Custom Convolutional Neural Network (CNN) Design in contrast with Pretrained Models

This assignment focuses on building and evaluating an image classification system using two approaches: a custom-designed convolutional neural network (CNN) and a pretrained CNN model. The goal is to understand differences in training time, accuracy, loss metrics, and classification performance reflected in confusion matrices. Kindly choose the images from different applications and realistic fields.

Part 1: Custom CNN Design

- Design a CNN architecture specifically for the image classification task. Include:
  - Input layer matching image dimensions.
  - Multiple convolutional layers with ReLU activations.
  - Pooling layers for spatial downsampling.
  - Fully connected layers leading to final classification output.
- Choose an optimizer (e.g., Adam) and appropriate loss function (categorical cross-entropy).
- Train the model on the dataset for a fixed number of epochs.
- Record training time and monitor training/validation accuracy and loss per epoch.

Part 2: Pretrained CNN Model

- Select a pretrained CNN (e.g., ResNet50, VGG16, or MobileNet) trained on ImageNet.
- Replace the top classification layer to match the number of classes.
- Fine-tune the model on the same dataset.
- Record training time, accuracy, and loss curves similarly.

Part 3: Performance Comparison

- Compare training durations between the custom CNN and pretrained model.
- Evaluate accuracy on validation and test datasets to check classification quality.
- Analyze loss functions' convergence behavior.
- Generate confusion matrices for each model to assess per-class performance, identifying strengths or weaknesses.

Part 4: Discussion and Conclusion

- Discuss how transfer learning in pretrained CNNs usually leads to faster convergence and improved accuracy due to learned feature representations.
- Highlight scenarios where custom CNNs might be more flexible or necessary, acknowledging likely longer training times and tuning requirements.
- Summarize findings on practical trade-offs between training efficiency and final model performance.

Additional Notes:

- Use appropriate libraries such as TensorFlow or PyTorch for implementation.
- Employ visualization tools to plot training/validation curves and confusion matrices.
- Ensure the dataset is properly preprocessed and split fairly.

By completing this assignment, trainees will gain hands-on experience in CNN design, training, evaluation, and the practical implications of leveraging pretrained networks versus building models from scratch. This knowledge is vital for applied machine learning in computer vision.

This assignment outline can be provided as a report or implemented in code notebooks for practical training sessions.