

Documentation: Experiment-3

By- AMAN

Roll no- 2K22/CO/48

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.

2. Dataset Overview

2.1 Cats vs. Dogs Dataset

- The dataset consists of images of cats and dogs, categorized into two classes.
- Preprocessing steps include:
 - Resizing images to 64×64 pixels.
 - Normalization using mean and standard deviation.
 - Splitting into **training (80%)** and **validation (20%)** sets.

2.2 CIFAR-10 Dataset

- Contains 10 object classes, including animals, vehicles, and objects.
 - Similar preprocessing steps as Cats vs. Dogs.
 - Used as an additional dataset for experimentation.
-

3. Model Architecture

A custom CNN model is implemented with the following structure:

- **Convolutional layers:**
 - Conv2D(3 → 32) → BatchNorm → Activation → MaxPool
 - Conv2D(32 → 64) → BatchNorm → Activation → MaxPool
- **Fully Connected Layers:**

- FC (64*16*16 → 512) → Dropout → Activation
 - FC (512 → Output Classes)
 - Supports different **activation functions** (ReLU, Tanh, Leaky ReLU) and **weight initialization methods** (Xavier, Kaiming, Random).
-

4. Training Setup

4.1 Training Parameters

- **Batch size:** 32
- **Optimizers tested:** SGD, Adam, RMSProp
- **Loss Function:** CrossEntropyLoss
- **Epochs:** 10

4.2 Training Process

1. Model is trained on Cats vs. Dogs and CIFAR-10 datasets.
 2. Training and validation losses/accuracies are recorded.
 3. The best model is selected based on validation accuracy.
-

5. 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.
-

6. Results & Performance Evaluation

- **Loss and accuracy curves** are plotted 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.

7. 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.

8. 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/Dark3302/Deep-learning-lab.git>