# **Assignment 2: Implementing Facial Expression Recognition using Deep Learning**

## **Problem Statement**

The problem is to develop a **facial expression recognition system** that can classify human faces into two categories:

* **Smile**
* **Not Smile**

This task uses deep learning techniques in combination with computer vision to automatically detect facial features and determine whether a person is smiling.

## **Objective**

* To preprocess and prepare face images for deep learning.
* To build and train a **Convolutional Neural Network (CNN)** for binary classification (Smile vs. Not Smile).
* To use **OpenCV** for face detection and preprocessing.
* To evaluate the accuracy of the model on test images.

## **Requirements**

* **Operating System**: Windows / Linux / macOS (Colab recommended)
* **IDE / Platform**: Google Colab / Jupyter Notebook

### **Libraries and Packages Used**

* **TensorFlow / Keras** → Deep learning model creation
* **OpenCV** → Face detection and preprocessing
* **NumPy** → Numerical computations
* **Matplotlib / Seaborn** → Visualization of results
* **Scikit-learn** → Evaluation metrics

## **Theory**

### **Definition**

**Facial Expression Recognition (FER)** is a computer vision task that identifies human facial expressions from images or videos. In this assignment, the focus is specifically on **binary classification**: detecting whether a person is smiling or not.

### **Structure**

1. **Face Detection** – Detect faces from images using OpenCV’s **Haar Cascade Classifier**.
2. **Image Preprocessing** – Convert detected face region to grayscale, resize to fixed size, normalize pixel values.
3. **CNN Model** – Extracts features and classifies expression into *Smile* or *Not Smile*.
4. **Output Layer** – A sigmoid activation function outputs probability of “smile.”

## **Methodology**

1. **Dataset Preparation**
   * Collect facial images with two labels: **smiling** and **not smiling**.
   * Detect and crop faces using OpenCV’s Haar Cascade.
   * Convert to grayscale, resize (e.g., 64×64), and normalize pixel values.
2. **Model Construction**
   * Build a Sequential CNN model with convolution + pooling layers.
   * Flatten feature maps and pass through dense layers.
   * Use **sigmoid activation** in the output layer for binary classification.
3. **Training**
   * Compile model using optimizer (Adam) and binary crossentropy loss.
   * Train on labeled facial expression dataset.
   * Use data augmentation (flipping, zoom, rotation) to generalize better.
4. **Evaluation**
   * Test model on unseen face images.
   * Evaluate with **accuracy, precision, recall, and confusion matrix**.

## **Advantages**

* Automates expression detection without manual feature engineering.
* Works well in real-time applications using **OpenCV**.
* High accuracy achievable with CNNs.
* Useful in **healthcare, security, human-computer interaction, and entertainment**.

## **Limitations**

* Performance depends on **lighting conditions** and **image quality**.
* May not generalize well on **different ethnicities or age groups** without diverse data.
* Sensitive to occlusions (glasses, masks, hand covering face).
* Training requires a sufficiently large and balanced dataset.

## **Working / Algorithm**

1. Input face image captured/detected with OpenCV.
2. Preprocess image (grayscale → resize → normalize).
3. CNN extracts hierarchical features (edges → facial curves → smile patterns).
4. Flattened features pass through dense layers.
5. Sigmoid output layer predicts:  
   * **0 = Not Smile**
   * **1 = Smile**
6. Final classification based on threshold (≥0.5 = Smile).

## **Conclusion**

This assignment successfully demonstrates **Facial Expression Recognition (Smile Detection)** using deep learning and computer vision. The CNN model effectively learned features of smiling and non-smiling faces, achieving reliable classification accuracy. This project highlights the role of AI in **emotion recognition systems, smart devices, and real-time monitoring applications**.