Architecture Design for CNN Image Classification

1. Input Layer

- Input Dimensions: Images with dimensions 256x256x3 (height, width, and color channels).
- Purpose: Accepts raw input images for preprocessing and feeding into the model.

2. Data Preprocessing

- Normalization: Pixel values are scaled to the range [0, 1] to improve model performance.
- Augmentation (Optional): Use techniques like rotation, flipping, zooming, and shearing

to increase dataset diversity during training.

- Batching: Images are processed in batches of size 32.

3. Convolutional Neural Network (CNN)

- Layer Block 1:
  - Convolutional Layer 1: 32 filters, kernel size (3x3), ReLU activation.
  - MaxPooling Layer 1: Pool size (2x2), stride 2.
- Layer Block 2:
  - Convolutional Layer 2: 64 filters, kernel size (3x3), ReLU activation.
  - MaxPooling Layer 2: Pool size (2x2), stride 2.

- **Layer Block 3:**
  - **Convolutional Layer 3: 128 filters, kernel size (3x3), ReLU activation.**
  - **MaxPooling Layer 3: Pool size (2x2), stride 2.**

4. Fully Connected Layers

- **Flatten Layer: Converts 3D feature maps into a 1D feature vector.**
- **Dense Layer 1: 128 neurons, ReLU activation.**
- **Dense Layer 2: 64 neurons, ReLU activation.**
- **Output Layer: 1 neuron, sigmoid activation (for binary classification).**

5. Output

- **Prediction: Binary classification output (e.g., 0 for Class A, 1 for Class B).**

System Architecture

1. User Interaction

- **Input: Users upload an image through an application interface (web or mobile).**
- **Output: Prediction label and confidence score.**

2. Backend Workflow

1. **Image Upload: Image is sent to the backend for processing.**
2. **Model Invocation: Pre-trained CNN model processes the image to generate predictions.**
3. **Post-Processing:**

- Convert prediction probabilities into class labels.
- Generate confidence scores (e.g., 0.89 = 89% confidence).

3. Storage

- Model Storage: Save the trained model (drctorai.h5) on a secure backend server or cloud storage (e.g., AWS S3, GCP).
- Dataset Storage: Store datasets for retraining or fine-tuning in a scalable storage solution.

4. Deployment

- Cloud Server: Deploy the model using TensorFlow Serving or Flask/Django APIs.

- Infrastructure: Use cloud platforms like AWS, GCP, or Azure for hosting.
- Scaling: Enable autoscaling for handling user traffic.

5. Monitoring

- Performance Metrics:
  - Training/Validation Accuracy and Loss.
  - Inference time for predictions.
- Error Logging: Log issues for debugging (e.g., misclassifications).
- Visualization: Use tools like TensorBoard or custom dashboards.

Visualization of Architecture

1. **Input (Image Upload)**
   ⬇
2. **Preprocessing (Resize, Normalize, Batch)**
   ⬇
3. **CNN Model (Feature Extraction)**
   ⬇
4. **Fully Connected Layers (Decision Making)**
   ⬇
5. **Output (Class Prediction + Confidence Score)**
   ⬇
6. **Result Display (User Interface)**