Project Title

Binary Image Classification Using CNNs:

Distinguishing Between Two Object Categories (Cars vs. Bikes)

Introduction

With the rise of image-based AI applications, binary classification tasks play a pivotal role in applications like security surveillance, quality control, and autonomous vehicles. This project aims to build a Convolutional Neural Network (CNN) to classify images into two categories — cars and bikes — leveraging state-of-the-art machine learning techniques.

Objective

The primary objective of this project is to design, train, validate, and test a robust CNN model for accurate binary image classification. The proposed solution will achieve the following:

- Accurately classify images as either "car" or "bike."
- Optimize performance using hyperparameter tuning.
- Evaluate generalization through rigorous testing.

Scope

The project will encompass:

1. **Dataset Preparation**:
    - Use a well-structured dataset containing images of cars and bikes, organized into train, validation, and test splits.
    - Apply preprocessing and transformations for data augmentation.
2. **Model Development**:
    - Build a CNN model using PyTorch, convolutional, pooling, and fully connected layers.
3. **Training and Validation**:
    - Train the model on the training set, with validation to monitor overfitting.
4. **Testing**:
    - Evaluate the model on unseen test data.
5. **Output**:
    - Save and visualize results with clear insights into model performance.

Technical Approach

1. **Technology Stack**:

    - Framework: PyTorch
    - Libraries: TorchVision, NumPy, Matplotlib, Tqdm, Streamlit

    - Hardware: GPU for accelerated training (if available)
2. **Model Architecture**:
    - Input: Images resized to 224x224.
    - Layers:
        - Convolutional layers with ReLU activation.
        - Max-pooling layers for dimensionality reduction.
        - Fully connected layers for classification.
    - Output: Single node
3. **Training Strategy**:
    - BCEWithLogitsLoss.
    - Adam optimizer for gradient updates.
    - Batch size: 32
    - Learning rate: 0.001
    - Epochs: 5 (adjustable based on results)

Expected Outcomes

- **Functional CNN Model**: Capable of classifying images with high accuracy.
- **Metrics**: Achieve at least 90% accuracy on the test set.
- **Scalability**: The model should be adaptable for multi-class classification tasks.

Resources Required

1. **Hardware**: A machine with GPU support for accelerated training (NVIDIA CUDA preferred).
2. **Software**:
    - Python 3.8+
    - Required Python libraries (`torch`, `torchvision`, `numpy`, `matplotlib`, `tqdm`).