Deep Learning Assignment: Convolutional Neural Networks for Image Classification

Overview

In this assignment, you will build and optimize convolutional neural networks (CNNs) for fine-grained image classification. You will work with the Stanford Dogs Dataset to classify dog breeds, a challenging task that requires sophisticated deep learning techniques.

Learning Objectives

By completing this assignment, you will:

- Implement and train convolutional neural networks from scratch
- Apply transfer learning with pre-trained models
- Design and implement data augmentation pipelines
- Visualize and interpret CNN feature maps and filters
- Optimize model performance for multi-class classification
- Address challenges with limited training data and class imbalance
- Compare different CNN architectures and training strategies

Dataset

You will work with the Stanford Dogs Dataset, which contains images of 120 breeds of dogs from around the world.

Dataset Description

- **Source**: Stanford Dogs Dataset
- **Size**: 20,580 images total
- **Classes**: 120 different dog breeds
- **Structure**: Organized in folders by breed

Assignment Tasks

Part 1: Data Preparation and Exploration (20 points)

1. Download and organize the Stanford Dogs Dataset
2. Implement a data loading pipeline using TensorFlow/PyTorch dataset utilities
3. Explore the dataset:
    - Visualize sample images from different breeds
    - Analyze the class distribution
    - Examine image size distribution
4. Preprocess the images:
    - Resize to a uniform dimension
    - Normalize pixel values
    - Create training, validation, and test splits (70%/15%/15%)
5. Implement data augmentation strategies:
    - Random crops, flips, and rotations
    - Color jittering
    - Custom augmentations appropriate for the dog classification task

Part 2: Building a CNN from Scratch (40 points)

1. Design and implement a CNN architecture with:
    - At least 4 convolutional layers
    - Appropriate pooling layers
    - Batch normalization
    - Proper activation functions
    - Global pooling before fully connected layers in case if you wanted this (not mandatory).
    - Dropout for regularization
2. Implement a training loop with:

    - Appropriate loss function for multi-class classification
    - Learning rate scheduling
    - Early stopping
    - Model checkpointing
3. Train the CNN from scratch on the dataset
4. Visualize training progress (loss and accuracy curves)
5. Evaluate the model on the validation set
6. Analyze common misclassifications and confusion between similar breeds

Part 3: Transfer Learning and Fine-tuning (40 points)

1. Implement a transfer learning approach using one of these pre-trained models:
    - ResNet50
    - EfficientNet
    - VGG16
    - MobileNetV2
2. Experiment with different transfer learning strategies:
    - Feature extraction (freezing pre-trained layers)
    - Fine-tuning (unfreezing and training with low learning rate)
    - Progressive unfreezing of layers
3. Train the model with the best transfer learning strategy
4. Compare performance with the CNN trained from scratch
5. Analyze the impact of transfer learning on convergence speed and final accuracy

Part 4: Model Interpretation and Visualization (Bouns)

1. Implement techniques to visualize and interpret your CNN:
    - Activation maps/feature maps visualization
    - Filter visualization
    - t-SNE or UMAP visualization of the feature space
2. Analyze what features the model is learning for different dog breeds

3. Identify cases where the model fails and provide explanations

Deliverables

1. Python code (Jupyter notebooks or Python scripts) with clear documentation as markdown

Evaluation Criteria

- Correctness and quality of implementation (35%)
- Model performance on test set (25%)
- Innovation and application of advanced techniques (20%)
- Quality of analysis and interpretation (15%)
- Code organization and documentation (5%)

Resources

- Stanford Dogs Dataset: http://vision.stanford.edu/aditya86/ImageNetDogs/
- PyTorch documentation: https://pytorch.org/docs/stable/index.html
- TensorFlow documentation: https://www.tensorflow.org/api_docs
- Papers on CNN architectures:
    - ResNet: https://arxiv.org/abs/1512.03385
    - EfficientNet: https://arxiv.org/abs/1905.11946
    - MobileNet: https://arxiv.org/abs/1704.04861
- Papers on visualization techniques:
    - Grad-CAM: https://arxiv.org/abs/1610.02391

Submission Instructions

Submit your code, trained model through LMS by 4/25/2025.

Notes

- You are encouraged to use GPU acceleration for training
- You can build your models in tensorflow or pytorch

- Consult with the instructor if you encounter memory or computational limitations
- You may form groups of up to 3 students for this assignment