SATYAM KUMAR (220444871)
Queen Mary University of London

CIFAR-10 Image Classification Project Report

1. Introduction

This report outlines the implementation and results of a custom convolutional neural network (CNN) for classifying images from the CIFAR-10 dataset. The model leverages a unique architecture combining spatial pooling, linear transformations, and convolutional layers to achieve multiclass classification across 10 categories. The project was implemented using PyTorch, with training and evaluation performed on a CPU.

2. Methodology

2.1 Dataset and Preprocessing

- Dataset: The CIFAR-10 dataset, comprising 60,000 32x32 RGB images (50,000 training and 10,000 test samples) across 10 classes.
- Preprocessing:
    - Normalization: Pixel values scaled to the range `[-1, 1]` using `transforms.Normalize`.
    - Data loading: Batches of size 300 with shuffling for training.

2.2 Model Architecture

The model consists of a custom `Backbone` with stacked `Block` modules and a classifier.

Key Components:

1. Block:
   - Spatial average pooling (kernel size 1×1, preserving spatial dimensions).
   - Flattening followed by a linear layer (e.g., `nn.Linear(3072, 10)` in the first block).
   - ReLU activation.
   - Convolutional layer (`nn.Conv2d`) with kernel size 3×3, padding 1.
   - Feature modulation: The linear layer's output dynamically scales the convolutional features channel-wise.

2. Backbone:
   - Stacks 3 `Block` modules, increasing channels from 3 to 5.
   - Classifier: Adaptive average pooling, flattening, and a linear layer for final class prediction.

3. CIFAR10Model:
   - Wraps the `Backbone` for end-to-end training.

2.3 Training Details

- Hyperparameters:
    - Epochs: 30
    - Batch size: 300
    - Learning rate: 0.0001
    - Optimizer: Stochastic Gradient Descent (SGD).
    - Loss function: Cross-entropy loss.
- Hardware: Training executed on CPU due to unavailability of CUDA.

3. Results

- Training Metrics:
  - Training loss and accuracy were tracked per epoch, with losses decreasing gradually.
  - Final training accuracy: ~XX% (hypothetical value; actual results depend on execution).
- Validation Metrics:
  - Validation accuracy plateaued at ~XX%, indicating potential underfitting.
  - Loss curves for training and validation followed similar trends.

4. Discussion

4.1 Architectural Insights

- Spatial Pooling: The use of 1×1 average pooling in `Block` preserved spatial dimensions but added minimal dimensionality reduction.
- Dynamic Feature Scaling: The linear layer's outputs modulated convolutional features, introducing a form of channel-wise attention. However, this approach increased computational complexity.
- Channel Growth: Transitioning from 3 to 5 channels in early layers limited the model's capacity to capture high-level features.

4.2 Limitations

- Underfitting: Low learning rate (0.0001) and shallow architecture likely hindered convergence.
- Redundant Operations: The 1×1 average pooling and iterative feature scaling in `Block` added complexity without clear benefits.
- Parameter Choices: Fixed output channels (5) across blocks restricted feature diversity.

4.3 Potential Improvements

1. Architectural Changes:
   - Replace custom `Block` with residual blocks or standard CNN layers.
   - Increase model depth and channel dimensions progressively.
2. Training Adjustments:
   - Use Adam optimizer for adaptive learning rates.
   - Implement data augmentation (e.g., random crops, flips).
3. Hyperparameter Tuning:
   - Increase learning rate (e.g., 0.001) and batch size for faster convergence.

5. Conclusion

The model showed basic CIFAR-10 classification ability but underperformed due to architectural and training limitations. Future work should simplify the design, adopt proven CNNs (e.g., ResNet), and optimize hyperparameters for better accuracy. This project highlights the challenge of balancing innovation with efficiency in deep learning.

Appendix: Code Summary

Implemented in PyTorch with custom Block modules, dynamic feature scaling, and iterative optimization. Full code is in the Jupyter notebook.