**Regular project Questions**

**●    Project Title:** Face Recognition and Detection Using Convolutional Neural Networks (CNN)  
**●    Description:**In this project, you will build a Convolutional Neural Network (CNN) model for face recognition and detection. You will work with a facial recognition dataset to implement a system that can accurately identify and verify faces in images. This project will require you to preprocess the dataset, build and train a CNN model, implement face detection, and evaluate the model's performance using appropriate metrics.  
**Part 1:** Face Detection and Dataset Preprocessing  
**●    1.1 Face Detection**  
○    Utilize OpenCV's DNN module for face detection.  
○    Pre-trained face detection model files (.prototxt and .caffemodel) are required.  
○    Confidence threshold set to 0.5.  
○    Support for multiple face detections in an image.  
**●    1.2 Dataset Preprocessing**  
○    Load the LFW dataset for face recognition.  
○    Convert grayscale images to RGB and resize them to (299, 299) for compatibility with InceptionV3.  
○    Normalize pixel values for consistent input processing.  
○    Split the dataset into training (80%), validation (20% of training), and test (20%) sets.  
○    Implement data augmentation techniques: rotation, width/height shifts, and horizontal flips.  
**Part 2:** Face Recognition with Inception V3 Transfer Learning  
**●    2.1 Load and Modify Inception V3**  
○    Use TensorFlow/Keras Inception V3 with pre-trained weights on ImageNet.  
○    Modify model architecture:  
■    Add a Global Average Pooling layer.  
■    Include a Dense layer (128 neurons, ReLU activation).  
■    Apply Dropout (0.5) for regularization.  
■    Use a final Dense layer with softmax activation for classification.  
■    Freeze base model layers to retain pre-trained feature extraction.  
**●    2.2 Training the Model**  
○    Optimizer: Adam with a learning rate of 0.001.  
○    Loss function: Categorical crossentropy.  
○    Metrics: Accuracy.  
○    Implement callbacks:  
■    Early stopping to monitor validation loss with a patience of 5 epochs.  
■    Learning rate reduction upon validation loss plateau (factor 0.1, patience 3).  
**●    2.3 Optimization Techniques**  
○    Convert and save the trained model in ONNX format.  
○    Apply TensorFlow Lite (TFLite) quantization for efficient mobile deployment.  
○    Extract feature embeddings from the trained model for similarity-based recognition.  
**Part 3:**Model Deployment and Enhancements  
**●    3.1 Non-Classified Face Recognition**  
○    Use cosine similarity to compare embeddings of detected faces.  
○    Set a similarity threshold (0.5) to classify matches.  
○    Introduce a "Non-Defined" class for unidentified individuals.  
**●    3.2 Live Camera Integration**  
○    Integrate real-time face recognition using OpenCV.  
○    Support image uploads and webcam streaming for detection.  
○    Display recognition results with names and confidence scores.  
**●    3.3 Streamlit Project**  
○    Develop an interactive UI using Streamlit.  
○    Provide options to:  
■    Upload images for recognition.  
■    Use live camera for real-time detection.  
■    Display classification results, including confidence levels.  
○    Deploy the application on a web server with ONNX-based inference support.

Dataset: https://www.kaggle.com/datasets/hereisburak/pins-face-recognition