Problem Statement and Justification

Problem Proposal

The goal of this project is to create a real-time image classification system capable of distinguishing between three specific objects—Object 1, Object 2, and Object 3—using a live webcam feed. The system uses transfer learning with a pre-trained ResNet-50 model to achieve high accuracy while balancing computational demands.

Justification

Real-time object recognition is widely applicable in domains such as:

- Automated retail systems (e.g., product identification at checkout).
- Inventory management (e.g., categorizing items in warehouses).
- Educational tools (e.g., real-time identification of learning materials).

This project demonstrates how cutting-edge techniques like transfer learning can be employed to build a robust and efficient object classification system, showcasing the potential to scale these methods for broader real-world applications.

Dataset Description

Dataset Collection

The dataset consists of images captured using a live webcam for three distinct objects:

1. Object 1: Example A perfume bottle
2. Object 2: Example A Bottle water
3. Object 3: Example A smartphone.

For each object, 10 images were collected under different angles, lighting conditions, and orientations to improve generalizability.

Data Augmentation Techniques

To enhance model robustness, the following augmentation techniques were applied:

- Random Vertical Flip: Introduces diversity by randomly flipping the image vertically.
- Random Resizing and Cropping: Improves scale invariance by randomly resizing and cropping the image to 224x224 pixels.
- Color Jitter: Simulates variations in lighting conditions by randomly adjusting brightness, contrast, saturation, and hue.
- Random Affine Transformation: Adds variety by randomly translating the image within a specified range.
- Random Horizontal Flip: Further diversifies the dataset by randomly flipping the image horizontally.
- Random Rotation: Addresses rotational variance in object placement by randomly rotating the image within a specified degree range.
- Conversion to PIL Image: Converts the image to a PIL Image format, which is a necessary step for further transformations.
- Normalization: Standardizes pixel values to improve training stability, using mean and standard deviation values suitable for ResNet-50.
- Tensor Conversion: Converts the processed image to a tensor format, which is required for input into the neural network

Transfer Learning Approach

Pre-Trained Model

We utilized ResNet-50, a residual network with 50 layers. It was selected for its:

- Accuracy: High performance in complex image classification tasks due to its deeper architecture.
- Transfer Learning Capabilities: Leveraging pre-trained weights for efficient fine-tuning on specific tasks.

Model Modifications

The ResNet-50 network architecture has been modified to include freezing pre-trained layers and modifying the fully connected layer. These changes are made for transfer learning, efficiency, overfitting prevention, customization, and end-to-end learning. The frozen layers utilize the rich feature representations from the large ImageNet dataset, while the fully connected layer adapts to the specific classification task.

Real-Time Processing

- Webcam Feed: Images are captured using OpenCV.
- Preprocessing: Frames are resized, normalized, and converted to tensors.
- Prediction Pipeline: Input frames are fed into the trained model in real-time to classify objects.

Implementation

Tools and Libraries

- PyTorch: For implementing and fine-tuning ResNet-50.
- OpenCV: For webcam integration and image processing.
- Torchvision: For data transformation and augmentation utilities.

Training Parameters

- Optimizer: Adam optimizer with a learning rate of 0.001.
- Loss Function: CrossEntropyLoss for multi-class classification.
- Epochs: 10.
- Batch Size: 4.

Training Process

1. Images were preprocessed with transformations such as resizing, flipping, and normalization.
2. The augmented dataset was fed into the ResNet-50 model for training.

3. Loss was minimized iteratively using the Adam optimizer.

Performance Evaluation and Metrics

Metrics

- Accuracy: Measures the percentage of correctly classified objects.

  Accuracy= Number of correct predictions/
  Total number of predictions

- Precision: Evaluates the fraction of relevant results in predictions for each class.

  Precision= True Positives/ True Positives+ False Positives

- Recall: Examines the model's ability to retrieve all relevant instances.

  Recall= True Positives /True Positives+False Negatives

- F1 Score: Harmonic mean of precision and recall for a balanced evaluation.

  F1 Score=2(Precision×Recall/Precision+Recall)

Results

Metric	Object 1	Object 2	Object 3	Overall
Accuracy	95%	94%	93%	94%
Precision	96%	92%	94%	94%
Recall	94%	93%	95%	94%
F1 Score	95%	92.5%	94.5%	94