Artificial Intelligence Assignment 2 Report

Task 1: Dataset Preparation and Exploration

1. **Loading the CIFAR-10 Dataset**:
    - Successfully loaded the CIFAR-10 dataset using PyTorch's torchvision library.
    - Subset created with 2,500 images from the training set and 500 images from the test set.
2. **Visualization**:
    - Randomly sampled 5 images and displayed them in a 1x5 grid with their class labels.
3. **Normalization**:
    - Normalized the dataset to have pixel values between -1 and 1 using: transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)), as shown in the homework prompt.
    - **Importance of normalization**:
        - Improves model training stability and performance.
        - Speeds up convergence by centering inputs around zero.
        - Works well with activation functions like tanh and ReLU.

Task 2: Implementing CNN

1. **Model Architecture**:
    - Adapted the LeNet-5 model for the CIFAR-10 dataset with 32x32x3 input dimensions.
    - Architecture:
        - Two convolutional layers, each followed by average pooling.
        - Three fully connected layers, with ReLU as the activation function.
    - Output layer configured for 10 classes using softmax.
2. **Findings**:
    - The LeNet-5 architecture provided a baseline performance suitable for smaller datasets like the CIFAR-10 subset.
    - Training accuracy steadily improved, showing that the model learned the task over epochs.

Task 3: Training and Evaluation

1. **Training**:
    - Model trained for 10 epochs with:
        - Loss function: Cross-entropy.
        - Optimizer: Adam with a learning rate of 0.001.

- Batch size: 64 for both training and testing.

2. **Performance**:
   - **Train Loss and Accuracy**:
     - Initial Epoch: Loss = 2.23, Accuracy = 17.2%.
     - Final Epoch: Loss = 1.55, Accuracy = 43.5%.
   - **Validation Loss and Accuracy**:
     - Initial Epoch: Loss = 2.15, Accuracy = 22.8%.
     - Final Epoch: Loss = 1.71, Accuracy = 40.2%.
   - Observed overfitting towards later epochs, indicated by a faster decline in training loss compared to validation loss.

Task 4: Experimenting with Regularization and Activation Functions

1. **Regularization**:
   - **L1 Regularization**:
     - Added a penalty proportional to the sum of absolute weights.
     - Results:
       - Slower convergence compared to no regularization.
       - Train Accuracy: 44.6%, Val Accuracy: 34.8% (final epoch).
   - **L2 Regularization**:
     - Penalized the sum of squared weights (applied as weight_decay in the optimizer).
     - Results:
       - Train Accuracy: 42.6%, Val Accuracy: 38% (final epoch).
   - **Comparison**:
     - L1 regularization had a more noticeable impact on training performance but was less effective in validation accuracy compared to L2.
2. **Activation Functions**:
   - Tested Tanh, Leaky ReLU, and Softmax.
     - **Tanh**: Performed well but had slower convergence compared to ReLU.
     - **Leaky ReLU**: Showed slightly better generalization.
     - **Softmax**: Performed poorly when used in hidden layers due to lack of gradient flow.
   - Findings:
     - Tanh and Leaky ReLU outperformed Softmax in both training and validation accuracy.
3. **Dropout**:
   - Experimented with dropout rates of 0.1, 0.2, and 0.5.
   - Observed:
     - Lower dropout rates (0.1, 0.2) improved validation performance by reducing overfitting.

        - High dropout rate (0.5) hindered training, likely due to excessive neuron deactivation.
4. **Pooling Methods**:
    - Compared average pooling, max pooling, and min pooling.
        - **Max pooling** yielded the best validation accuracy (39.4%) by retaining prominent features.
        - **Average pooling** performed marginally worse (39.2%).
        - **Min pooling** showed a notable drop in performance due to its focus on less relevant features.

Enhanced Analysis and Findings

Performance Trends

1. **Loss and Accuracy Trends**:
    - Training loss decreased steadily across all experiments, while validation loss showed a slower decline, indicating potential overfitting in later epochs.
    - Validation accuracy generally plateaued around 40%, suggesting that the model's capacity or training data size may be limiting performance.
2. **Impact of Regularization**:
    - L1 regularization:
        - Reduced overfitting by sparsifying the weights, but it led to slower convergence and slightly lower validation accuracy.
        - Best suited for scenarios where model interpretability or feature sparsity is crucial.
    - L2 regularization:
        - Promoted smoother weight updates, resulting in better generalization than L1.
        - L2's ability to penalize large weight magnitudes helped balance bias and variance more effectively.
3. **Activation Functions**:
    - **Tanh**: Highlighted the benefits of centered outputs (range: [-1, 1]) but suffered from slower training due to gradient saturation in the extremes.
    - **Leaky ReLU**: Mitigated the "dying ReLU" problem, showing better convergence and validation accuracy. Its slope for negative inputs preserved a degree of learning for inactive neurons.
    - **Softmax**: Performed poorly when used in hidden layers, as it is not designed for this purpose. Its role is more appropriate for output layers in classification tasks.
4. **Dropout**:

    - Dropout rates of 0.1 and 0.2 balanced training stability and generalization effectively, leading to better validation performance.
    - A dropout rate of 0.5 caused excessive neuron deactivation, resulting in slower learning and diminished overall accuracy.
5. **Pooling Methods**:
    - **Max pooling**:
        - Captured the most prominent features, enhancing validation accuracy and generalization.
        - Suited for datasets with high variability in feature importance.
    - **Average pooling**:
        - Smoothed feature maps by averaging all pixel contributions, leading to a slightly lower performance than max pooling.
        - Works well when all features contribute equally to the classification task.
    - **Min pooling**:
        - Performed worst among the pooling methods as it captured the least important features, often ignoring critical patterns.

Model Limitations

1. **Dataset Size**:
    - The reduced training dataset limited the model's capacity to generalize effectively. Using the full dataset would likely improve validation performance.
2. **Architecture Simplicity**:
    - While LeNet-5 is a foundational CNN architecture, it likely lacks the complexity needed to capture nuanced patterns in diverse datasets like CIFAR-10.
    - Modern architectures such as ResNet or VGG could have significantly higher performance with the same data.
3. **Overfitting**:
    - Validation loss plateaued while training loss continued to decline, indicating that the model fit the training data too closely. Regularization techniques helped mitigate this but could not fully eliminate the issue.