

**CNN MODEL –**

**Waste Image Classification**

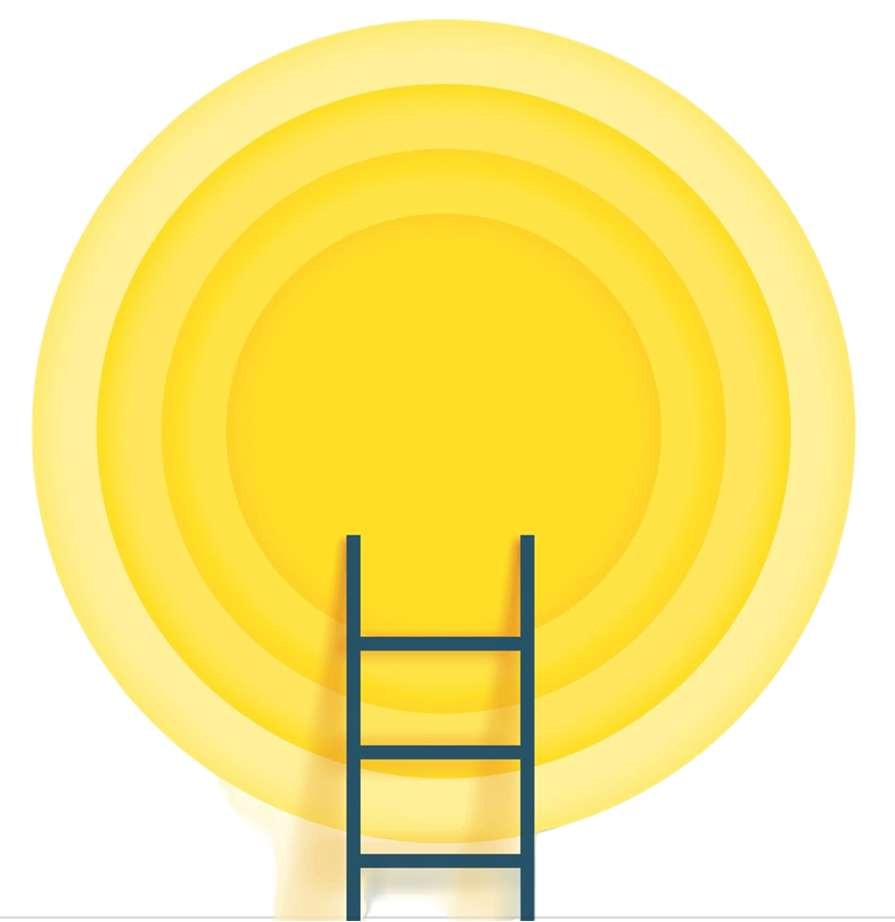
ABHSIHEK KATARYA – 7th Feb 2025

# Learning Objectives

**CNN is supervised learning technique for finding patterns in images to classify and recognize objects, classes and categories. CNN architecture has three layers Convolution layers, Rectified linear unit, and Pooling layers. CNN also use to classify audio, time series and signal data.**

* **Understanding CNN architecture and its layers.**
* **Learning how CNNs classify images, audio, and time- series data.**
* **Exploring tools and technologies used in CNN.**
* **Implementing a CNN model for image classification.**

**GOAL**



**Source :** **https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/**

**Data set:** **https://www.kaggle.com/datasets/techsash/waste-classification-data**

**Goals**

* **CNN** is a **supervised learning technique** used to recognize and classify objects in images.

## The architecture consists of three main layers:

* **Convolutional Layers** – Extract features from images.
* **ReLU Activation Layers** – Introduce non-linearity.

## **Pooling Layers** – Reduce spatial dimensions while retaining important features.

* CNNs are also applied in **audio processing, time-series analysis, and signal data classification**

# Tools and Technology used

* **Programming Language:** Python

### Libraries & Frameworks:

* TensorFlow / PyTorch
* OpenCV for image processing
* Matplotlib & Seaborn for visualization

### Hardware Requirements:

* GPU acceleration (Kaggle Cloud Resources – Free Tier)
* Cloud computing platforms (Streamlit)

# Methodology

### Import Libraries

* + Use TensorFlow or PyTorch for deep learning, OpenCV for image processing, and Matplotlib for visualization.

1. **Organize Dataset**
   * Structure Directories:

dataset/

├── train/

│ ├── class\_1/

│ └── class\_2/

├── validation/

├── test/

* + Ensure balanced class distribution for training, validation, and test sets.

1. **Preprocess Images**
   * Resize Images: Scale all images to a fixed size, e.g., 224x224 pixels.
   * Normalize Pixels: Scale pixel values to [0, 1]:
   * Augmentation: Apply transformations for variability (rotation, flipping, cropping):

### Load Data Using Generators

* + Dynamically load batches of images during training:

### Build CNN Model

* + Design a model with convolutional layers to extract features and dense layers for classification:

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(224, 224, 3)))

model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(128, activation='relu’))

model.add(Dense(num\_classes, activation='softmax’))

### Train the Model

* + Compile with Adam optimizer and categorical cross-entropy loss:
  + Train the model:

### Evaluate and Analyze

* + Test performance on unseen data:
  + Generate a confusion matrix and classification report:

Problem statement :

* + **Objective:** Build a CNN model that classifies images based on their content.
  + **Challenges:**
  + Handling large datasets efficiently.
  + Avoiding overfitting in deep architectures.
  + Optimizing model performance on test data.

# Solution:

* + CNN models use: **Convolutional layers** to detect image features.
  + **Pooling layers** to reduce spatial dimensions.
  + **Fully connected layers** to classify images.
  + **Regularization techniques** to prevent overfitting.
  + **Optimization algorithms** like Adam or SGD

# Conclusion:

* + - CNNs **revolutionized computer vision** and are widely used in:
      * **Object Recognition**
      * **Medical Image Analysis**
      * **Facial Recognition**
      * **Art & Style Transfer**
    - **Future Enhancements:**
      * Adding **attention mechanisms** for better accuracy.
      * Implementing **self-supervised learning**.
      * Using **transfer learning** for improved performance.