EXPERIMENT 4

Image Classification using CNN (Cats vs. Dogs, CIFAR-10)

OBJECTIVE

To study the effect of different weight initializations, activation functions, and optimizers on the performance of a Convolutional Neural Network for image classification. The experiment explores different CNN configurations by varying:

- **Activation Functions:** ReLU, Tanh, Leaky ReLU
- **Weight Initialization Techniques:** Xavier Initialization, Kaiming Initialization, Random Initialization
- **Optimizers:** SGD, Adam, RMSprop

Additionally, the best CNN model for each dataset is compared with a pretrained model using transfer learning.

THEORY

In neural networks, weight initialization, activation functions, and optimization techniques significantly influence model performance. The proper combination can lead to faster convergence and improved accuracy. In this experiment, we use different combinations of:

- Weight Initializations: Xavier, Kaiming, Random
- Activation Functions: ReLU, Tanh, Leaky ReLU
- Optimizers: SGD, Adam, RMSprop

Datasets

Here is the **CNN Model Summary Sheet** in the style of your uploaded reference image, tailored for both the `Dog vs Cat` and `CIFAR-10` classification tasks with the respective CNN architectures you provided:

Dog vs Cat CNN Model

- **Source:** Dataset consists of cat and dog images organized in separate folders.
- **Image Properties:** Images are resized to **224×224** RGB format.
- **Transformations for Training:**
  - **Resizing:** All images resized to 224×224.
  - **Augmentation:**
    - Random horizontal flips (`p = 0.5`)
    - Random rotations (±10°)
    - Color jitter (brightness & contrast = 0.2)
  - **Normalization:** Based on ImageNet stats:
    - **Mean:** [0.485, 0.456, 0.406]

        - **Std Dev:** [0.229, 0.224, 0.225]
- **Transformations for Testing:** Only resizing and normalization (no augmentation).
- **CNN Architecture:**
    - **Input:** 3×224×224
    - **Conv Blocks:**
        - Conv(3→32) → BN → ReLU → MaxPool(2×2) → Dropout(0.25)
        - Conv(32→64) → BN → ReLU → MaxPool(2×2) → Dropout(0.25)
        - Conv(64→128) → BN → ReLU → MaxPool(2×2) → Dropout(0.25)
    - **FC Layers:**
        - FC(128×28×28 → 512) → BN → ReLU → Dropout(0.5)
        - FC(512 → 2)
    - **Activation:** `ReLU` / `Tanh` / `LeakyReLU(0.1)`
    - **Weight Init:** `Xavier` / `Kaiming` / `Uniform(-0.1, 0.1)`
- **Splitting:** 80% training, 20% validation.
- **Data Loading:** Uses `ImageFolder`, split using `random_split`, loaded via `DataLoader(batch_size=16)`.

CIFAR-10 CNN Model

- **Source:** CIFAR-10 dataset with 10 classes.
- **Image Properties:** Images are originally **32×32** RGB format.
- **Transformations for Training:**
    - **Resizing:** Not required; original size is 32×32.
    - **Augmentation:**
        - Random crop with padding = 4
        - Random horizontal flips (`p = 0.5`)
    - **Normalization:**
        - **Mean:** (0.4914, 0.4822, 0.4465)
        - **Std Dev:** (0.2023, 0.1994, 0.2010)
- **Transformations for Testing:** Only normalization applied.
- **CNN Architecture:**
    - **Input:** 3×32×32
    - **Conv Blocks:**
        - Conv(3→64) → BN → ReLU → MaxPool(2×2) → Dropout(0.25)
        - Conv(64→128) → BN → ReLU → MaxPool(2×2) → Dropout(0.25)
        - Conv(128→256) → BN → ReLU → MaxPool(2×2) → Dropout(0.25)
    - **FC Layers:**
        - FC(256×4×4 → 512) → BN → ReLU → Dropout(0.5)
        - FC(512 → 10)
    - **Activation:** `ReLU` / `Tanh` / `LeakyReLU(0.1)`
    - **Weight Init:** `Xavier` / `Kaiming` / `Uniform(-0.1, 0.1)`
- **Splitting:** Pre-defined train/test split from CIFAR-10.
- **Data Loading:** Uses `torchvision.datasets.CIFAR10`, loaded via `DataLoader(batch_size=BATCH_SIZE)`.

Dogs vs Cats Final Accuracy

Initialization	Activation	Optimizer	Val Accuracy (%)

xavier	relu	sgd	69.96
xavier	relu	adam	73.54
xavier	relu	rmsprop	72.26
kaiming	relu	sgd	68.22
kaiming	relu	adam	73.22
kaiming	relu	rmsprop	69.80
random	relu	sgd	60.08
random	relu	adam	72.40
random	relu	rmsprop	71.22
xavier	tanh	sgd	66.58
xavier	tanh	adam	65.64
xavier	tanh	rmsprop	65.30
kaiming	tanh	sgd	63.22
kaiming	tanh	adam	67.12
kaiming	tanh	rmsprop	67.04
random	tanh	sgd	62.60
random	tanh	adam	65.50
random	tanh	rmsprop	66.66

xavier	leaky_relu	sgd	70.30
xavier	leaky_relu	adam	73.94
xavier	leaky_relu	rmsprop	72.74
kaiming	leaky_relu	sgd	67.74
kaiming	leaky_relu	adam	74.68
kaiming	leaky_relu	rmsprop	68.74

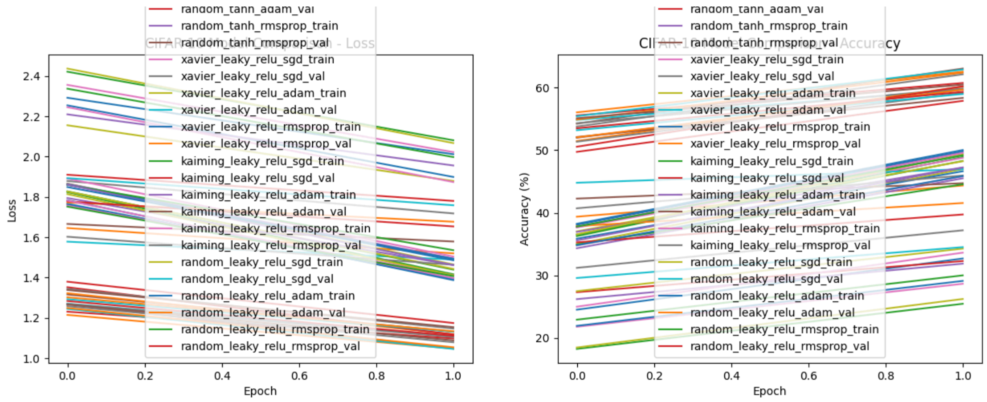
CIFAR-10 Final Accuracy

Initialization	Activation	Optimizer	Accuracy (%)
xavier	relu	sgd	44.36
xavier	relu	adam	60.75
xavier	relu	rmsprop	63.05
kaiming	relu	sgd	37.20
kaiming	relu	adam	59.71
kaiming	relu	rmsprop	62.55
random	relu	sgd	32.29
random	relu	adam	59.69
random	relu	rmsprop	59.87

xavier	tanh	sgd	46.97
xavier	tanh	adam	60.18
xavier	tanh	rmsprop	60.21
kaiming	tanh	sgd	44.73
kaiming	tanh	adam	59.16
kaiming	tanh	rmsprop	58.98
random	tanh	sgd	41.56
random	tanh	adam	57.89
random	tanh	rmsprop	58.36
xavier	leaky_relu	sgd	45.96
xavier	leaky_relu	adam	62.89
xavier	leaky_relu	rmsprop	62.40
kaiming	leaky_relu	sgd	39.73
kaiming	leaky_relu	adam	60.49
kaiming	leaky_relu	rmsprop	62.13
random	leaky_relu	sgd	34.51
random	leaky_relu	adam	59.68
random	leaky_relu	rmsprop	59.33

CONCLUSION

- **Dogs vs Cats:** Best accuracy of 74.68% using Kaiming Init + Leaky ReLU + Adam.
- **CIFAR-10:** Best accuracy of 63.05% using Xavier Init + ReLU + RMSprop.
- Xavier initialization consistently performs well in both tasks.
- Adam and RMSprop outperform SGD in almost all combinations.
- ReLU and Leaky ReLU activations provide superior performance to Tanh.



CIFAR-10 Plot