Yoga Pose Detection using Deep Learning

1. Introduction:

The aim of this project was to build a deep learning-based system for detecting yoga poses and providing feedback on alignment and accuracy. The model was trained on a dataset of various yoga poses, and the goal was to offer corrections for users to improve their posture during yoga practice.

2. Approach:

This project follows the following major steps:

- **Data Collection & Preprocessing**: The dataset consists of images of various yoga poses such as **'Downdog**,' **'Tree**,' **'Warrior 2**,' etc. Images were preprocessed to ensure consistency in size and normalization.
- **Model Development**: We used a pre-trained MobileNetV2 model for feature extraction and added custom layers on top for pose classification.
- **Model Training & Evaluation**: The model was trained using a data generator for efficient image loading and augmentation. The performance was evaluated on a separate test set.

3. Data Preprocessing :

The images in the dataset were processed using the following steps:

1. **Resizing:** All images were resized to 224 x 224 pixels to match the input size expected by MobileNetV2.
2. **Normalization:** Pixel values were scaled between 0 and 1 to facilitate faster convergence during training.
3. **Data Augmentation:** Techniques like rotation, zooming, and horizontal flipping were used to increase the robustness of the model by artificially expanding the training dataset.
4. **Validation:** Corrupted or invalid images were removed using the Image.verify() function from the PIL library.

4. Model Architecture :

The model architecture is based on **MobileNetV2**, a lightweight convolutional neural network that performs well on mobile devices. The architecture consists of:

- Base Model (MobileNetV2): Pre-trained weights from the ImageNet dataset were used, and the top layer was excluded. This base model served as a feature extractor.
- Custom Layers:
    - Flatten Layer: To convert the output from the convolutional base into a 1D array.
    - Dense Layer (128 units): Added for classification, with ReLU activation.
    - Dropout Layer (0.5): Prevented overfitting by randomly dropping half of the neurons during training.
    - Output Layer: Softmax activation function was used for multi-class classification.

The model was compiled using the **Adam optimizer** and **categorical crossentropy loss**.

5. Model Training :

The model was trained for 10 epochs using the following setup:

- **Batch Size:** 32
- **Training Data:** Loaded using ImageDataGenerator with augmentation techniques.
- **Validation Data:** Loaded similarly but without augmentation.

The model **was trained using a small dataset to begin with**, with the aim to further expand and fine-tune once the initial results were satisfactory.

6. Results :

- The trained model was able to classify the poses with an accuracy of **96.38%** on the validation set.
- The predictions were evaluated based on accuracy and the model's performance improved with each epoch.
- The model also provided feedback on alignment for each pose, helping users correct their posture.

7. Results :

- **Fine-tuning the Model:** The model can be improved by unfreezing layers in the pre-trained MobileNetV2 and training the entire network for further performance improvement.
- **Real-time Pose Detection:** Implement a real-time pose detection feature using a webcam to give immediate feedback to users.
- **Increase Dataset Size:** Expand the dataset with more poses and variations to make the model more robust.
- **Deployment:** The final model can be deployed on mobile apps or the web to offer interactive yoga training assistance.
- **Integration with Wearables:** Integrate with wearable devices for more accurate real-time feedback.

8. Conclusion :

This project demonstrates how a deep learning-based model can be trained to detect yoga poses and provide valuable feedback for users. With further refinement and real-time deployment, the system has the potential to assist yoga practitioners worldwide.

9. Appendix (Code Explanation)

1. **Model Code Overview:**
    - The core of the model is MobileNetV2, a convolutional neural network that was trained on the ImageNet dataset and used as a feature extractor.
    - Custom layers were added on top to classify the poses into categories.

- The model is trained with augmentation techniques to generalize better to new poses.

2. **Preprocessing Code:**
   - The preprocessing involved resizing the images to a consistent size and normalizing pixel values to ensure that the data fed to the model is clean and standardized.

3. **Training and Evaluation:**
   - The model was trained using the ImageDataGenerator class, which allows efficient loading and augmentation of images.
   - After training, the model was evaluated on a test set to gauge its performance.

Dataset Used : https://www.kaggle.com/datasets/niharika41298/yoga-poses-dataset