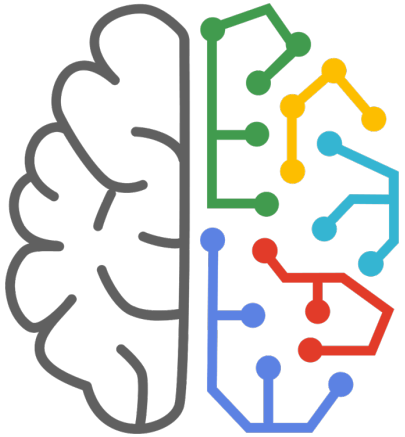


CODEBUGGED

Catalyze AI, Amplify Life

We Help Enterprises in Enabling AI at Scale

Discover the Power of Predictive Analytics and Transform Your Manufacturing Operations

Let’s Welcome the **WITH** World

HMC
Human Machine Collaboration

Enterprise Productivity ∝ Machine Health

Is Your Machine feeling FEVERISH* or Tachycardia# ?

* Increased Temperature
\# Increased Vibration

Reactive

Preventive

Predictive

Start

Predictive Monitoring

before it breaks down

Introduction

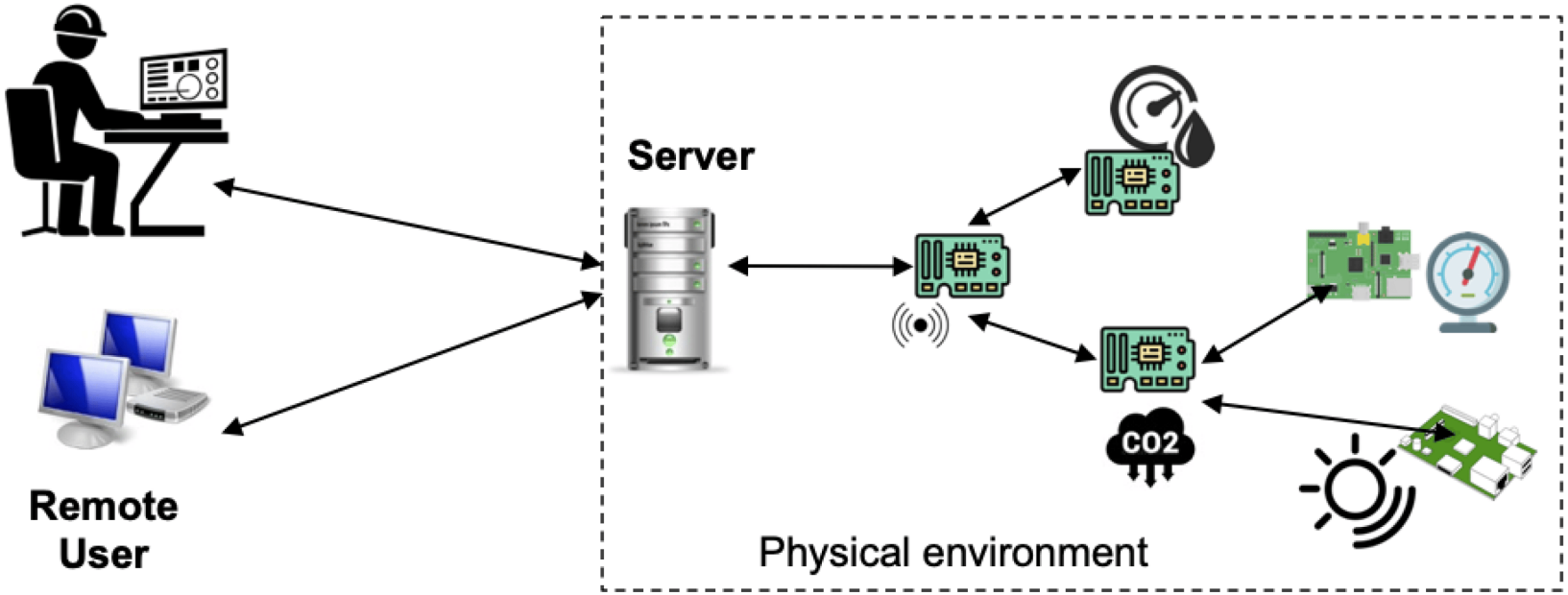
- Predictive analytics involves extracting information from existing data sets to determine patterns and predict future outcomes and trends.

- Importance: In manufacturing, it **reduces costs**, **improves quality**, and **shortens time-to-market**.

Basics of Predictive Analytics

- Data Collection: IoT, sensors, logs, feedback.
- Data Cleaning: Removing outliers, filling missing values.
- Data Analysis: Statistical methods, machine learning algorithms, neural networks.

Internet of Things



The Process

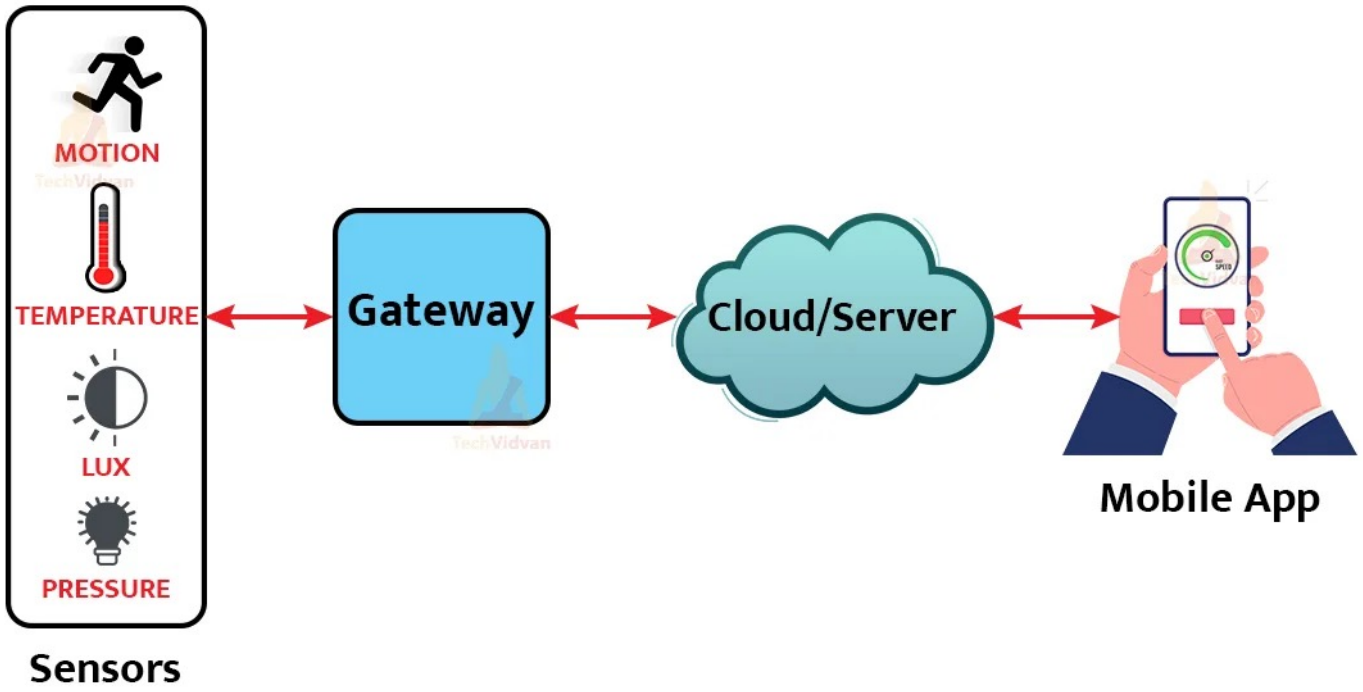
The Predictive Analytics Process

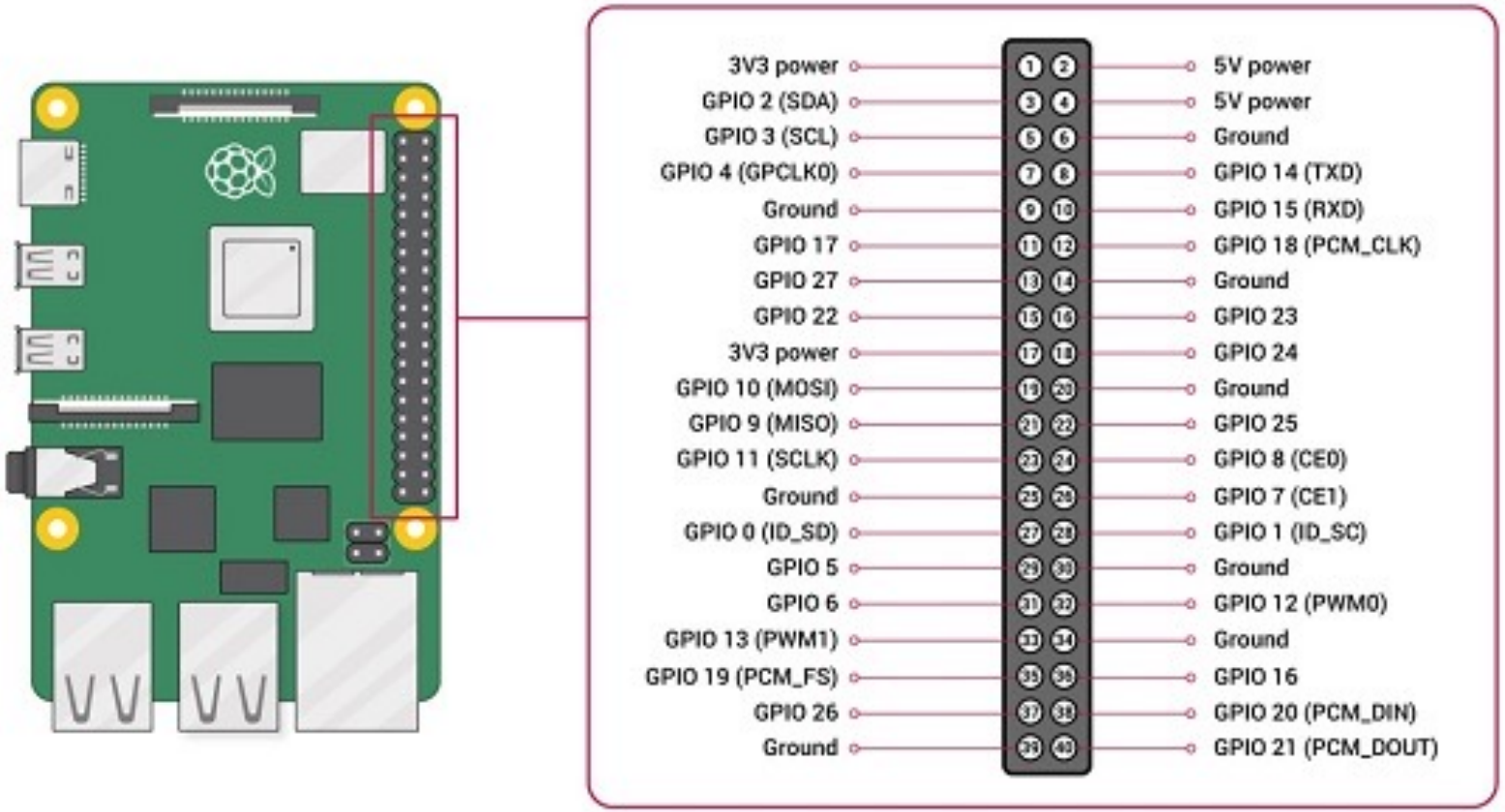
Pull	Prepare	Pick	Predict	Plan
Extract the data from where it lives.	Clean, refine, and prepare it.	Identify what to predict.	Create the prediction.	Develop a plan of action.

Introduction to Internet of Things

- **IoT (Internet of Things)**:
  - Definition: Connecting everyday objects to the internet for data exchange.
  - Examples: Smart homes, wearables, connected cars.
- **Industrial IoT (IIoT)**:
  - Definition: Application of IoT in industries for optimization & automation.
  - Examples: Predictive maintenance, smart factories, connected logistics.
- **Importance**:
  - **Efficiency**: Automates and optimizes processes.
  - **Data Utilization**: Gathers and analyzes data for insights.
  - **Cost Savings**: Reduces operational costs through predictive maintenance.
  - **Safety**: Monitors and ensures safety in industrial environments.
  - **Innovation**: Opens doors for new business models and services.

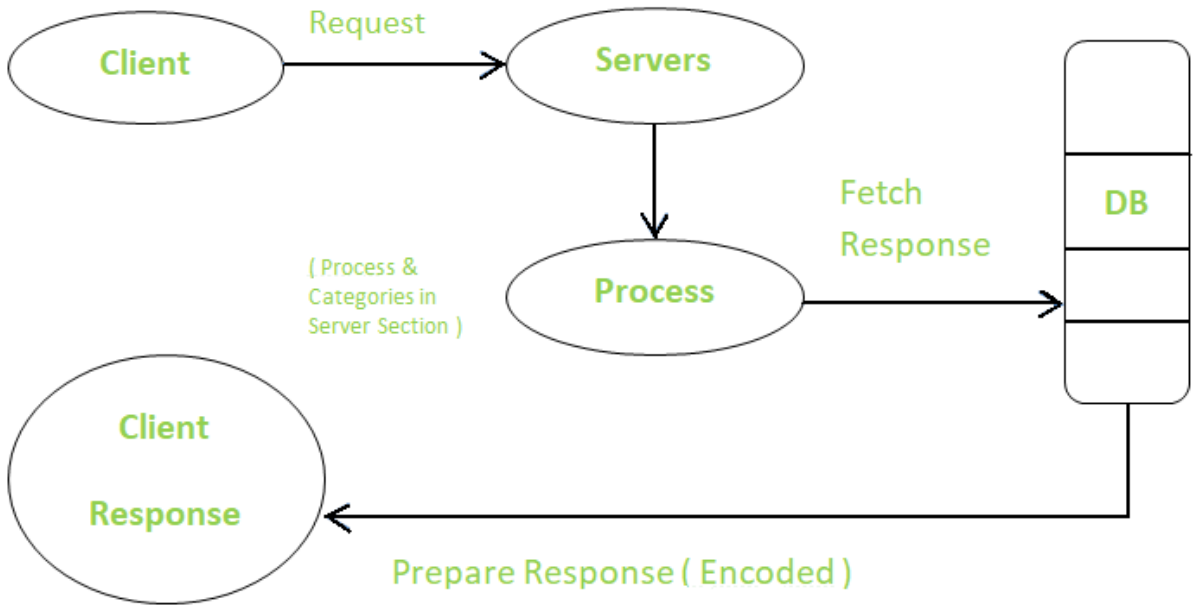
Working of IoT





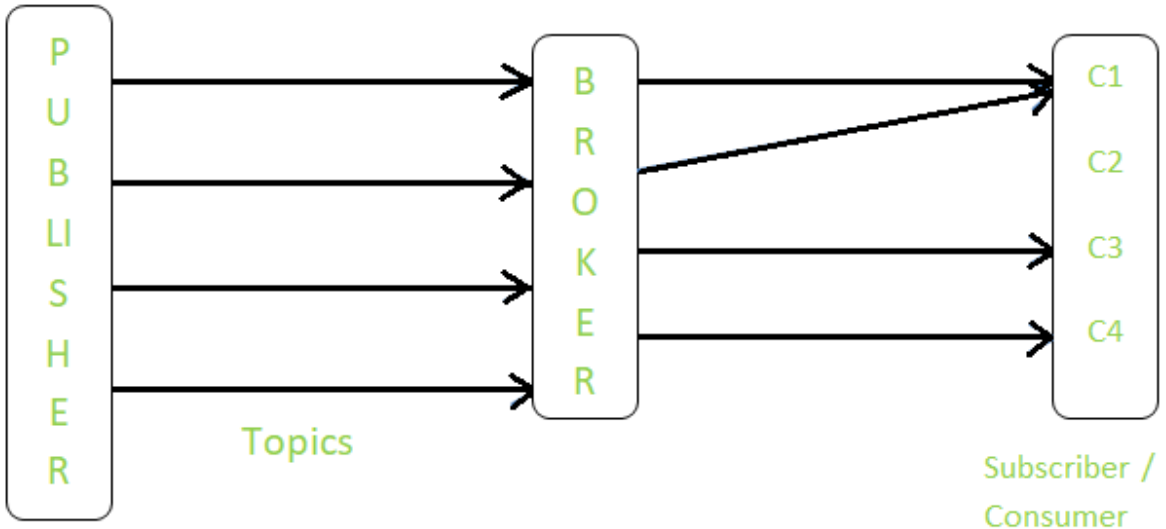
Request Response Model

1. Client-server architecture used.
2. Client sends encoded requests to the server.
3. Model is stateless; no data retention between requests.
4. Server:
   - Categorizes the request.
   - Retrieves data from the database.
   - Prepares and sends an encoded response.
5. Follows the Request-Response communication model.
   - Server determines response upon client request.



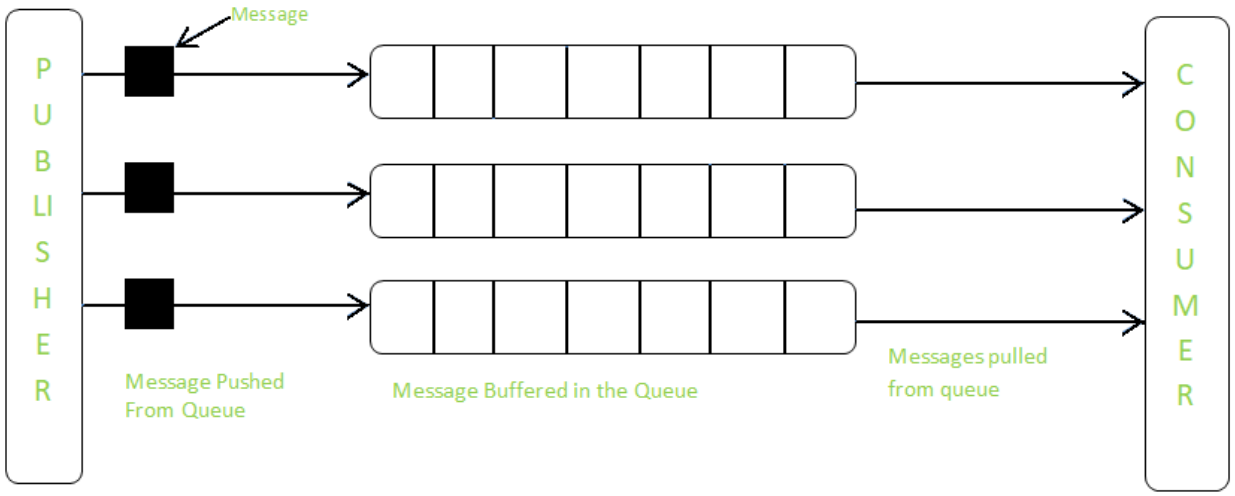
Publisher Subscriber Model

1. Three main entities:
   - Publishers
   - Brokers
   - Consumers
2. Publishers:
   - Source of data.
   - Send data to topics managed by brokers.
   - Unaware of consumers.
3. Consumers:
   - Subscribe to topics managed by brokers.
4. Brokers:
   - Accept data from publishers.
   - Distribute data to appropriate consumers.
   - Aware of consumer-topic relationships, unlike publishers.



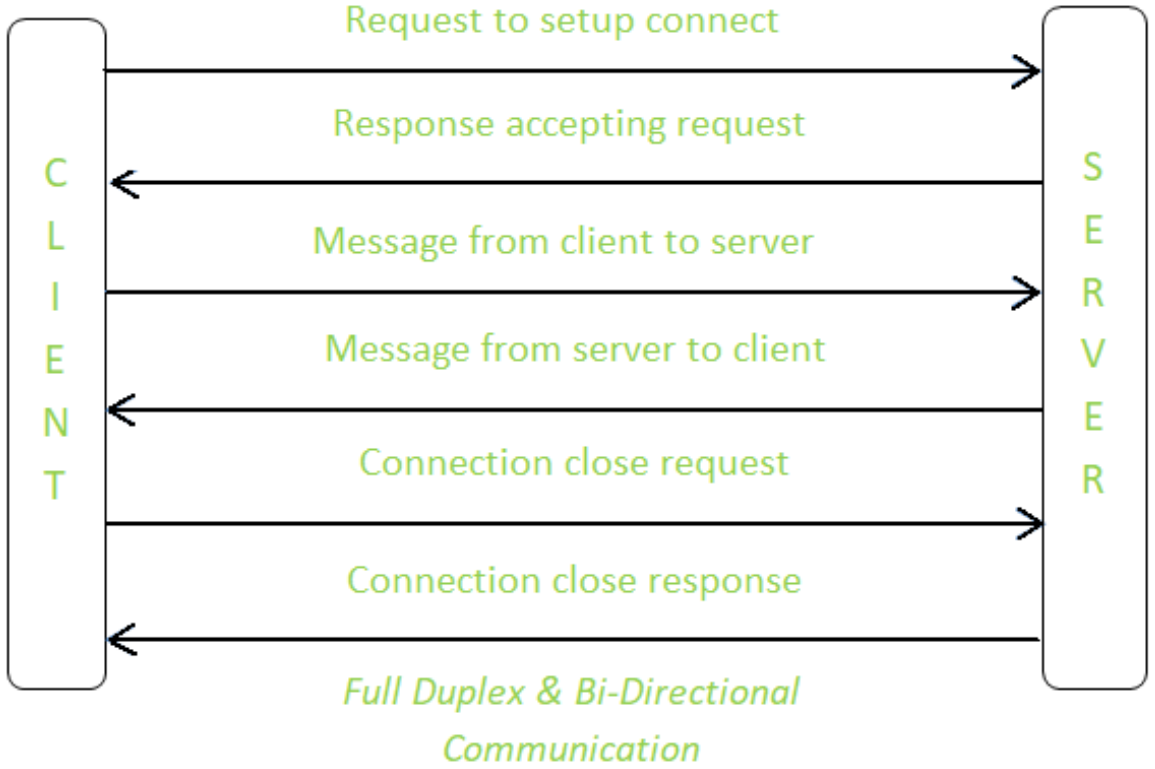
Push Pull Model

1. Push-pull model includes:
   - Data publishers
   - Data consumers
   - Data queues
2. Publishers and Consumers:
   - Unaware of each other.
3. Publishers:
   - Publish and push data into the queue.
4. Consumers:
   - Pull data out of the queue.
5. Queues:
   - Act as a buffer for messages.
   - Decouple messaging between producer and consumer.
   - Buffer helps with rate mismatches between data push (producers) and data pull (consumers).



Exclusive Pair Model

1. **Exclusive Pair Model**:
   - Bi-directional communication between client and server.
   - Full-duplex communication, allowing simultaneous sending and receiving of data.
   - Connection remains open until the client requests to close it.
2. **Server's Role**:
   - Keeps a record of all opened connections.
3. **Connection Characteristics**:
   - State-full connection model.
   - Server is aware of all open connections.
4. **Implementation**:
   - WebSocket-based communication API is built on this model.



Key Benefits of PA in Manufacturing

- **Predictive Maintenance:** Reduces downtime by predicting equipment failures before they happen.
- **Quality Assurance:** Identifies potential defects in real-time, thereby reducing waste.
- **Supply Chain & Inventory Management:** Forecasting demand to reduce inventory costs and improve service levels.
- **Energy Management:** Predicting energy consumption patterns to optimize energy usage and costs.

Condition Based Monitoring & Predictive Maintenance

- CBM is a maintenance strategy based on the **current health state** of the equipment using the sensor data collected from the equipment

- Models:-
  - Simple Thresholding
  - Machine Learning and Deep Learning Models

- Benefits:-
  - Reduce downtime by pinpointing the source of faults
  - Reduce costs due to unplanned failure and unnecessary maintenance

- Predictive Maintenance is a maintenance strategy based on the predicted future health state of the equipment

- Predictive maintenance can be formulated in one of the two ways:
- Classification approach - predicts whether there is a possibility of failure in next n-steps.
- Regression approach - predicts how much time is left before the next failure. We call this Remaining Useful Life (RUL).

The former approach only provides a boolean answer, but can provide greater accuracy with less data. The latter needs more data although it provides more information about when the failure will happen.

Machine Learning Algorithms:

1. **Regression Algorithms:**
    - Linear Regression: Predicts the Remaining Useful Life (RUL) as a continuous value.
    - Cox Proportional Hazards Model: A regression model used for survival analysis, which can predict the time to a specific event (like equipment failure).
2. **Classification Algorithms:**
    - Logistic Regression: Predicts the probability of failure within a specified time frame.
    - Decision Trees and Random Forests: Used to classify equipment as "likely to fail" or "not likely to fail" based on various features.
    - Support Vector Machines (SVM): Can be used for both classification and regression tasks in predictive maintenance.
    - Gradient Boosting Machines (GBM): An ensemble method that can predict equipment failures based on historical and operational data.
3. **Time Series Analysis:**
    - ARIMA (Autoregressive Integrated Moving Average): Predicts future values based on past observations.
    - Exponential Smoothing: Forecasts time series data considering trends and seasonality.
4. **Clustering Algorithms:**
    - K-means Clustering: Groups similar equipment or failure modes together, which can help in understanding common patterns or anomalies.
    - Hierarchical Clustering: Useful for understanding nested patterns in equipment behavior.
5. **Anomaly Detection:**
    - Isolation Forest: Detects anomalies in the data, which can indicate impending failures.
    - One-Class SVM: Used for novelty detection, identifying new or unknown failure patterns.

Deep Learning Algorithms:

1. **Recurrent Neural Networks (RNNs):**
   - Long Short-Term Memory (LSTM): Especially useful for time series data, as they can capture long-term dependencies in the data.
   - Gated Recurrent Units (GRU): A variation of LSTM with a simpler structure but similar capabilities.
2. **Convolutional Neural Networks (CNNs):**
   - Used for analysing sensor data, especially when the data can be structured in a grid-like fashion (e.g., vibration or sound waveforms).
3. **Autoencoders:**
   - A type of neural network used for anomaly detection. They learn to reconstruct input data, and deviations between the input and reconstruction can indicate anomalies.
4. **Sequence-to-Sequence Models:**
   - Useful for predicting sequences of events, such as a series of failures or maintenance activities.
5. **Transfer Learning:**
   - Leveraging pre-trained models on related tasks to improve predictive maintenance models, especially when labeled data is scarce.
6. **Attention Mechanisms and Transformers:**
   - Can capture important features in time series data and have been used in some advanced predictive maintenance applications.

Method 1: Vibration Analysis

1. **Data Collection**: The first step is to collect vibrational data from sensors attached to the machinery. These sensors continuously record vibration signals.
2. **Data Transmission**: Once collected, this data is transmitted to a data center or a cloud platform using a network of IoT devices.
3. **Data Storage**: The data is stored in a database which could be a part of a larger Industrial IoT platform.
4. **Data Processing & Analysis**: Analysts or automated software analyze the data using techniques like Fast Fourier Transform (FFT) to transform the time-domain signals to frequency-domain signals, facilitating the identification of abnormal vibration patterns.
5. **Fault Detection & Diagnosis**: Analysts identify potential issues by recognizing patterns and anomalies in the vibration data. Machine learning algorithms can also be used to automate this process, learning to recognize the signatures of various faults.
6. **Predictive Maintenance**: Insights derived from the analysis can be used to predict when maintenance will be needed, helping to avoid unscheduled downtime and extend the lifespan of the machinery.
7. **Action & Feedback**: Finally, actionable insights are used to schedule maintenance, make adjustments, or carry out other necessary actions. Feedback from these actions can be used to further refine the vibrational analysis system.

Method 2: Acoustic Analysis

1. **Data Collection:** Acoustic sensors placed near or on the machinery pick up sound waves produced during operation. These sensors can be sensitive to a wide range of frequencies to detect abnormalities in the acoustic patterns.
2. **Data Transmission:** The sensors transmit the recorded data to a centralized system for further analysis, typically via a secure IoT network.
3. **Data Storage:** Acquired data is stored in a centralized database, which can be cloud-based or on-premises, depending on the specific setup of the Industrial IoT system.
4. **Data Analysis:** Software tools analyze the acoustic data, utilizing algorithms to identify patterns and potential issues. Techniques such as Fourier Transform can be employed to break down complex sound signals into simpler components, making it easier to identify specific sound signatures associated with known issues.
5. **Fault Detection and Diagnosis:** By examining the frequency, amplitude, and other characteristics of the sound signals, analysts can identify issues such as misalignments, imbalances, or other mechanical faults.
6. **Predictive Maintenance:** Based on the analysis, a predictive maintenance strategy can be developed to address potential issues before they become serious problems, effectively preventing unplanned downtime.
7. **Action and Feedback Loop:** Actionable insights derived from the acoustic analysis guide the maintenance team in performing necessary repairs or adjustments. The feedback loop helps in refining the analysis strategy over time.

Method 3: Thermography

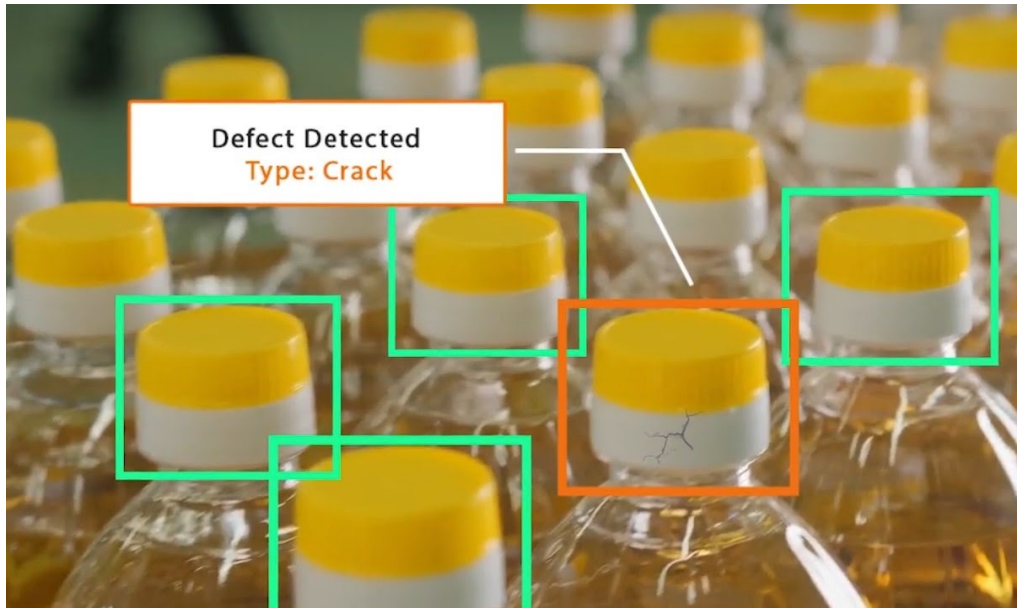
1. **Data Collection**: Thermal sensors or infrared cameras are used to gather thermal data of the equipment while it is operational. These devices capture the heat emitted from various parts of the machinery.
2. **Data Transmission**: The thermal data collected is transmitted in real time to a centralized system through a secure IoT network.
3. **Data Storage**: Once transmitted, this data is stored in a centralized database, which can be on-premises or cloud-based, facilitating easy access for analysis.
4. **Data Analysis**: Specialized software tools are utilized to analyze the thermal data, identifying patterns and anomalies which might indicate issues such as overheating or energy waste.
5. **Fault Detection and Diagnosis**: The analysis can pinpoint areas of concern based on temperature gradients and other thermal characteristics. This can be used to detect mechanical faults, electrical issues, or inefficient processes.
6. **Predictive Maintenance**: The insights derived from the thermal analysis help to develop a predictive maintenance strategy, scheduling interventions before major faults occur.
7. **Action and Feedback Loop**: Maintenance teams use the insights gained from the thermographic analysis to undertake necessary actions, including adjustments, repairs, or replacements. Feedback from these actions helps to refine the thermographic analysis over time.

Predictive Maintenance for Production Lines:

- **Challenge:** Unexpected machinery breakdowns leading to halts in the production line.
- **Solution:** Use predictive models to identify signs of wear and tear in machinery, allowing for timely maintenance.
- **Impact:** Decrease in unplanned downtimes and cost savings.

Quality Control and Defect Detection:

- **Challenge:** Detecting defects in parts can be time-consuming and costly.
- **Solution:** AI Algorithms can be trained to recognize defects from images or sensor readings.
- **Impact:** Improved product quality, reduced wastage and recalls.

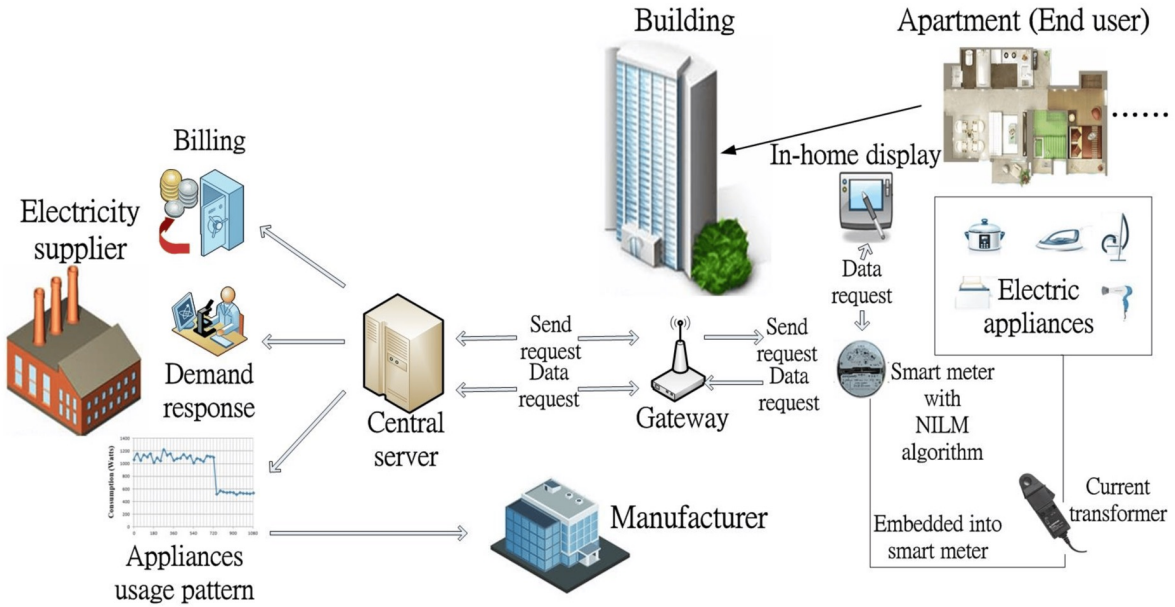


Demand Forecasting for Inventory Management:

- **Challenge:** Overproduction or underproduction due to poor demand forecasts.
- **Solution:** Analyzing historical sales data, market trends, and external factors (like economic indicators) to predict demand.
- **Impact:** Optimized inventory levels, reduced storage costs, and improved customer satisfaction.

Energy Consumption Prediction in Manufacturing Plants:

- **Challenge:** Fluctuating energy bills and inefficiencies in consumption.
- **Solution:** Use predictive analytics to forecast energy needs and schedule operations accordingly.
- **Impact:** Reduction in energy costs and improved sustainability.



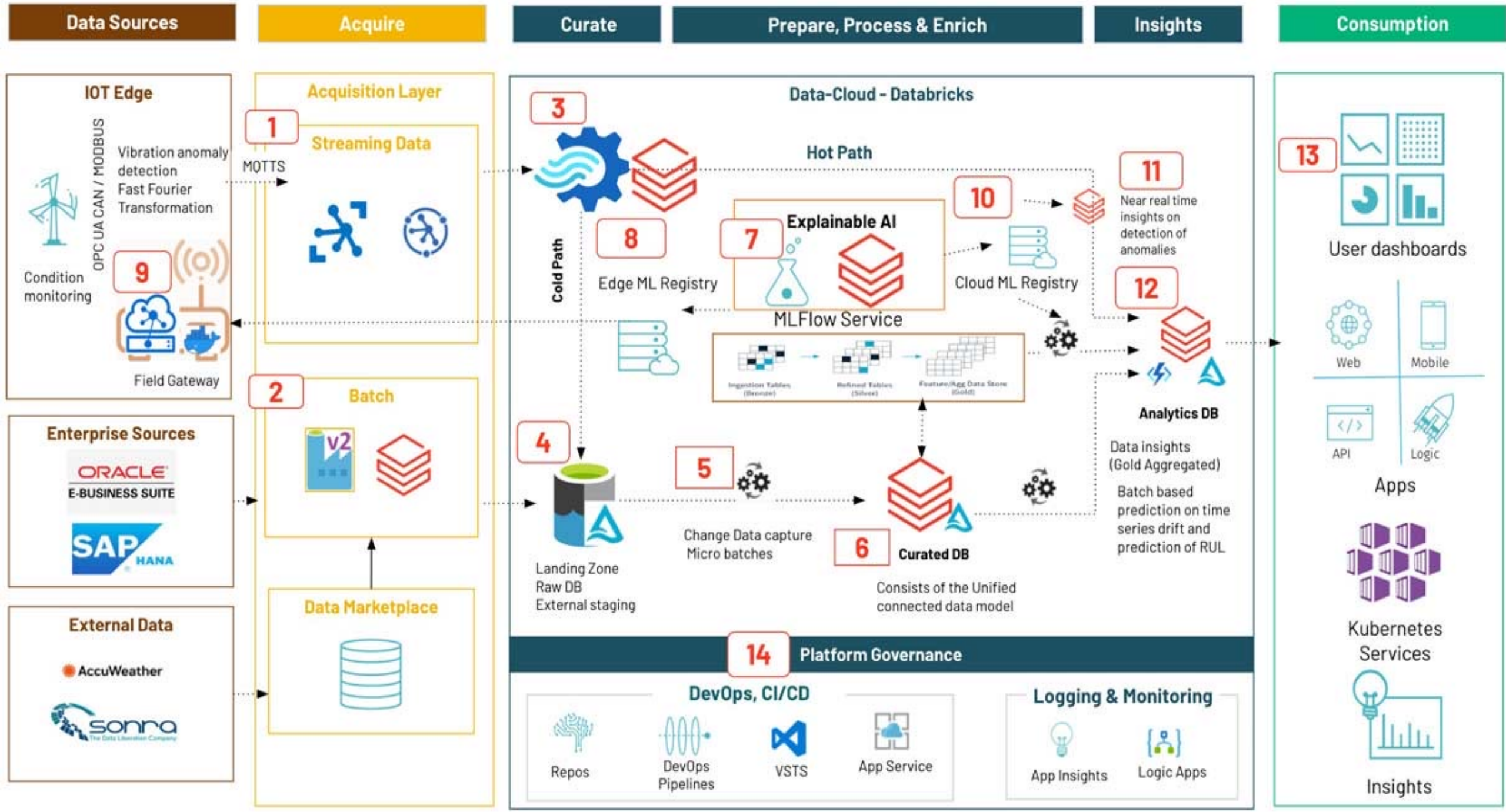
~~Challenges~~ I Pre-requisites

- **Data Quality:** Reliable predictions require quality data.
- **Data Volume:** Managing vast amounts of data can be challenging.
- **Skill Gap:** A need for skilled professionals to analyze the data.
- **Change Management:** Employees need to trust and act on predictive insights.

Conclusion:

- The integration of predictive analytics into manufacturing operations, especially in the automobile industry, can be a game-changer.

- With proper implementation and overcoming challenges, companies can realize significant cost savings, improve operational efficiency, and ensure product quality.

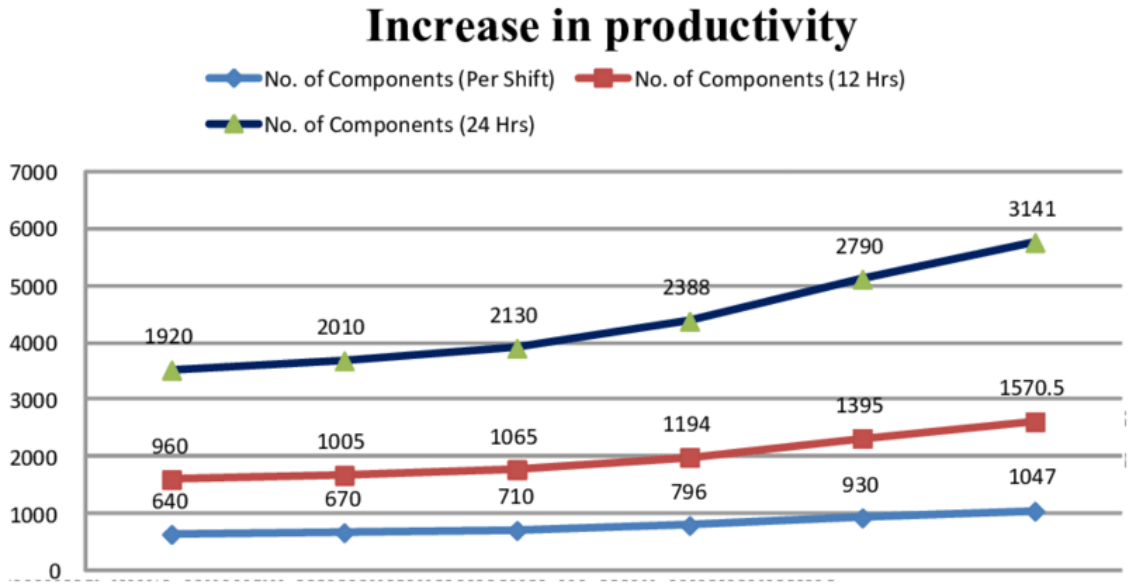

Data Sources
Acquire
Curate
Prepare, Process & Enrich
Insights
Consumption
IOT Edge
OPC UA CAN / MODBUS
Vibration anomaly detection Fast Fourier Transformation
Condition monitoring
9
Field Gateway
Enterprise Sources
ORACLE E-BUSINESS SUITE
SAP HANA
External Data
AccuWeather
sonra
Acquisition Layer
1
Streaming Data
MQTTS
2
Batch
Data Marketplace
Data-Cloud - Databricks
3
Hot Path
Cold Path
8
Edge ML Registry
7
Explainable AI
MLFlow Service
10
Cloud ML Registry
11
Near real time insights on detection of anomalies
12
Analytics DB
Data insights (Gold Aggregated)
Batch based prediction on time series drift and prediction of RUL
4
Landing Zone
Raw DB
External staging
5
Change Data capture
Micro batches
6
Curated DB
Consists of the Unified connected data model
14
Platform Governance
DevOps, CI/CD
Repos
DevOps Pipelines
VSTS
App Service
Logging & Monitoring
App Insights
Logic Apps
13
User dashboards
Web
Mobile
API
Logic
Apps
Kubernetes Services
Insights

Productivity and Quality Enhancement

What is Productivity

- Definition: The efficiency in which input (resources) is converted to output (goods/services).
- Measured as a ratio of output to input.
- Directly linked to profitability and growth.
- A key performance indicator in industries.

- For e.g., Productivity = Revenue / Burn-Rate * 100



What is Quality Enhancement?

- Definition: Systematic efforts to improve the standard or quality of products, services, or processes.

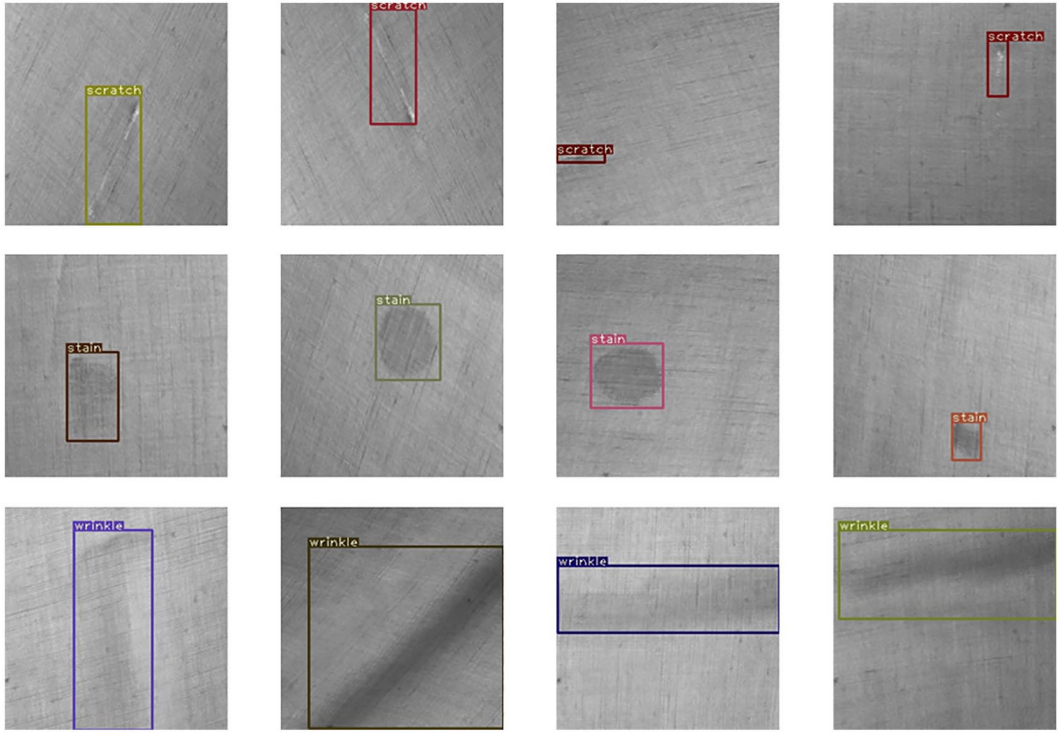
- Ensures customer satisfaction.
- Reduces wastage and rework.
- Enhances brand reputation.

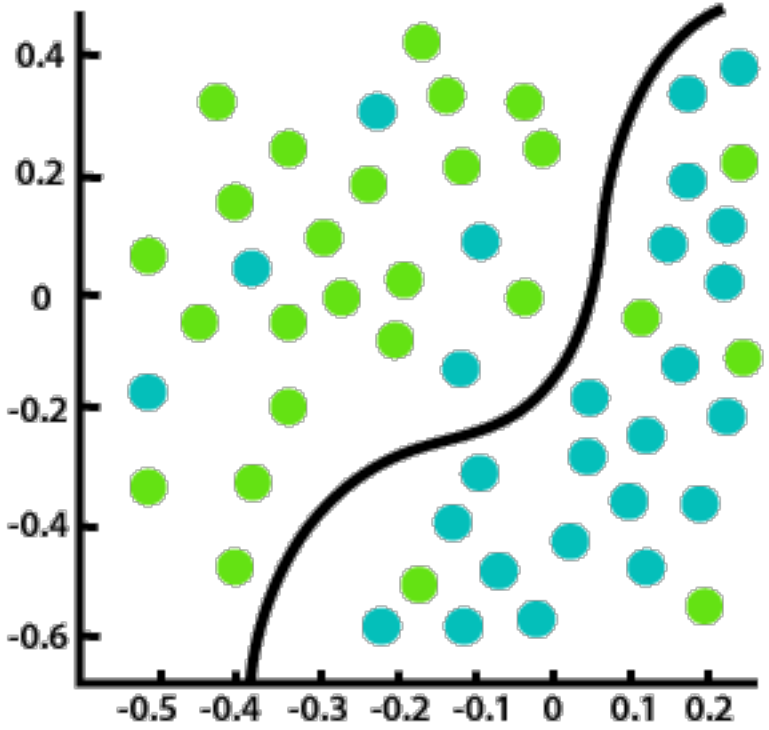
The Symbiotic Relationship

- Productivity & Quality: Hand in Hand
- Quality enhancement leads to reduced defects, increasing productivity.
- Higher productivity can provide more resources for quality checks.
- Balancing both is key for industrial success.

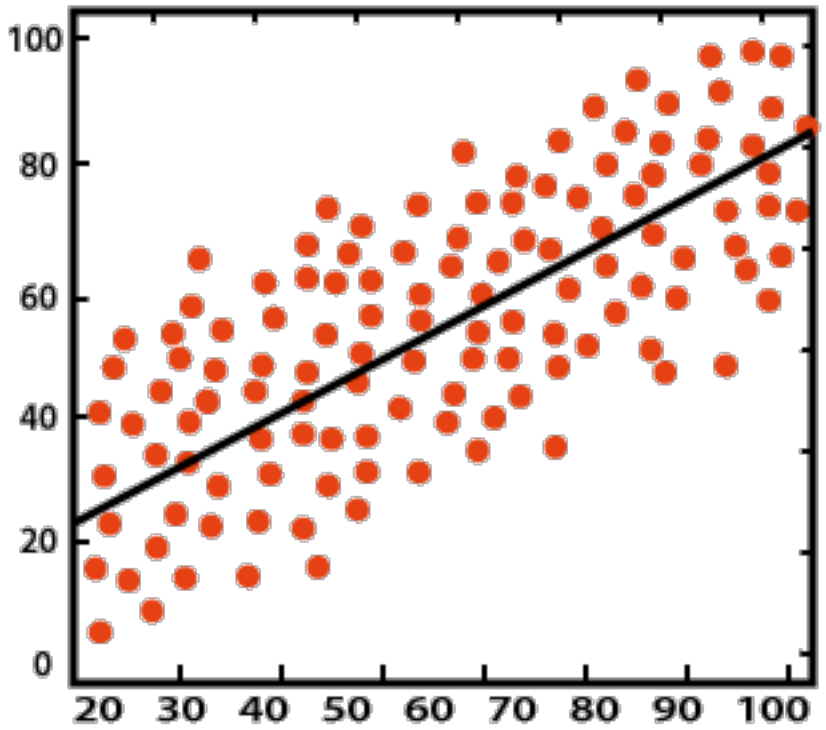
Case Study: Image Analytics in Textile Industry

- Cameras used to detect fabric defects
- Enhanced quality checks leading to higher customer satisfaction
- https://www.kaggle.com/datasets/priemshpathirana/fabric-stain-dataset





Classification



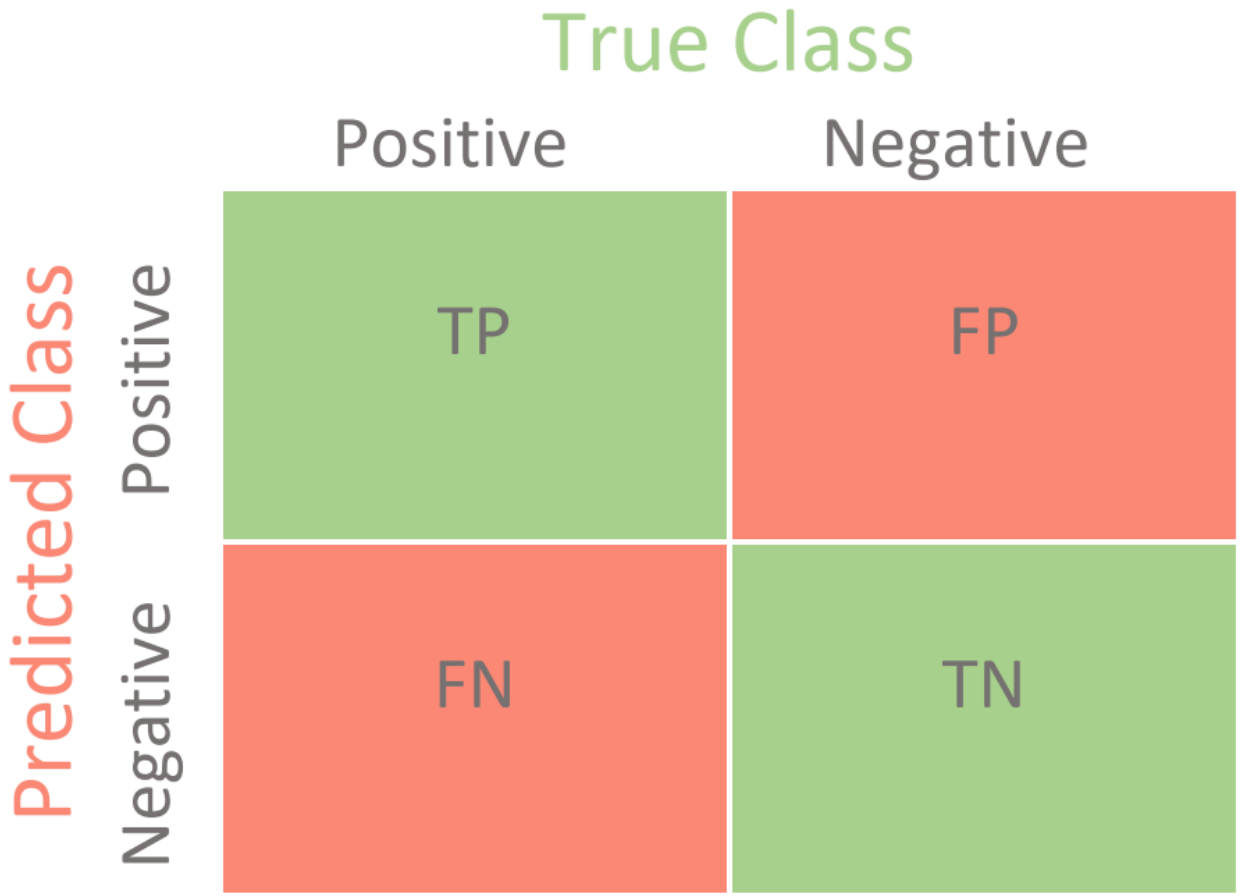
Regression

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

One Hot Encoding

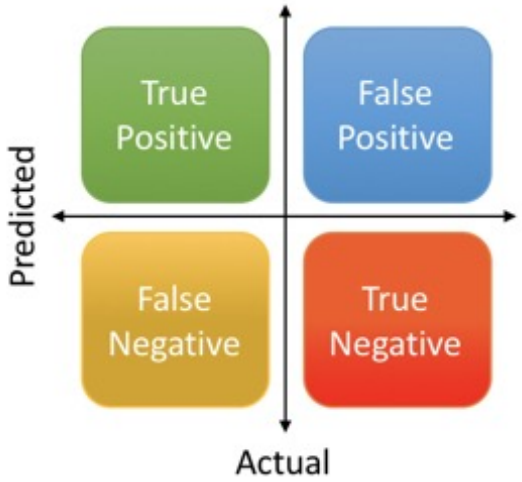
Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

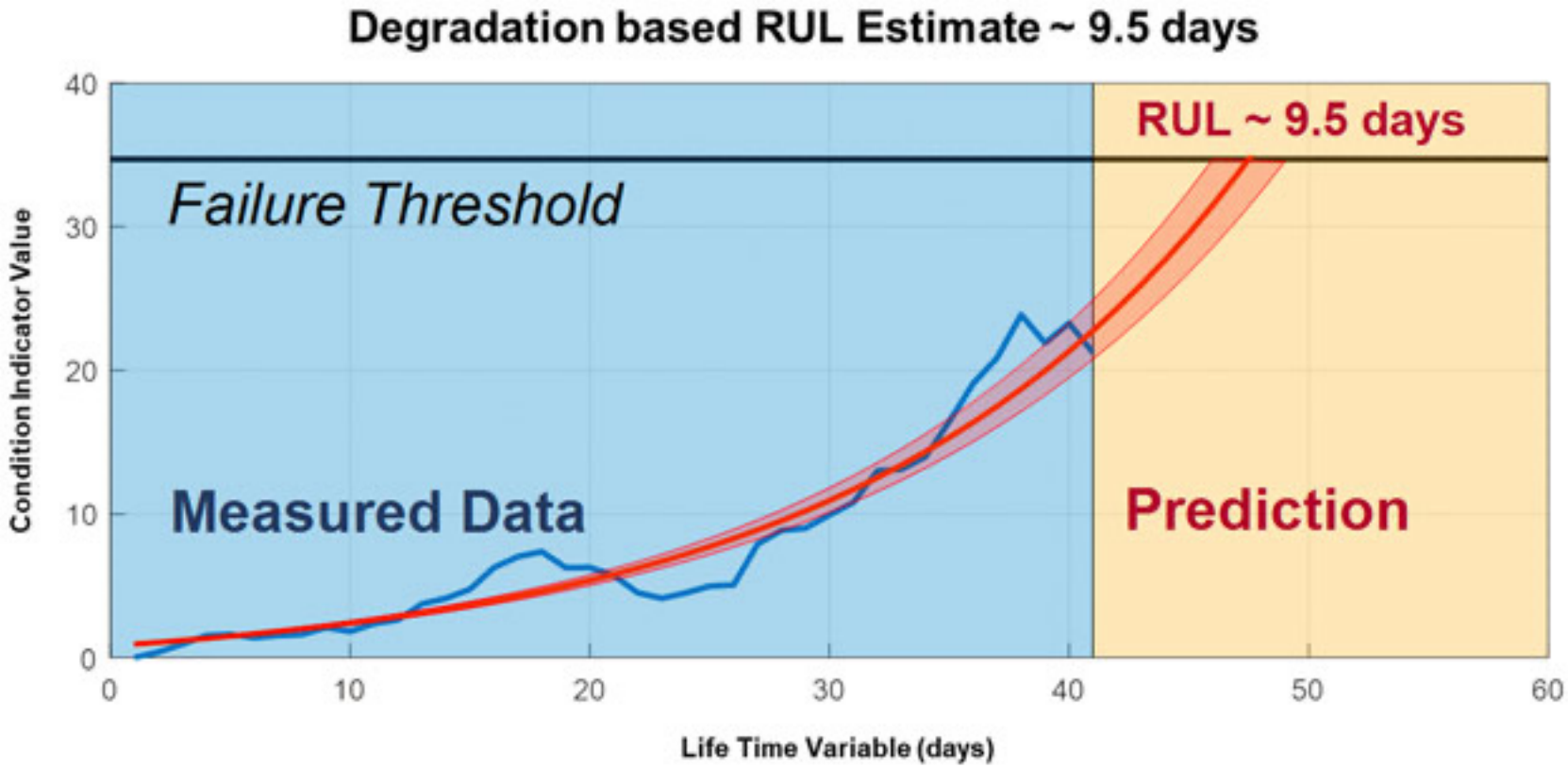


$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

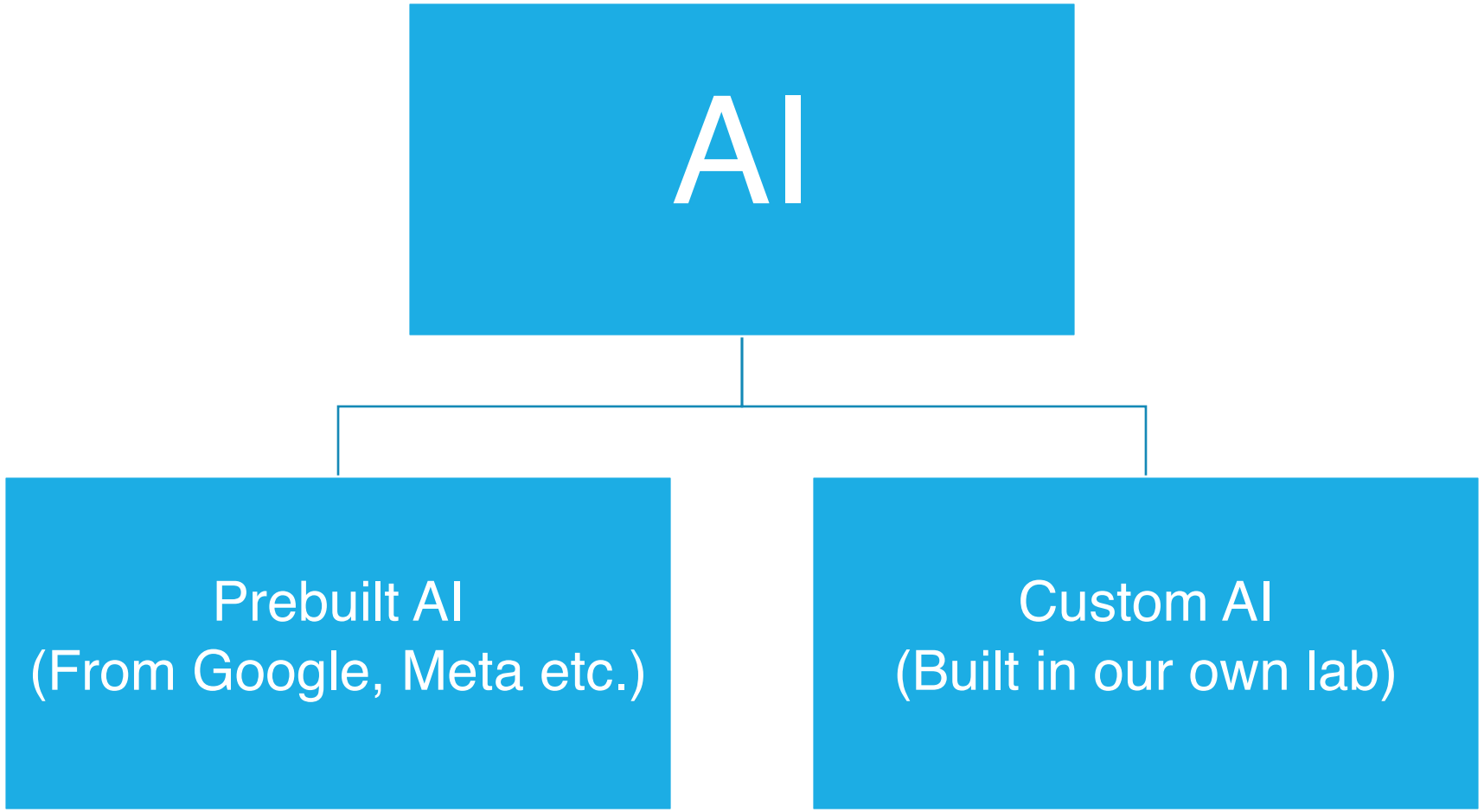
$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$





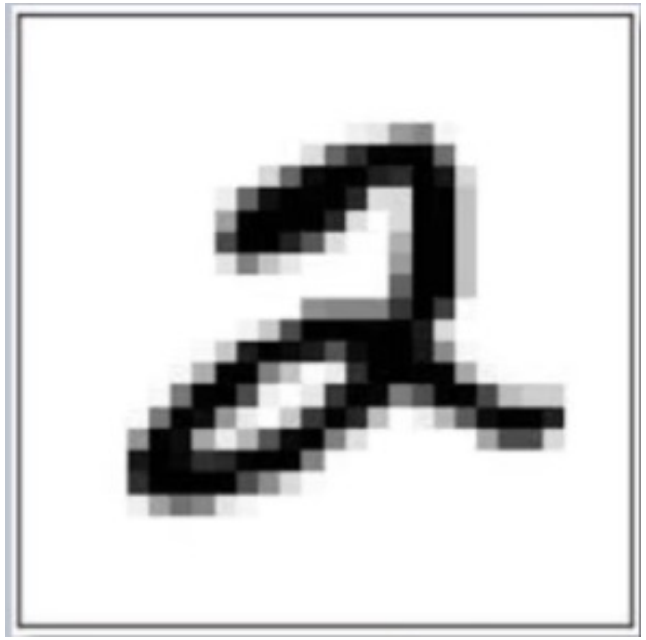
Computer Vision for Quality Enhancement



What is Computer Vision ?

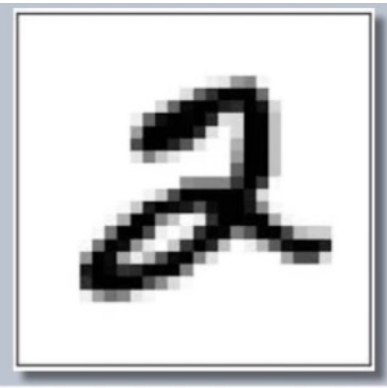
- Computer vision is a field of AI that trains computers to capture and interpret information from image and video data.
- By applying machine learning (ML) models to images, computers can classify objects and respond—like unlocking your smartphone when it recognizes your face.

What do you see ?



CODEBUGGED

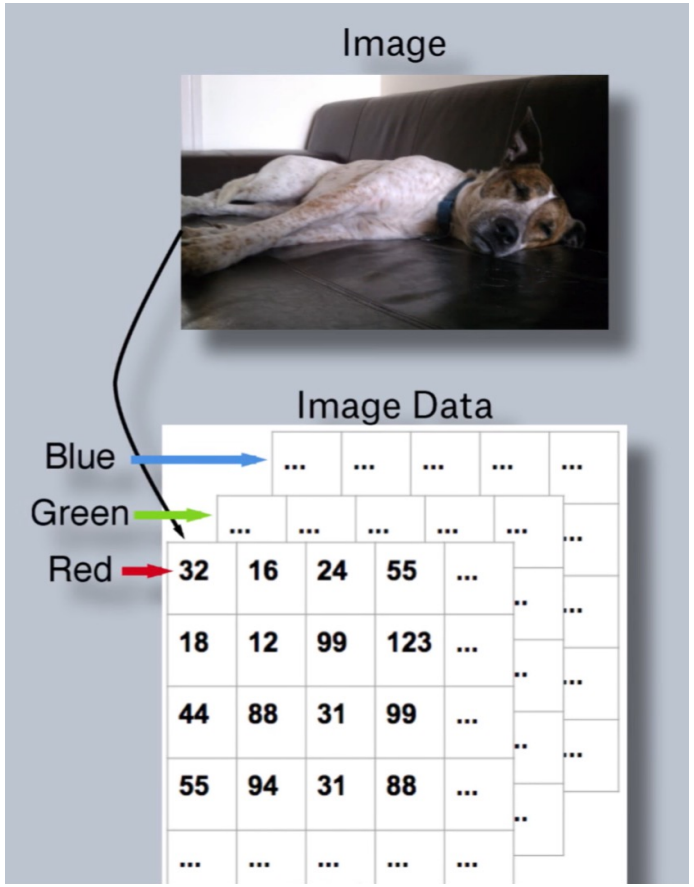
How machine stores an image?



0	0	200	150	0	0
0	143	55	99	222	0
0	188	0	0	181	0
0	0	0	200	0	0
0	0	149	0	0	0
0	245	202	140	225	0
0	0	0	0	0	0

CODEBUGGED

Coloured Images



CODEBUGGED

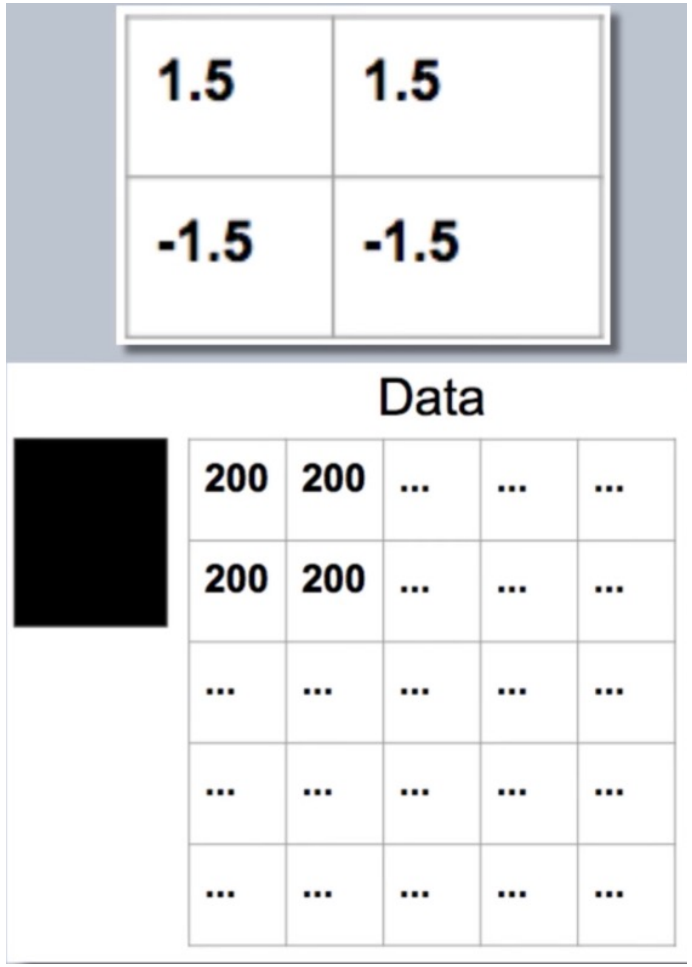
Tensors and Convolutional Operations

Tensors is like a matrix but it can have any number of dimension. The images stored in 2D matrix are tensors. We apply convolutions on tensors.

This is an example of convolution – A small tensor that can be multiplied to tensors.
The one you see below is a horizontal line detector. AMAZING ?

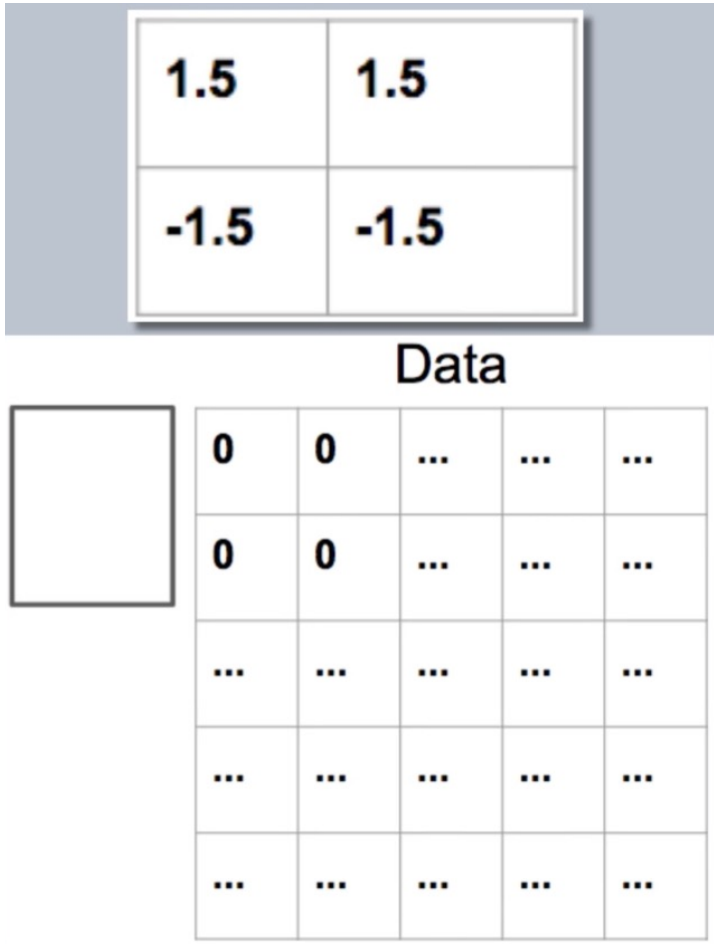
CODEBUGGED

1.5*200 + 1.5*200 – 1.5*200 – 1.5*200



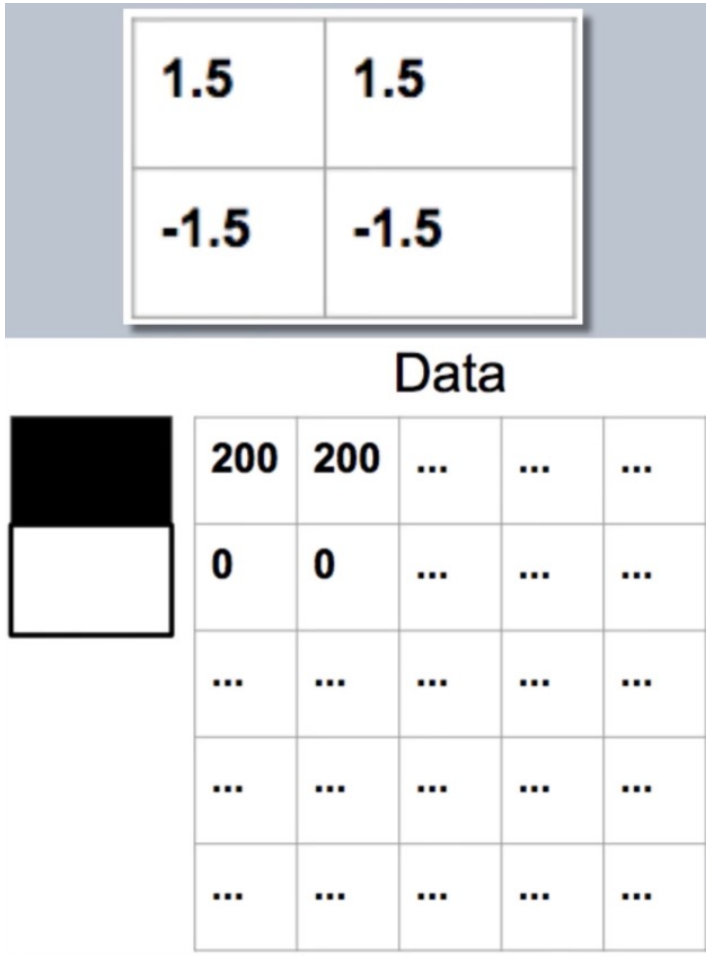
CODEBUGGED

1.5*0 + 1.5*0 – 1.5*0 – 1.5*0



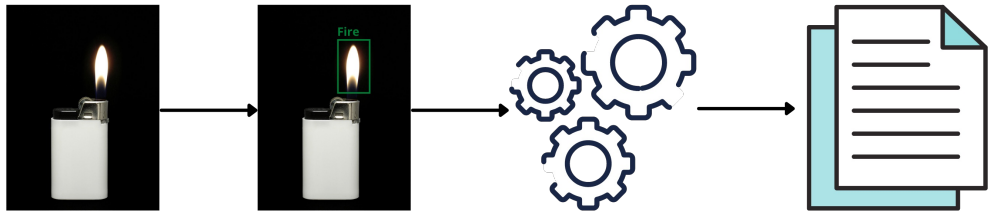
CODEBUGGED

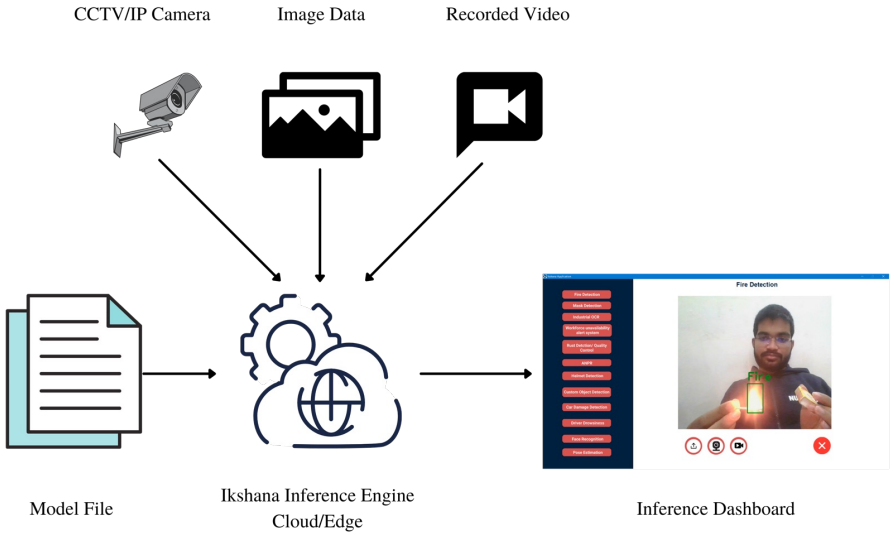
1.5*200 + 1.5*200 – 1.5*0 – 1.5*0



CODEBUGGED

Working of a Computer Vision Program





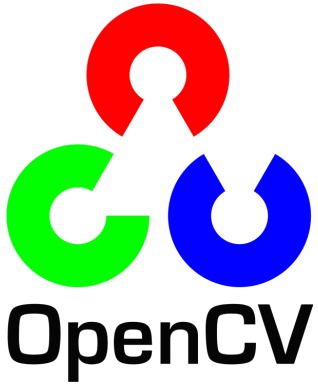
Model Training Engine

- The model training engine helps you to train an object detection model without any coding
- The user needs to drag and drop the image data for annotation and then run the engine for his choice of baseline model
- The model can be downloaded and exported at required site

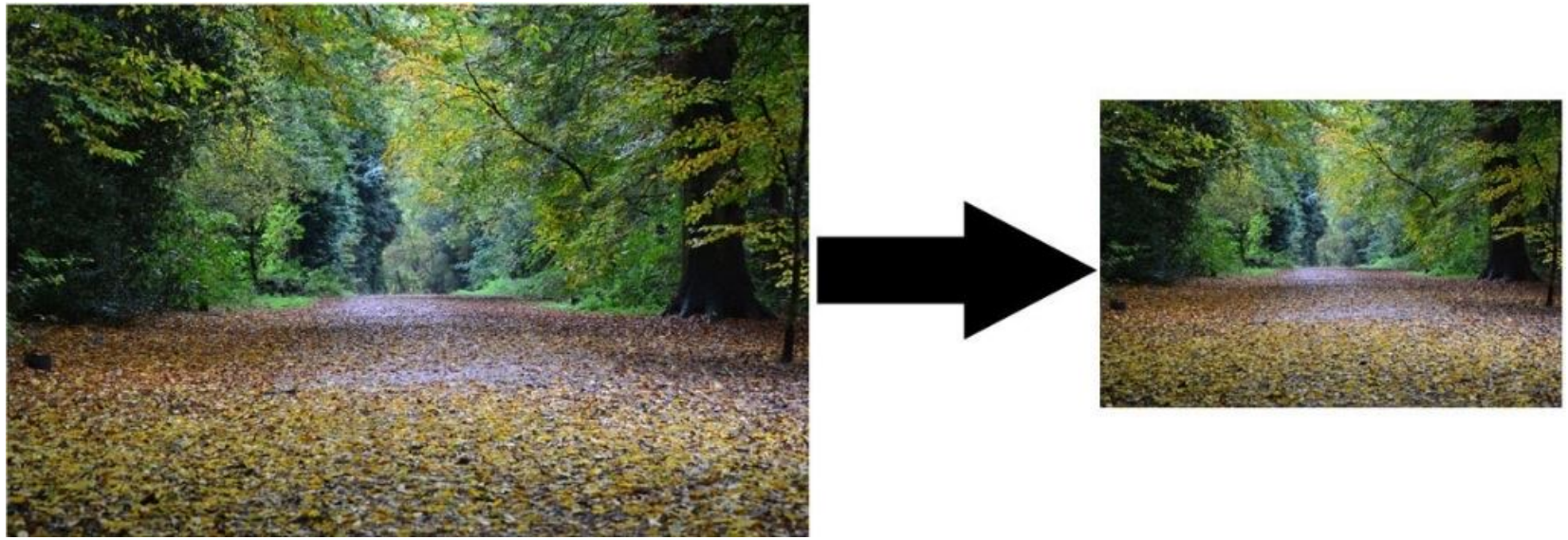
Inference Engine

- Inference engine supports inference on live CCTV feed, images and recorded videos.
- It provides expedite visual analytics on the dashboard
- It supports your custom model as well has pre trained models for top enterprise use cases

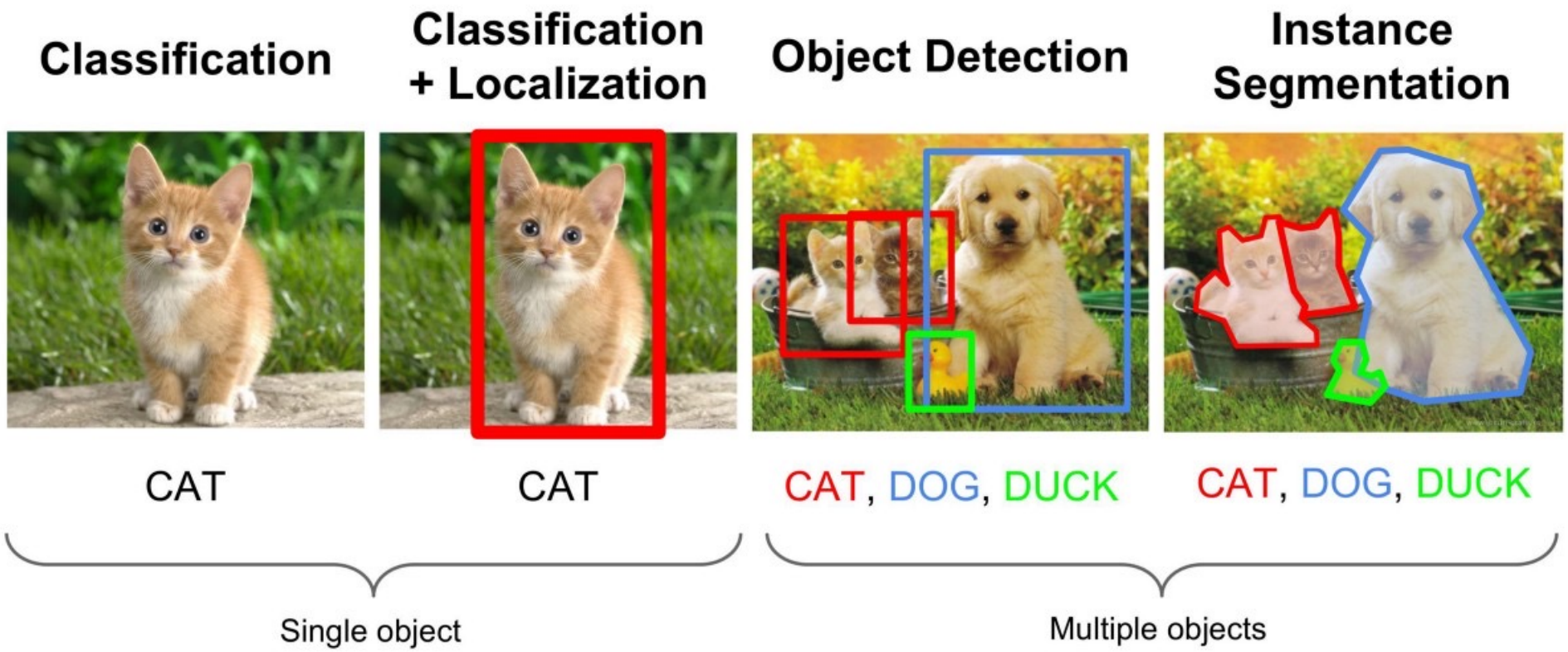
Computer Vision Libraries



Geometric Transformation



A set of image transformations where the geometry of image is changed without altering its actual pixel values are commonly referred to as “Geometric” transformation.

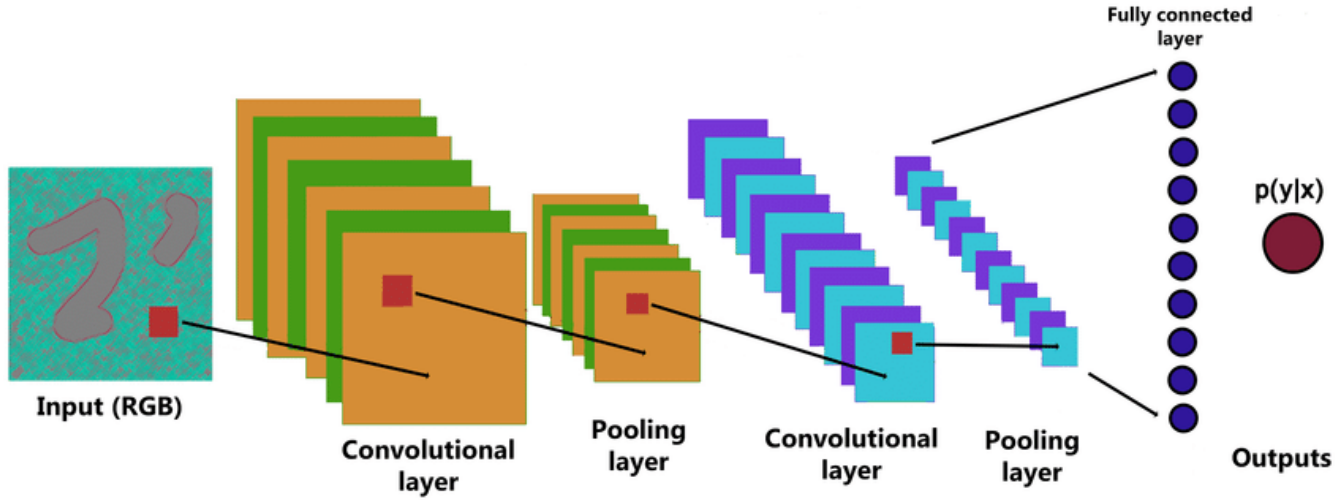


Object Detection

- Object detection is **a computer vision technique for locating instances of objects in images or videos**.
- Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results.

Convolutional Neural Network

A CNN is a type of neural network that is particularly well-suited for image classification tasks. It consists of several layers, including convolutional layers, pooling layers, and fully connected layers.



Here is a brief overview of how a CNN works:

Components of a CNN

1. **Convolutional layers**: The first layer of a CNN is typically a convolutional layer, which applies a set of learnable filters to the input image. Each filter is a small matrix that slides over the image to perform convolution, producing a set of feature maps. The feature maps represent different aspects of the image, such as edges and textures, and are learned through the training process.
2. **Activation functions**: Each feature map is then passed through an activation function, such as the Rectified Linear Unit (ReLU), which introduces nonlinearity into the network.
3. **Pooling layers**: After each set of convolutional and activation layers, a pooling layer is typically added to the network. Pooling layers reduce the spatial dimensions of the feature maps by taking the maximum, average, or other statistic of a group of values. This helps to reduce the computational cost of the network and make it more robust to small variations in the input image.
4. **Fully connected layers**: The output of the pooling layers is then flattened and passed through one or more fully connected layers, which act as a classifier. The fully connected layers take in the features extracted by the convolutional and pooling layers and output a probability distribution over the classes in the dataset.

layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3))

layers: This is a reference to the layers module of the Keras API, which is used to define the architecture of the neural network.

Conv2D: This is the type of layer we're defining, which is a 2D convolutional layer. Convolutional layers are commonly used in image classification tasks to extract features from images.

32: This is the number of filters (also known as "channels" or "kernels") in the layer. Each filter learns to recognize a specific feature in the input image.

(3, 3): This is the size of the filter. In this case, the filter is a 3x3 matrix that slides over the input image to perform convolution.

activation='relu': This is the activation function used by the layer. The Rectified Linear Unit (ReLU) function is commonly used in convolutional neural networks to introduce nonlinearity into the network.

input_shape=(150, 150, 3): This specifies the shape of the input to the layer. In this case, the input is a 150x150 RGB image (i.e., 3 color channels).

Why have I chosen 32 as number of filters ?

- The number of filters in a convolutional layer is a hyperparameter that needs to be tuned during the model development process. The value of the hyperparameter will depend on the complexity of the task and the size of the input data.

- In general, it is common to start with a smaller number of filters in the initial layers and gradually increase the number of filters in deeper layers. This is because the initial layers are responsible for learning lower-level features, such as edges and textures, which are more general and can be reused across different images. Deeper layers are responsible for learning higher-level features, such as object parts and shapes, which are more specific to the task and may require more filters.

- In the example line of code I provided, **32** is a relatively common number of filters to use as a starting point for an image classification task. However, this is not a hard and fast rule, and the optimal number of filters for a given task will depend on the specifics of the data and the model architecture.

- In practice, you may need to experiment with different values of the number of filters, as well as other hyperparameters, such as the learning rate, optimizer, and dropout rate, to find the optimal model for your task.

Why mostly batch size is in 2^n and same is with number of filters ?

- Choosing a batch size that is a power of two, such as 32, 64, or 128, is not a hard requirement for training neural networks, but it has become a common practice due to several practical benefits.
- One reason is that using a power of two can allow for more efficient memory allocation and usage on modern computer hardware. Many systems, including GPUs, operate more efficiently with power-of-two sizes because they can use bit-shifting operations instead of more complex integer division and modulo operations. This can lead to faster training times and more efficient use of resources.
- Another reason for using powers of two for the batch size is that it can help ensure that the training process is stable and that the model converges correctly. Specifically, using a power of two can help ensure that the number of samples in each batch is evenly distributed among the multiple GPUs or CPU cores that are used for training. This can prevent situations where some processors are idle while others are overloaded, which can lead to slower training times and unstable performance.
- Similarly, using powers of two for the number of filters in a convolutional layer is also a common practice. This is partly because it can simplify the design and implementation of the network architecture, making it easier to reason about and debug. Additionally, using powers of two can help ensure that the model has a symmetric and balanced architecture, which can improve its ability to learn useful features and generalize to new data

RULES but not BIBLE

Activation functions:

- Use ReLU (Rectified Linear Unit) as the default activation function for hidden layers.
- Use sigmoid or softmax activation functions for the output layer, depending on the nature of the problem you are trying to solve.
- Use tanh activation function if the input data has negative values and there is a need for zero centering.
- Avoid using very small activation functions like sigmoid or tanh for deep networks, as they can cause vanishing gradient problem.

Zero Centering

- Zero centering is a preprocessing step often used in machine learning and deep learning, where the mean of the data is subtracted from each feature or pixel in the input data. This results in a new dataset with a mean of zero, and this step is done to help improve the training and convergence of machine learning algorithms.
- Zero centering is useful because the gradient descent optimization algorithm used in many machine learning models works better when the input features are centered around zero. This is because the gradient descent algorithm tries to find the minimum of a loss function, and it moves in the direction of steepest descent. If the input data is not centered around zero, the optimization algorithm might take longer to converge to the minimum, or it might even get stuck in a local minimum.
- By subtracting the mean from each feature or pixel, zero centering ensures that the data is roughly symmetric around zero, which can help the optimization algorithm to converge faster and more reliably.

Vanishing Gradient Problem

- The vanishing gradient problem is a common issue in deep neural networks, particularly in networks with many layers. During the backpropagation process, gradients are computed and propagated backwards through the layers of the network in order to update the model's parameters. The problem occurs when the gradients become very small as they are propagated backwards through the network. When the gradients become too small, the weights of the earlier layers in the network are not updated effectively and the network fails to learn.
- The vanishing gradient problem is caused by the chain rule of differentiation. In a deep neural network with many layers, the gradient of the loss function with respect to the weights of the earlier layers is calculated by multiplying the gradients of the later layers together. Since the gradients are typically less than 1, multiplying them together can cause the gradients to become exponentially small as they are propagated backwards through the network. This means that the earlier layers of the network may not learn effectively.
- The vanishing gradient problem can be mitigated by using techniques such as weight initialization, batch normalization, and skip connections. Weight initialization involves setting the initial weights of the network in a way that prevents the gradients from becoming too small. Batch normalization involves normalizing the inputs to each layer of the network to have zero mean and unit variance, which can help stabilize the gradients. Skip connections involve adding connections that bypass some of the layers in the network, which can help prevent the gradients from becoming too small.

Optimizers:

- Use Adam as the default optimizer, as it is a good all-purpose optimizer that usually works well for most problems.
- Use SGD with momentum or Nesterov momentum if you are training a deep network with a large number of parameters.
- Use Adagrad or Adadelta if the gradient varies significantly from one feature to another, which makes it difficult for other optimizers to make progress.

Loss functions:

- Use binary cross-entropy loss for binary classification problems.
- Use categorical cross-entropy loss for multi-class classification problems.
- Use mean squared error (MSE) loss for regression problems.
- Use mean absolute error (MAE) loss for regression problems where outliers can skew the result.

Metrics:

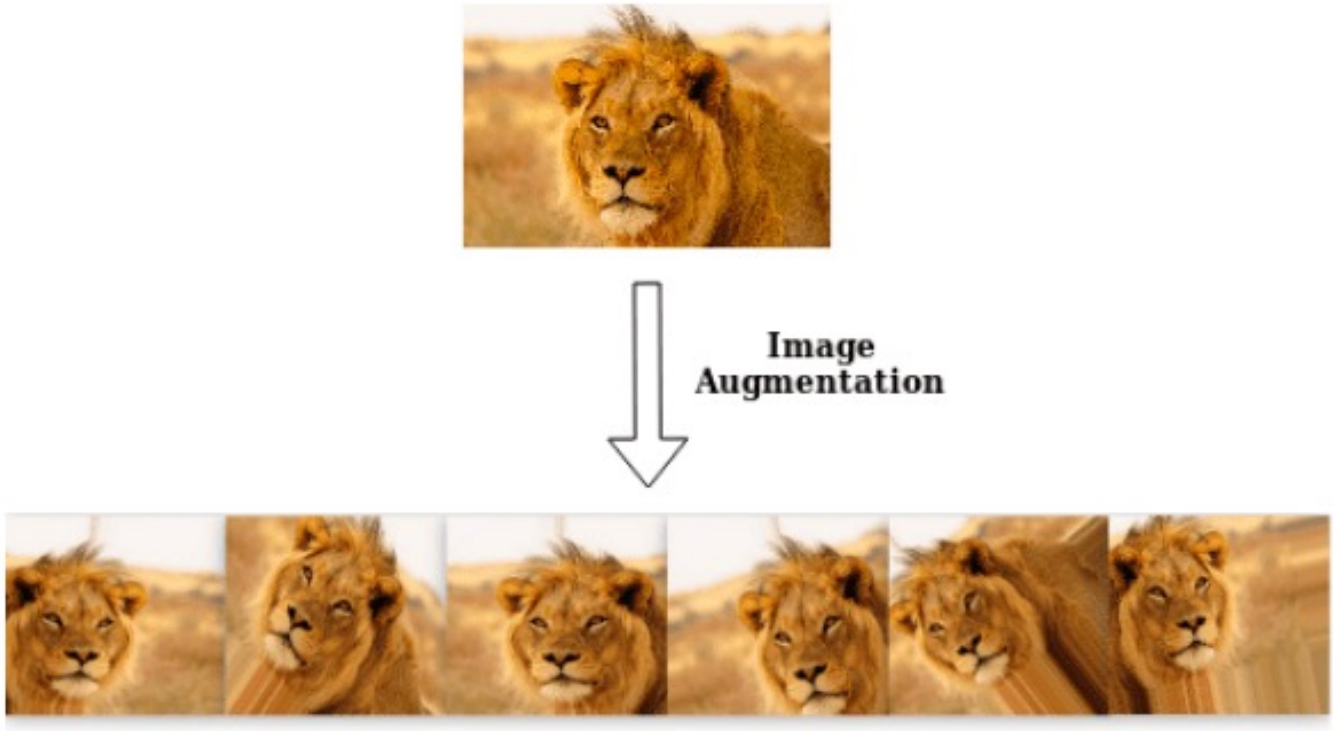
- Use accuracy for classification problems, as it is a commonly used metric and easy to interpret.
- Use precision and recall if the problem requires you to balance false positives and false negatives, such as in a medical diagnosis.
- Use F1 score if you want to balance both precision and recall.
- Use mean squared error (MSE) or mean absolute error (MAE) for regression problems.

Concept of Transfer Learning

- Transfer Learning is **“Re-training”** a pre trained model.
- Used when our data is less and it gives high accuracy result
- Early layers identify simple shapes, further layers identify more complex shapes and last layer does the prediction
- We will replace the final layer that is used to make predictions

CODEBUGGED

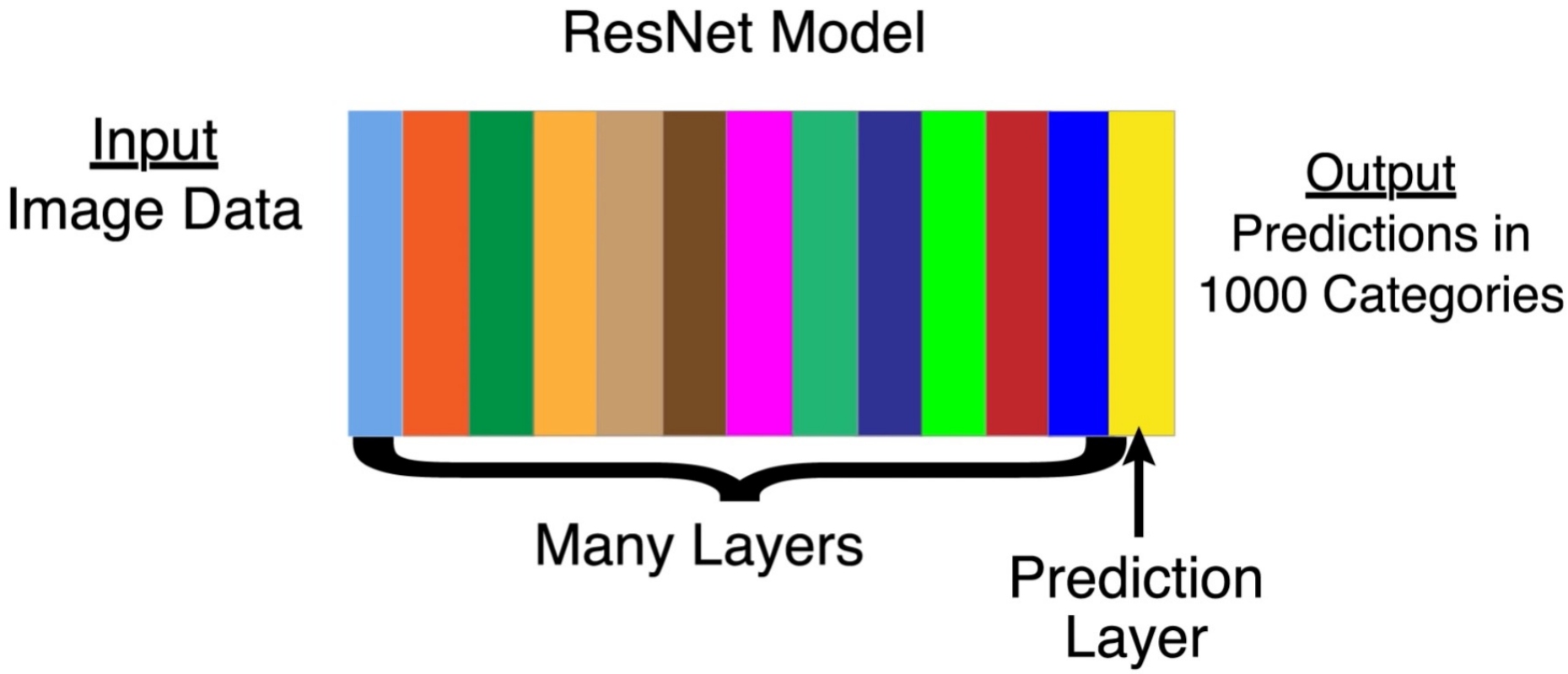
Data Augmentation



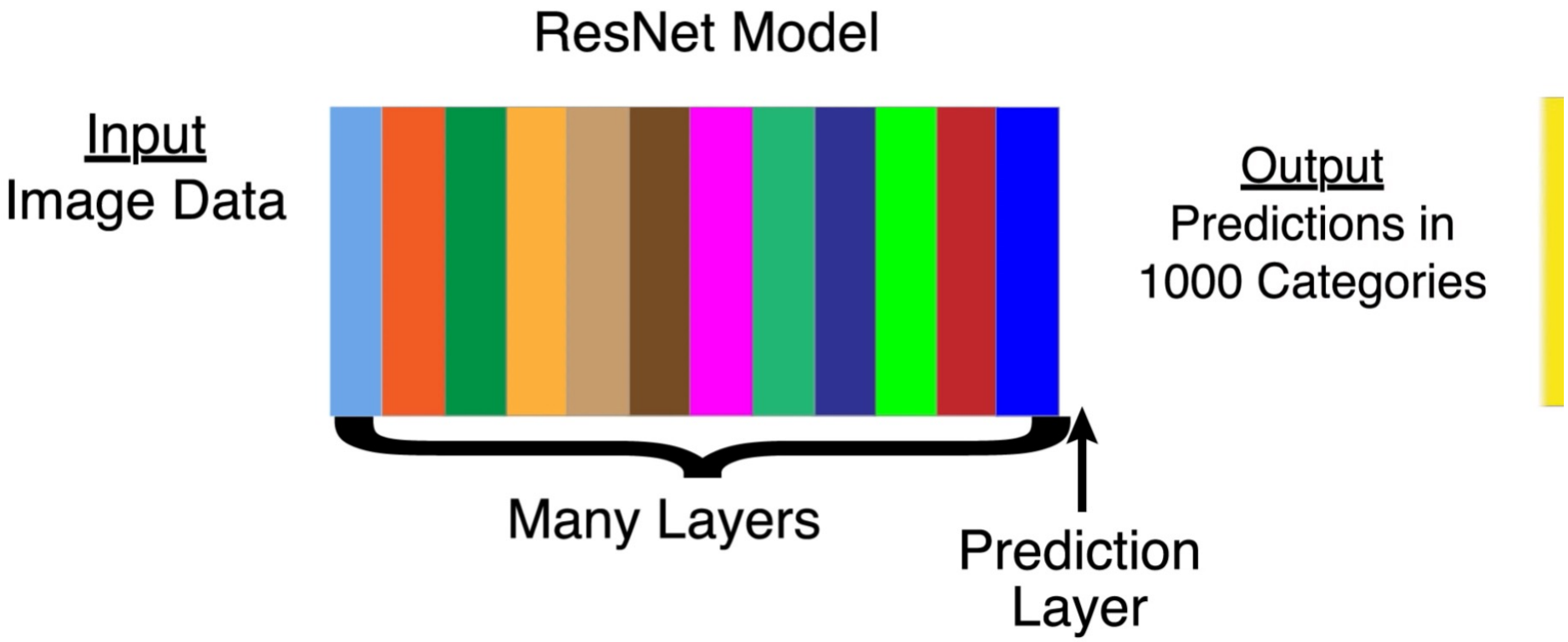
CODEBUGGED

Where it fails ?

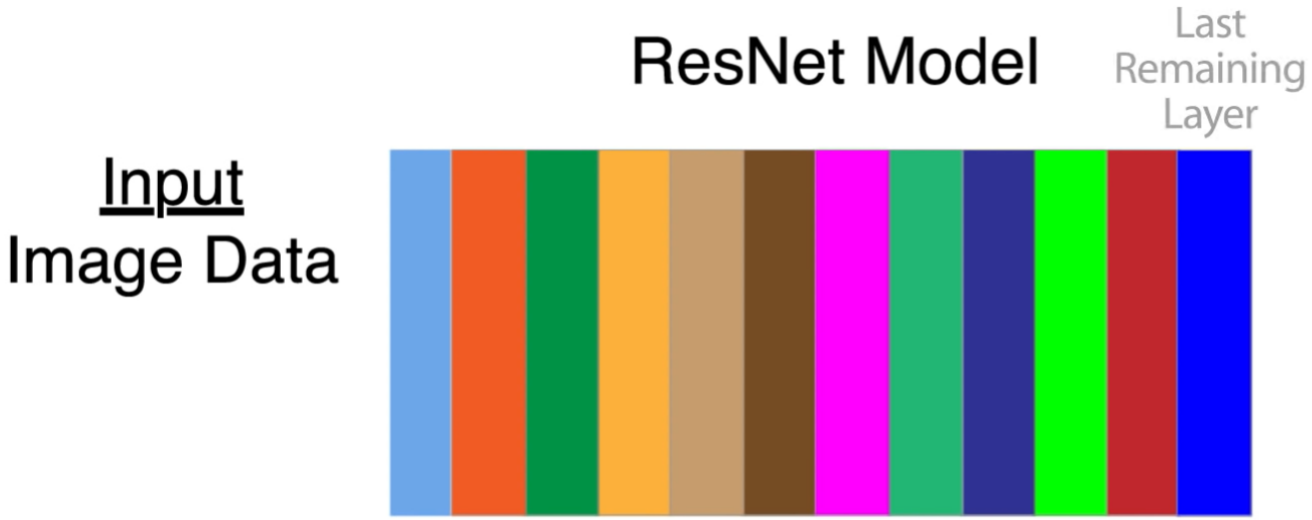
CODEBUGGED



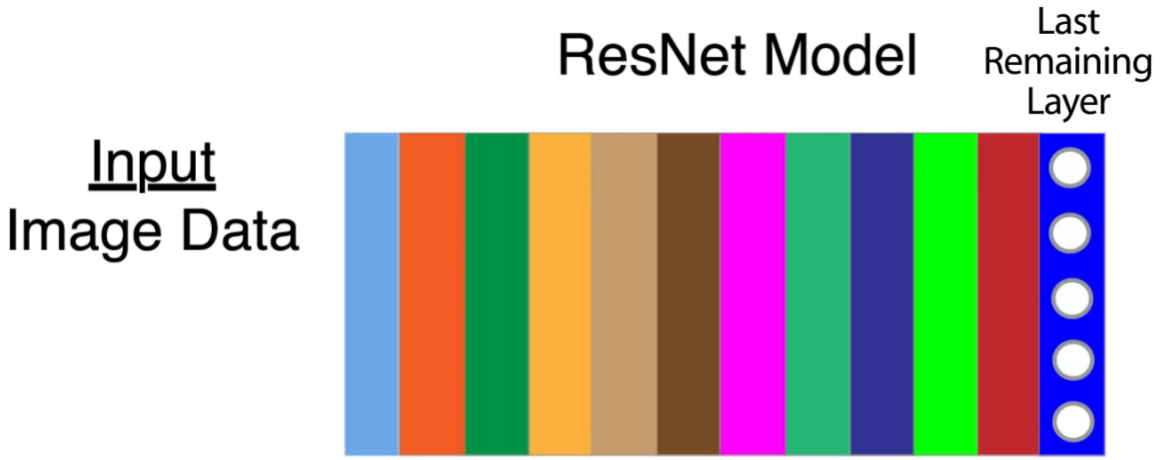
CODEBUGGED



CODEBUGGED



CODEBUGGED



CODEBUGGED

Process Scheduling and Optimization

Process scheduling is a method used in systems to manage the execution of processes in a system. It is essential for optimizing system performance, ensuring fairness, and facilitating effective resource sharing. The scheduler, a component of the operating system, is responsible for selecting the next process to run from a queue of available processes.

- **First-Come, First-Served (FCFS)**
- **Principle**: Processes are executed in the order they arrive.
- **Example**: If processes P1, P2, and P3 arrive in that order, they will be executed in the same order: P1 → P2 → P3.

- **Shortest Job Next (SJN) / Shortest Job First (SJF)**
- **Principle**: The process with the shortest burst time is executed next.
- **Example**: If P1 has a burst time of 10ms, P2 has 5ms, and P3 has 7ms, the execution order will be: P2 → P3 → P1.

- **Priority Scheduling**
- **Principle**: Each process is assigned a priority, and the process with the highest priority is executed next.
- **Example**: If P1 has priority 3, P2 has priority 1, and P3 has priority 2, the execution order will be: P2 → P3 → P1.

- **Round Robin (RR)**
- **Principle**: Each process is assigned a fixed time slice or quantum. Processes are executed in a cyclic manner, utilizing their time quantum.
- **Example**: If the time quantum is 4ms, and P1, P2, and P3 have burst times of 10ms, 5ms, and 7ms respectively, the execution might look like: P1 (4ms) → P2 (4ms) → P3 (4ms) → P1 (4ms) → P2 (1ms) → P3 (3ms) → P1 (2ms).

Optimization

- Optimization in process scheduling involves tweaking the scheduling algorithms or parameters to improve system performance. It might involve:
- **Minimizing waiting time**: Adjusting the scheduling to reduce the total waiting time of all processes.
- **Maximizing throughput**: Enhancing the scheduling to increase the number of processes that complete execution per unit time.
- **Balancing load**: Distributing the processes evenly across all available resources to avoid overloading a single resource

Question 1: Process Scheduling Analysis

Given the following processes with their arrival times and burst times, schedule the processes using FCFS, SJF, PS, and RR (with a time quantum of 4 units) algorithms. After scheduling, determine the waiting time and turnaround time for each process in each scheduling algorithm. Finally, analyze which algorithm is the most suitable based on the average waiting time and average turnaround time.

Process	Arrival Time	Burst Time	Priority
P1	0	24	3
P2	4	3	1
P3	5	3	4
P4	6	12	2

Question 2: Real-World Scheduling Scenario

Imagine a busy hospital emergency room where patients arrive at different times and have different levels of urgency. Given the data below, use the FCFS, SJF, PS, and RR scheduling algorithms to determine the order in which patients should be attended to. After applying each algorithm, discuss which one provides the most efficient and fair scheduling.

Patient	Arrival Time	Estimated Treatment Time	Urgency Level (1-5, 5 being most urgent)
A	00:00	30 minutes	3
B	00:10	20 minutes	5
C	00:15	40 minutes	2
D	00:20	15 minutes	4

Theory of Sensors

- IoT sensors are devices that collect data from their environment and send it to other devices or systems, typically over the internet. They play a crucial role in the Internet of Things (IoT) ecosystem.
  - **IoT Sensors - In Short:**
  - **Data Collection**: Capture information from the environment.
  - **Connectivity**: Transmit data to other devices or central systems, often wirelessly.
  - **Real-time Monitoring**: Provide continuous or periodic data updates.
  - **Diverse Applications**: Used in various sectors like agriculture, healthcare, transportation, and home automation.
  - **Miniaturization**: Often small and energy-efficient to fit various applications and environments.
  - **Integration**: Can be embedded in other devices or systems.
- Common Types of IoT Sensors:
  - **Temperature Sensors**: Measure heat or cold.
  - **Humidity Sensors**: Measure moisture levels in the air.
  - **Motion Sensors (PIR)**: Detect movement in a given area.
  - **Proximity Sensors**: Measure how close an object is to the sensor.
  - **Gas Sensors**: Detect levels of various gases in the environment.
  - **Pressure Sensors**: Measure force exerted on a surface.
  - **Light Sensors (Photodetectors)**: Measure light intensity.
  - **Water Quality Sensors**: Monitor parameters like pH, turbidity, and dissolved oxygen.
  - **Gyroscope & Accelerometers**: Measure orientation and movement.

Theory of Actuators

- Actuators and sensors are both fundamental components in many systems, especially in the realm of IoT. While they are closely related in function, they serve opposite roles.
- **Actuators:**
  - **Function**: Actuators perform actions or induce changes in a system based on received signals or commands.
  - **Types**: Common types include motors (that can turn or move), solenoids (that can push or pull), heaters (that can produce heat), and LEDs (that can produce light).
  - **Applications**: Used in systems that require physical actions, such as opening a valve, adjusting the temperature, or turning on a light.

- **1. MQTT (Message Queuing Telemetry Transport)**
- **Communication Model**: Publish/Subscribe
- **Use Cases**: Remote monitoring, telemetry, real-time analytics, mobile applications.
- **When to Use**: When you need a lightweight messaging protocol with low bandwidth requirements, especially over unreliable networks or remote locations.

- **2. CoAP (Constrained Application Protocol)**
- **Communication Model**: Request/Response (similar to HTTP but for constrained devices)
- **Use Cases**: Smart home, building automation, web of things applications.
- **When to Use**: When you need a web-based protocol for constrained nodes and networks, with low overhead and parsing complexity.

- **3. HTTP/HTTPS**
- **Communication Model**: Request/Response
- **Use Cases**: Web services, cloud integration, data presentation.
- **When to Use**: When integrating with web services or cloud platforms, or when data security (via HTTPS) is a priority.

- **4. WebSocket**
- **Communication Model**: Full-duplex communication over a single, long-lived connection.
- **Use Cases**: Real-time applications, web-based real-time monitoring, chats.
- **When to Use**: When real-time, bidirectional communication between a client (often a web browser) and server is required.

- **5. AMQP (Advanced Message Queuing Protocol)**
- **Communication Model**: Peer-to-Peer, Publish/Subscribe, Request/Response
- **Use Cases**: Business messaging, cloud solutions, cross-platform communication.
- **When to Use**: When you need a reliable and secure messaging protocol with advanced features like message orientation, queuing, routing, and more.

- **6. LoRaWAN (Long Range Wide Area Network)**
- **Communication Model**: Star topology, where devices communicate with a centralized gateway.
- **Use Cases**: Smart cities, agriculture, supply chain, outdoor monitoring.
- **When to Use**: For long-range, low-power applications where devices send data infrequently.

- **7. Zigbee**
- **Communication Model**: Mesh networking
- **Use Cases**: Home automation, smart energy, healthcare.
- **When to Use**: When you need a low-power, low-data rate communication in personal area networks, and benefit from mesh networking for reliability.

- **8. Bluetooth & Bluetooth Low Energy (BLE)**
- **Communication Model**: Point-to-point, point-to-multipoint, and mesh (for BLE 5.0 and above).
- **Use Cases**: Wearables, health monitors, smart homes, proximity sensing.
- **When to Use**: For short-range communication, especially when power efficiency is crucial.

- **9. 6LoWPAN**
- **Communication Model**: Mesh networking
- **Use Cases**: Wireless sensor networks, home and building automation.
- **When to Use**: When you need to transmit IPv6 packets over low-power wireless networks like Zigbee.

- **10. NB-IoT (Narrowband IoT)**
- **Communication Model**: Point-to-point
- **Use Cases**: Smart cities, agriculture, industrial monitoring.
- **When to Use**: For applications that require wide-area coverage, deep penetration, and low power consumption.

- **11. Thread**
- **Communication Model**: Mesh networking
- **Use Cases**: Home automation, connected devices.
- **When to Use**: When you need a low-power, secure, and scalable mesh network for smart homes.

- **12. Z-Wave**
- **Communication Model**: Mesh networking
- **Use Cases**: Home automation, security systems.
- **When to Use**: For residential control and automation applications, especially when interoperability between devices is essential.

- **13. DDS (Data Distribution Service)**
- **Communication Model**: Publish/Subscribe
- **Use Cases**: Real-time systems, autonomous vehicles, medical devices.
- **When to Use**: When you need real-time, high-performance communication with Quality of Service (QoS) guarantees.

Unleash the power of AI in your Enterprise

Thank You

sarthak@codebugged.com
+91-8004006979

www.codebugged.com