DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DEEP LEARNING

CO-328

DOCUMENTATION

SUBMITTED TO

Prof. Anil Parihar

SUBMITTED BY:

Anshuman Sharma(2K22/CO/72)

EXPERIMENT-3

GITHUB REPOSITORY: GITHUB

OBJECTIVE:

The objective of this lab is to implement Convolutional Neural Networks (CNNs) to classify images in the Cats vs. Dogs dataset and the CIFAR-10 dataset. You will explore different configurations by experimenting with:

- ➔ 3 Activation Functions
- ➔ 3 Weight Initialization Techniques
- ➔ 3 Optimizers

INTRODUCTION:

Image classification is a fundamental task in deep learning where a model learns to categorize images into different classes. In this lab, we experiment with Convolutional Neural Networks (CNNs) for image classification tasks such as Cats vs. Dogs and CIFAR-10. Additionally, we explore the impact of weight initialization techniques, activation functions, and optimizers, and compare custom CNNs with ResNet-18, a powerful deep learning architecture.

Convolutional Neural Network (CNN)

What are CNNs?

A Convolutional Neural Network (CNN) is a deep learning model designed for processing structured grid data, such as images. CNNs use specialized layers like convolutional layers and pooling layers to extract features automatically, making them highly effective for image-related tasks.

How do CNNs work?

CNNs operate in a hierarchical manner:

1. Convolutional Layers: Apply filters (kernels) to detect local features like edges, textures, and patterns.
2. Pooling Layers: Reduce spatial dimensions while preserving important features.
3. Fully Connected Layers: Process the extracted features for classification.
4. Activation Functions: Introduce non-linearity to improve learning.

Common Applications of CNNs

- ☑ Image classification (e.g., object detection, facial recognition).
- ☑ Medical imaging (e.g., tumor detection in X-rays).
- ☑ Self-driving cars (e.g., detecting pedestrians and road signs).
- ☑ Natural language processing (e.g., handwritten digit recognition).

Weight Initialization Techniques

Why is Weight Initialization Important

When training deep neural networks, proper weight initialization prevents problems like:

-> Vanishing gradients – When gradients shrink, preventing deep layers from learning.

-> Exploding gradients – When gradients grow too large, causing instability.

Common Weight Initialization Techniques

1. Random Initialization
    - Assigns weights randomly from a small range.
    - Works for shallow networks but can lead to unstable training in deep networks.
2. Xavier (Glorot) Initialization
    - Balances variance across layers by scaling weights based on the number of neurons.
    - Best for sigmoid & tanh activations.
3. Kaiming (He) Initialization
    - Designed for ReLU activation, scales weights to avoid vanishing gradients.
    - Best for deep CNNs.

What is the Role of Activation Functions?

Activation functions introduce **non-linearity** into neural networks, allowing them to model complex relationships. Without them, a neural network would behave like a simple linear model.

Types of Activation Functions & Use Cases

Activation Function	*Formula*	*Best Used For*	*Drawbacks*
ReLU (Rectified Linear Unit)	*max(0, x)*	*CNNs, deep networks*	*Can cause dead neurons (dying ReLU problem)*
Leaky ReLU	*max(0.01x, x)*	*Prevents dead neurons*	*Adds small negative slope*
Tanh	*(e^x - e^(-x)) / (e^x + e^(-x))*	*Recurrent Neural Networks (RNNs)*	*Can cause vanishing gradients*
Sigmoid	*1 / (1 + e^(-x))*	*Binary classification*	*Saturates at extreme values, leading to slow training*

💡 **ReLU is the most common activation function in deep CNNs due to its efficiency.**

Optimizers

What Do Optimizers Do?

Optimizers adjust a model's weights **to minimize loss** and **improve learning efficiency**. The choice of optimizer impacts **convergence speed**, **stability**, and **final accuracy**.

Common Optimizers

1. **SGD (Stochastic Gradient Descent)**
    - Updates weights using a small batch of data.
    - Works well with momentum to avoid local minima.
    - Best for **large-scale datasets & computer vision tasks**.
2. **Adam (Adaptive Moment Estimation)**
    - Combines **SGD + RMSprop**, adapting learning rates dynamically.
    - Works well for most deep learning tasks.
    - Best for **non-stationary problems like NLP & time-series data**.
3. **RMSprop**
    - Divides learning rate by moving average of squared gradients.
    - Best for **RNNs and reinforcement learning**.

💡 **Adam is widely used for deep learning models due to its fast convergence.**

ResNet (Residual Neural Network)

What is ResNet?
ResNet (Residual Neural Network) is a deep learning architecture introduced in 2015 that solves the vanishing gradient problem by using skip connections (residual connections).
ResNet Architecture

- Skip connections allow gradients to flow directly through layers, preventing vanishing gradients.
- Deep networks (e.g., ResNet-50, ResNet-101) outperform traditional CNNs in image classification.
- Batch Normalization is used to stabilize training.

Why is ResNet Important?
✅ Enables training of very deep networks (e.g., 152 layers).
✅ Achieves higher accuracy than traditional CNNs.
✅ Used in ImageNet classification, object detection, and medical imaging.

CIPHER-10 Classification

What is CIFAR-10?
CIFAR-10 is a widely used dataset for image classification, consisting of 60,000 images in 10 categories: 🚗 Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck.

Challenges in CIFAR-10

- Small images (32×32 pixels) require deeper networks.
- High inter-class similarity makes classification harder.

Approach
We train CNNs with different activation functions, weight initializations, and optimizers, then compare them with ResNet-18.

Activation	*Optimizer*	*Weight Init*	*Test Accuracy*
ReLU	*SGD*	*Xavier*	*0.7193*
ReLU	*SGD*	*Kaiming*	*0.7129*
ReLU	*SGD*	*Random*	*0.6867*
ReLU	*Adam*	*Xavier*	*0.7179*
ReLU	*Adam*	*Kaiming*	*0.7138*
ReLU	*Adam*	*Random*	*0.7240*
ReLU	*RMSprop*	*Xavier*	*0.7119*
ReLU	*RMSprop*	*Kaiming*	*0.6541*
ReLU	*RMSprop*	*Random*	*0.7293*
Tanh	*SGD*	*Xavier*	*0.6905*
Tanh	*SGD*	*Kaiming*	*0.6921*
Tanh	*SGD*	*Random*	*0.6639*
Tanh	*Adam*	*Xavier*	*0.6997*
Tanh	*Adam*	*Kaiming*	*0.6945*
Tanh	*Adam*	*Random*	*0.6940*
Tanh	*RMSprop*	*Xavier*	*0.6566*
Tanh	*RMSprop*	*Kaiming*	*0.6574*
Tanh	*RMSprop*	*Random*	*0.6545*
Leaky ReLU	*SGD*	*Xavier*	*0.7189*
Leaky ReLU	*SGD*	*Kaiming*	*0.7112*
Leaky ReLU	*SGD*	*Random*	*0.6741*
Leaky ReLU	*Adam*	*Xavier*	*0.7284*
Leaky ReLU	*Adam*	*Kaiming*	*0.7132*
Leaky ReLU	*Adam*	*Random*	*0.7255*
Leaky ReLU	*RMSprop*	*Xavier*	*0.7259*
Leaky ReLU	*RMSprop*	*Kaiming*	*0.6557*
Leaky ReLU	*RMSprop*	*Random*	*0.7137*

Best CNN Model

- Best test accuracy: 0.7293
- Best combination: ReLU + RMSprop + Random Initialization

ResNet-18 outperformed the best CNN model, achieving 80.24% accuracy vs. 72.93%.

◈ The skip connections in ResNet-18 allow deeper learning, leading to better generalization

Conclusion

✅ CNNs can classify images effectively, but ResNet-18 outperforms standard CNNs in deep learning tasks.

✅ Weight initialization, activation functions, and optimizers significantly impact model performance.

✅ ResNet-18's residual connections help train deeper networks efficiently.

Cat vs Dog Classification

The Cats vs. Dogs dataset is a binary classification problem where a model predicts whether an image contains a cat or a dog.

Challenges:

✔ High intra-class variation (dogs and cats of different breeds look different).

✔ Similar inter-class features (both animals have fur, ears, eyes).

✔ Data imbalance (some classes may have more images).

Approach:

🔷 Train a custom CNN model with different activations, weight initializations, and optimizers.

🔷 Fine-tune ResNet-18 on the same dataset.

🔷 Compare CNN vs. ResNet-18 in terms of accuracy.

Weight Initialization Techniques

Weight initialization prevents vanishing/exploding gradients and improves convergence.

1. Xavier Initialization – Best for Sigmoid/Tanh activations.
2. Kaiming Initialization – Optimized for ReLU.
3. Random Initialization – Basic approach, often unstable.

Activation Functions

Activation functions introduce non-linearity, enabling CNNs to learn complex patterns.

- ReLU (best for CNNs)
- Leaky ReLU (solves the dead neuron issue)
- Tanh (good for feature scaling but can cause vanishing gradients)

Optimizers

Optimizers adjust weights to minimize loss efficiently.

- SGD – Simple, stable, good for large datasets.
- Adam – Adaptive learning rates, fast convergence.
- RMSprop – Suitable for tasks with changing gradients.

Activation	*Optimizer*	*Weight Init*	*Test Accuracy*
ReLU	*SGD*	*Xavier*	*0.7964*
ReLU	*SGD*	*Kaiming*	*0.7750*
ReLU	*SGD*	*Random*	*0.7670*
ReLU	*Adam*	*Xavier*	*0.7714*
ReLU	*Adam*	*Kaiming*	*0.4916*
ReLU	*Adam*	*Random*	*0.7838*
ReLU	*RMSprop*	*Xavier*	*0.5084*
ReLU	*RMSprop*	*Kaiming*	*0.4916*
ReLU	*RMSprop*	*Random*	*0.4916*
Tanh	*SGD*	*Xavier*	*0.7666*
Tanh	*SGD*	*Kaiming*	*0.7196*
Tanh	*SGD*	*Random*	*0.6530*
Tanh	*Adam*	*Xavier*	*0.4916*
Tanh	*Adam*	*Kaiming*	*0.7800*
Tanh	*Adam*	*Random*	*0.7650*
Tanh	*RMSprop*	*Xavier*	*0.6200*
Tanh	*RMSprop*	*Kaiming*	*0.4916*
Tanh	*RMSprop*	*Random*	*0.4916*
Leaky ReLU	*SGD*	*Xavier*	*0.7890*
Leaky ReLU	*SGD*	*Kaiming*	*0.7600*
Leaky ReLU	*SGD*	*Random*	*0.7010*
Leaky ReLU	*Adam*	*Xavier*	*0.8010*
Leaky ReLU	*Adam*	*Kaiming*	*0.7850*
Leaky ReLU	*Adam*	*Random*	*0.7680*
Leaky ReLU	*RMSprop*	*Xavier*	*0.6150*
Leaky ReLU	*RMSprop*	*Kaiming*	*0.4916*
Leaky ReLU	*RMSprop*	*Random*	*0.4916*

Conclusion

◈ ResNet-18 outperformed the best CNN model for Cats vs. Dogs classification, achieving 82.00% accuracy vs. 80.10%.

◈ Residual connections in ResNet-18 help train deeper models effectively.

Final Verdict

➔ CNNs are powerful for image classification but ResNet-18 consistently achieves better performance due to its superior architecture.

Key Takeaways

✅ Activation, Optimizer, and Weight Initialization impact CNN performance.
✅ SGD is stable but slower, Adam converges faster.
✅ ResNet-18 consistently outperforms CNNs in complex classification tasks.
🔷 Future Work:

- Train deeper models like ResNet-50 for better accuracy.
- Use data augmentation to improve generalization.
- Experiment with learning rate scheduling for optimal convergence.

GITHUB REPOSITORY LINK: [GITHUB]