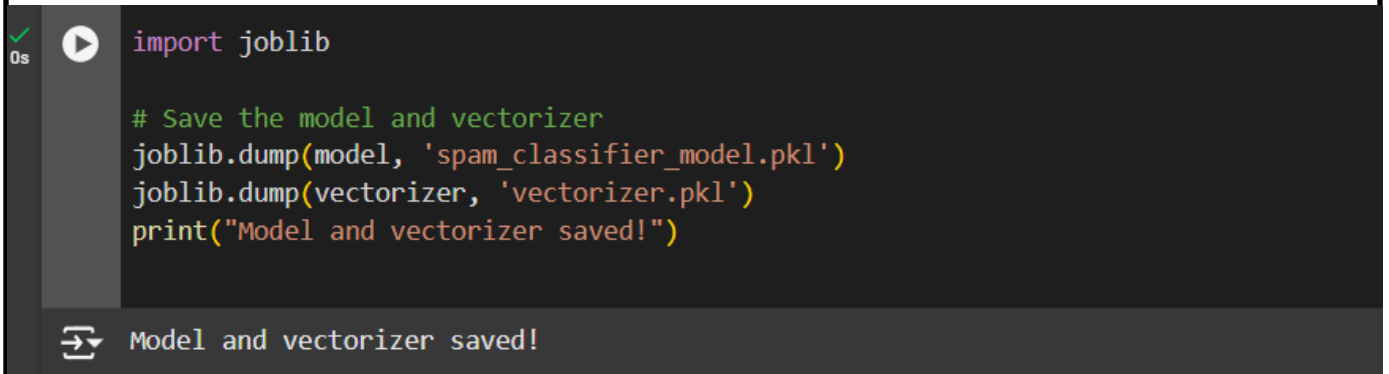
```

# Test the model with your own input
test_message = ["congrats you have won the lottery"]
test_vectorized = vectorizer.transform(test_message)

# Predict spam or ham
prediction = model.predict(test_vectorized)
print("Prediction:", "Spam" if prediction[0] == 1 else "Ham")

```

Prediction: Spam



A terminal window with a dark background. On the left, there is a vertical sidebar with a green checkmark and the text '0s'. The main area contains Python code for saving a model and vectorizer using joblib. Below the code, a grey bar shows the output of the print statement.

```
import joblib

# Save the model and vectorizer
joblib.dump(model, 'spam_classifier_model.pkl')
joblib.dump(vectorizer, 'vectorizer.pkl')
print("Model and vectorizer saved!")
```

Model and vectorizer saved!

**Conclusion:**

This project successfully built a spam detection system using a Naive Bayes classifier. The model achieved high accuracy in classifying emails as spam or ham. This demonstrates the effectiveness of machine learning for this task.

**Key Takeaways:**

- High accuracy:** The model effectively identifies spam emails.
- Practical application:** The trained model and vectorizer can be integrated into real-world systems.
- Potential improvements:** Exploring other algorithms and feature engineering techniques could further enhance performance

## **CHAPTER - 6**

### **CONCLUSION**

In summary, our Data Science (DS) and Artificial Intelligence (AI) internship has been transformative, providing hands-on experience with cutting-edge technologies and real-world projects. Collaborating with a dynamic team, I honed my problem-solving skills, applied theoretical knowledge to practical situations, and deepened my understanding of DS and AI concepts. The challenges encountered fostered creativity and critical thinking, contributing to a well-rounded skill set that encompasses various ML algorithms, data preprocessing techniques, and model evaluation strategies.

Additionally, the internship emphasized the importance of ethical considerations in AI development. It highlighted the need for responsible and transparent practices, broadening my awareness of the societal impact of AI technologies. Looking back, this experience has not only shaped my career path in the DS and AI field but has also provided a solid foundation for future endeavors. I am grateful for the opportunity and eager to contribute to the ongoing advancement of these technologies.