Documentation: Experiment-3

By- Aditya Luthra

Roll no- 2K22/CO/30

1. Introduction

This project focuses on image classification using Convolutional Neural Networks (CNNs) to classify images from two datasets: **Cats vs. Dogs** and **CIFAR-10**. The implementation involves training custom CNN models, experimenting with different activation functions and weight initializations, and utilizing Transfer Learning with **ResNet-18** for improved performance.

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for image classification and other computer vision tasks. They utilize convolutional layers to extract spatial features from images, significantly reducing the number of parameters while preserving important patterns. CNNs are widely used in applications such as image recognition, object detection, and medical imaging.

Weight initialization is a critical factor in training deep neural networks effectively. Poor initialization can lead to issues such as slow convergence or vanishing/exploding gradients. Common weight initialization techniques include: -

- **Xavier Initialization:**Maintains uniform variance of activations across layers, suitable for sigmoid and tanh activations.
- **Kaiming Initialization:** Optimized for ReLU activations, addressing the vanishing gradient problem.
- **Random Initialization:** Assigns random values to weights, which can sometimes hinder learning efficiency.

Activation functions introduce non-linearity into the network, enabling it to learn complex patterns. Commonly used activation functions include: -

- **ReLU (Rectified Linear Unit):** Popular due to its simplicity and effectiveness in mitigating vanishing gradients.
- **Tanh:** Outputs values between -1 and 1, centering the data.
- **Leaky ReLU:** A variant of ReLU that allows small negative gradients, preventing dead neurons.

Optimizers are algorithms used to update the weights of the network based on the computed gradients. Popular optimizers include: -

- **SGD (Stochastic Gradient Descent):** Simple and efficient but may converge slowly.
- **Adam:** An adaptive optimizer that dynamically adjusts learning rates for each parameter.
- **RMSprop**: Maintains a moving average of squared gradients, similar to Adam.

ResNet-18 is a deep learning architecture known for its residual connections, which help mitigate the vanishing gradient problem in deep networks. These connections enable the training of very deep models, leading to improved classification accuracy.

---

2. Dataset Overview

2.1 Cats vs. Dogs Dataset

- The dataset consists of images of cats and dogs, categorized into two classes.

2.2 CIFAR-10 Dataset

- It consists of 60,000 32x32 color images divided into 10 classes, with 50,000 training images and 10,000 test images.

---

3. Model Architecture

A **custom CNN model** is implemented with the following structure:

Convolutional Layers: Three layers with 32, 64, and 128 filters, respectively.
Batch Normalization: Applied to the last convolutional layer.
Max Pooling Layers: Used to reduce spatial dimensions.
Fully Connected Layers: One hidden layer with 256 neurons and an output layer with 10 neurons (one for each class).
Dropout: A dropout rate of 0.5 was applied for regularization.

The following configurations were used for training: -

Batch Size: 64
Learning Rate: 0.001

Epochs: 10
Activation Functions: ReLU, Tanh, Leaky ReLU
Weight Initialization: Xavier, Kaiming, Random
Optimizers: SGD, Adam, RMSprop
Loss Function: CrossEntropy

---

4. Transfer Learning with ResNet-18

- A pretrained **ResNet-18** model is fine-tuned for classification.
- The final fully connected layer is modified to classify:
    - 2 classes (Cats vs. Dogs)
    - 10 classes (CIFAR-10)
- **Adam optimizer** is used for training.
- The trained model weights are saved for inference.

---

5. Results & Performance Evaluation

- **Loss and accuracy** are recorded for both training and validation phases.
- **Comparisons are made** between different activation functions, weight initializations, and optimizers.
- **ResNet-18 outperforms custom CNN**, showing the effectiveness of Transfer Learning.

CiFar-10

The best-performing model used leaky_Relu activation, Xavier initialization, and the Adam optimizer , achieving a test accuracy of **77.56%** .

Using Resnet - The best-performing model used sgd optimiser with test accuracy of **90.79%** .

Dog/Cat Classifier

The best-performing model used ReLU activation, Xavier initialization, and the Adam optimizer , achieving a test accuracy of **78.10%** .

Using Resnet - The best-performing model used Adam optimiser with test accuracy of **95.35%** .

---

6. Conclusion

This project demonstrates:

- The impact of different hyperparameters on CNN performance.
- How Transfer Learning can significantly improve classification accuracy.
- The importance of dataset preprocessing and augmentation techniques.

---

7. Future Enhancements

- Experiment with deeper CNN architectures.
- Implement **data augmentation** to improve generalization.
- Utilize **other pretrained models** like VGG-16, EfficientNet, etc.

References

- PyTorch Documentation
- Torchvision Datasets
- Deep Learning Papers on Transfer Learning

Technologies Used: Python, PyTorch, Torchvision, Matplotlib
Github repository link: https://github.com/Adiiii02/DL-LAB/tree/main/EXP-3
Drive Link for weights of all Model : -
https://drive.google.com/drive/folders/17odWBbwDLfek_rDTckCmLvv8SZULDGy1?usp=sharing