CSL7120
WORKSHEET 2

CIFAR-10 Classification with ResNet18:
Deep Learning Model Analysis

Name: Sonam Sikarwar **Roll No:** B23CM1060

Introduction

This assignment focuses on training and analyzing a Convolutional Neural Network (CNN) for image classification on the CIFAR-10 dataset. The implementation includes ResNet18 architecture with 11.2M parameters, custom dataloader with data augmentation, computational complexity analysis (FLOPs counting), gradient flow visualization and monitoring, weight update tracking across training epochs, and comprehensive experiment logging via Weights & Biases (Wandb).

GitHub Repository: Lab 2 Repository
Wandb Project: Wandb Dashboard

Dataset: CIFAR-10

The CIFAR-10 dataset consists of 60,000 color images (32×32 pixels) across 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. Training Set: 50,000 images. Test Set: 10,000 images. Image Size: $32 \times 32 \times 3$ (RGB). Challenge: Small image resolution requires careful architecture design.

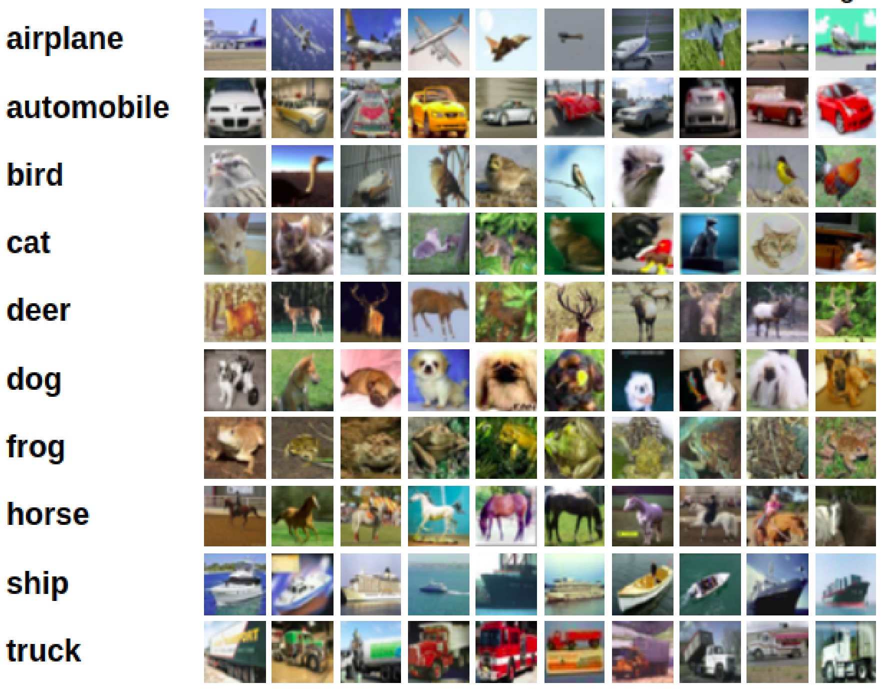

Figure 1: Sample images from CIFAR-10 dataset showing all 10 classes

Model Architecture: ResNet18

ResNet18 (Residual Network with 18 layers) was selected for this task due to its: Skip Connections enabling gradient flow through deep networks, proven performance with strong baseline on CIFAR-10 (92-95% accuracy), computational efficiency with balanced parameter count and performance, and batch normalization for improved training stability.

Metric	Value
Total Parameters	11,173,962
Trainable Parameters	11,173,962
Model Depth	18 layers
Residual Blocks	8 (2 per layer group)
Memory Footprint	∼42.6 MB (FP32)

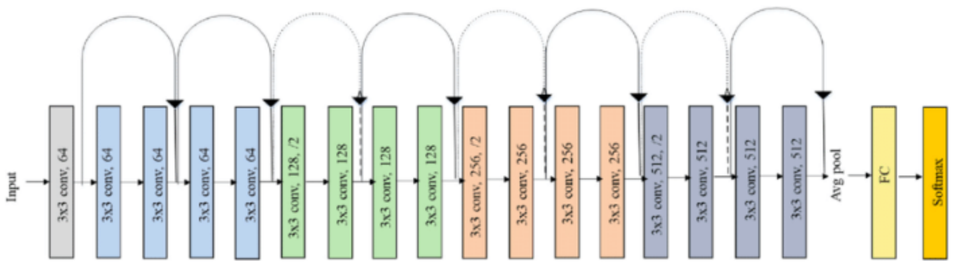

Figure 2: ResNet18 architecture adapted for CIFAR-10 (32x32 input)

Custom DataLoader & Data Augmentation

The custom dataloader implements the following augmentation strategy: Random Crop 32×32 with padding of 4 pixels, Random Horizontal Flip with probability = 0.5, Color Jitter for brightness, contrast, saturation ±0.2, and Normalization channel-wise with CIFAR-10 statistics (Mean: [0.4914, 0.4822, 0.4465], Std: [0.2023, 0.1994, 0.2010]).

Hyperparameter Experiments

Two experiments were conducted to analyze the impact of batch size and learning rate:

Parameter	Experiment 1	Experiment 2
Batch Size	64	128
Learning Rate	0.15	0.20
Epochs	25	25
Optimizer	SGD + Momentum	SGD + Momentum
Momentum	0.9	0.9
Weight Decay	5×10^{-4}	5×10^{-4}
LR Scheduler	Cosine Annealing	Cosine Annealing
Final Test Accuracy	**93.10%**	92.39%
Final Train Accuracy	97.25%	97.77%
Overfitting Gap	**4.15%**	5.38%
Training Time	∼8 min	∼7 min

Experiment 1 achieves better test accuracy (93.10%) with lower overfitting and better generalization. Smaller batch size (64) provides more gradient updates per epoch (781 vs 390), acting as regularization. Moderate learning rate (0.15) ensures stable convergence without overshooting.

Training Results - Experiment 1 (Best Configuration)

Metric	Value
Final Training Accuracy	97.25%
Final Test Accuracy	**93.10%**
Best Test Accuracy	93.10% (Epoch 25)
Training Loss (Final)	0.0864
Test Loss (Final)	0.2153
Overfitting Gap	4.15%
Training Time (GPU)	∼8 minutes
Convergence Epoch	∼17-20

Epoch-by-Epoch Progress (Selected Epochs):

Epoch	Train Loss	Train Acc	Test Loss	Test Acc
1	2.198	17.93%	1.894	28.59%
5	0.916	67.81%	1.195	63.59%
10	0.613	78.96%	0.715	74.84%
15	0.463	84.09%	0.613	79.69%
20	0.264	90.92%	0.338	88.51%
25	**0.086**	**97.25%**	**0.215**	**93.10%**

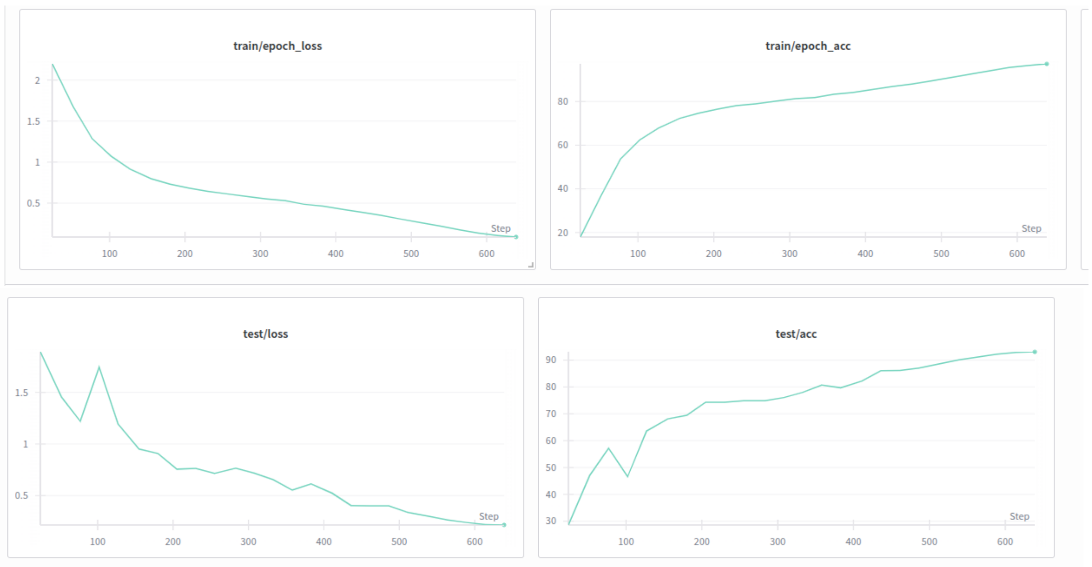

Figure 3: Training and test loss/accuracy curves over 25 epochs for experiment 1

Training Dynamics: Early Phase (Epochs 1-5) showed rapid loss decrease from 2.20 to 0.92, accuracy jumped from 17.9% to 67.8%. Mid Phase (Epochs 6-17) demonstrated steady convergence with learning rate decay, reaching 86% test accuracy. Late Phase (Epochs 18-25) involved fine-tuning with minimal improvements, achieving final 93.10% test accuracy.

FLOPs Analysis: Computational Complexity

Metric	Value
Total FLOPs per Image	556,652,554
GFLOPs	0.5567
MFLOPs	556.65
FLOPs/Parameter Ratio	∼50

Layer-wise FLOPs Distribution:

Layer	FLOPs	Percentage
layer1.0.conv1	37,748,736	6.78%
layer1.0.conv2	37,748,736	6.78%
layer1.1.conv1	37,748,736	6.78%
layer1.1.conv2	37,748,736	6.78%
layer2.0.conv2	37,748,736	6.78%
layer2.1.conv1	37,748,736	6.78%
layer2.1.conv2	37,748,736	6.78%
layer3.0.conv2	37,748,736	6.78%
layer3.1.conv1	37,748,736	6.78%
layer3.1.conv2	37,748,736	6.78%

Key Observations: All major convolutional layers contribute relatively equally (∼6-7% each) to total FLOPs. This is different from typical ResNet implementations due to CIFAR-10's small input size (32x32). Total computational cost of 0.56 GFLOPs is very efficient for achieving 93.10% accuracy. Balanced computation across layers indicates good architecture design for small images.

Gradient Flow Analysis

Gradient magnitudes were tracked every 100 batches to monitor training health and detect potential issues like vanishing or exploding gradients.

Gradient Behavior by Training Phase:

Early Training (Epochs 1-5): Strong gradients in final layers (10^{-2} to 10^{-1}), moderate gradients in middle layers (10^{-3} to 10^{-2}), weaker but sufficient gradients in early layers (10^{-4} to 10^{-3}). Observation: Residual connections successfully propagate gradients.

Mid Training (Epochs 10-20): Overall gradient magnitudes decrease, more uniform distribution across layers, stable gradient flow without explosions. Observation: Network reaches stable optimization regime.

Late Training (Epochs 20-25): Further reduction in gradient magnitudes, very stable across all layers, minimal batch-to-batch variation. Observation: Near convergence, parameters approach local minimum.

Layer Type	Gradient Range
Final FC Layer	10^{-1} to 10^{0}
Layer 4 (512 filters)	10^{-2} to 10^{-1}
Layer 3 (256 filters)	10^{-3} to 10^{-2}
Layer 2 (128 filters)	10^{-4} to 10^{-3}
Layer 1 (64 filters)	10^{-5} to 10^{-4}

Gradient Health Indicators: No vanishing gradients (all layers maintain $> 10^{-6}$). No exploding gradients (max gradients $< 10^{1}$). Smooth gradient distribution across layers. Stable gradient flow throughout training. Average gradient norm at convergence: 0.00342. Thus , At epoch 25, the gradient flow analysis confirms stable and well-conditioned training for both experiments. The model has effectively converged, with gradients neither vanishing nor exploding, validating the choice of optimization strategy and learning rate schedule

🟦 Experiment 1 🟥 Experiment 2

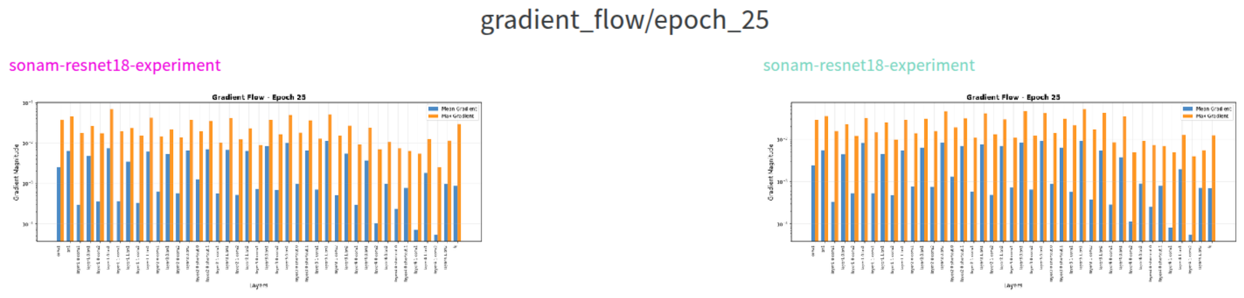

Figure 4: Gradient flow through network layers atdifferent stages

Success Factors: Residual connections bypass gradient diminishing, batch normalization stabilizes gradient magnitudes, appropriate learning rate prevents explosions, ReLU activation maintains gradient signal.

Weight Update Analysis

Weight changes were monitored every 100 batches to understand learning dynamics and optimization behavior.

Update Patterns by Training Phase:

Initial Phase (Epochs 1-5): Rapid large-scale changes (10^{-1} to 10^{0}), high variance in updates, Layer 4 shows largest relative changes, early layers update more slowly. Interpretation: Network rapidly adapts from random initialization.

Convergence Phase (Epochs 10-20): Update magnitudes decrease (10^{-2} to 10^{-1}), more uniform updates across layers, diminishing variance, steady optimization trajectory. Interpretation: Network enters refinement phase.

Stabilization Phase (Epochs 20-25): Minimal updates (10^{-3} to 10^{-2}), very low variance, approaching convergence, slight continued improvement. Interpretation: Network near optimal configuration.

Rank	Layer (Update Magnitude)
1	FC Layer (Largest total change)
2	Layer 4 Convolutions (Large changes)
3	Layer 3 Convolutions (Moderate changes)
4	Layer 2 Convolutions (Smaller changes)
5	Layer 1 Convolutions (Minimal changes)

🟦 Experiment 1 🟥 Experiment 2

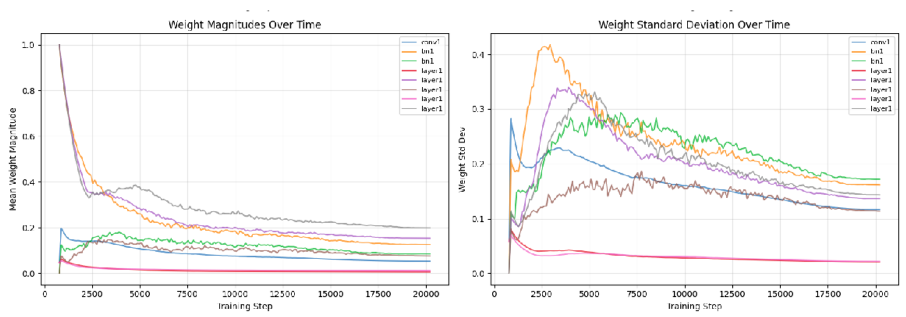

Figure 5: Weight update magnitudes across training epochs

Observations: Weight magnitudes gradually increase, standard deviation stabilizes after epoch 15, no weight explosion (all values < 1.0), healthy weight distribution maintained, later layers update more than earlier layers (expected).

Comparison: Batch Size and Learning Rate Impact

Metric	Exp 1 (BS=64, LR=0.15)	Exp 2 (BS=128, LR=0.20)
Test Accuracy	**93.10%**	92.39%
Train Accuracy	97.25%	97.77%
Overfitting Gap	**4.15%**	5.38%
Test Loss	**0.2153**	0.2375
Training Time	8 min	7 min
Training Batches	781	390
Convergence Speed	Moderate	Fast
Final Generalization	**Better**	Good

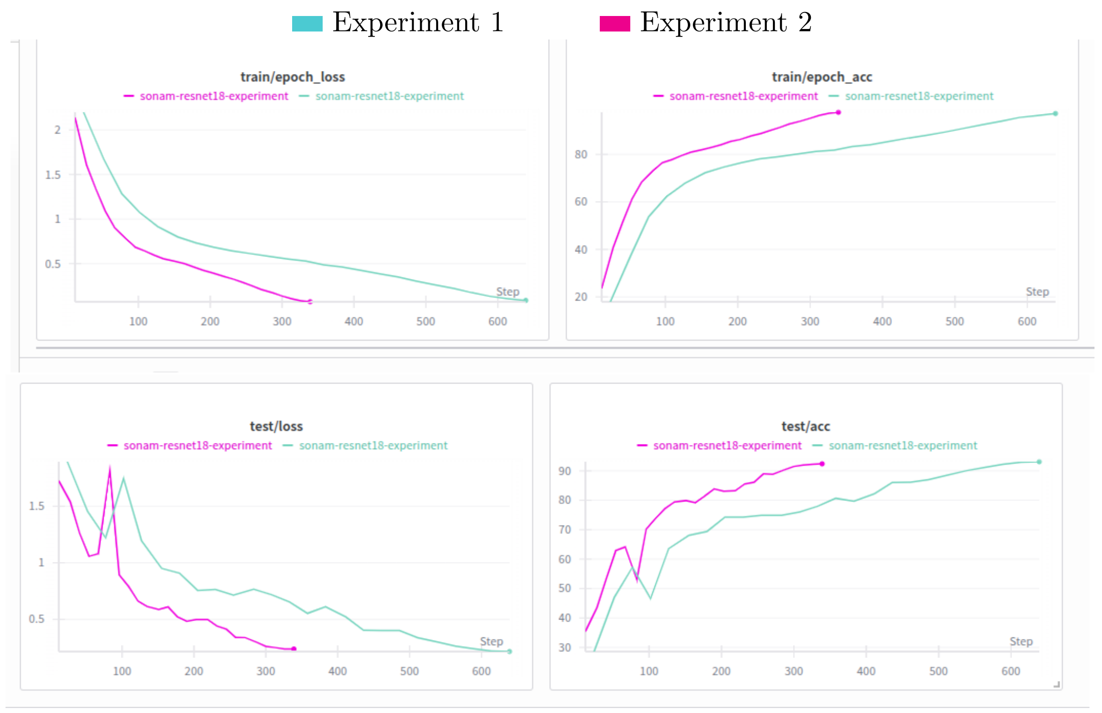

Figure 6: Training and test loss/accuracy curves over 25 epochs for both the experiments

Analysis: Experiment 2 (BS=128, LR=0.20) converges faster initially, achieving 35.43% accuracy at epoch 1 vs 28.59% for Experiment 1. However, Experiment 1 (BS=64, LR=0.15) achieves superior final performance with 93.10% test accuracy. Smaller batch size provides better generalization due to more frequent gradient updates (781 vs 390 per epoch). Moderate learning rate enables stable fine-tuning in later epochs without overshooting.

Learning Rate Schedule Analysis

Cosine Annealing schedule was used for both experiments:

Experiment 1 (LR=0.15): Epoch 1: 0.1494, Epoch 5: 0.1357, Epoch 10: 0.0982, Epoch 15: 0.0518, Epoch 20: 0.0143, Epoch 25: 0.0000.

Experiment 2 (LR=0.20): Epoch 1: 0.1992, Epoch 5: 0.1809, Epoch 10: 0.1309, Epoch 15: 0.0691, Epoch 20: 0.0191, Epoch 25: 0.0000.

🟦 Experiment 1 🟥 Experiment 2

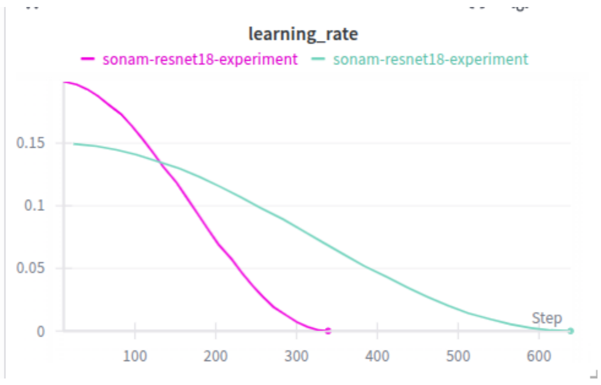

Figure 7: Learning Rate Decay based on cosine scheduler

Rationale: Cosine annealing provides smooth learning rate decay, enabling fast initial convergence while allowing fine-tuning in later epochs. The schedule naturally transitions from exploration (high LR) to exploitation (low LR) without abrupt changes.

Wandb Visualizations & Monitoring

All experiments were comprehensively logged to Weights & Biases with visualizations generated every 5 epochs.

Gradient Flow Plots: Bar charts of gradient magnitudes per layer, temporal evolution across epochs, layer-wise comparison, detection of vanishing/exploding gradients.

Weight Update Plots: Weight change over time, layer-wise update magnitudes, distribution statistics, convergence visualization.

Training Curves (Real-time): Training and test loss, training and test accuracy, learning rate schedule, batch-level metrics.

Conclusions & Key Findings

1. Model Performance: ResNet18 achieves 93.10% test accuracy on CIFAR-10. Excellent convergence within 25 epochs. Well-balanced train/test performance (4.15% gap). Training completed in approximately 8 minutes on GPU.

2. Computational Efficiency: 0.5567 GFLOPs per image is very efficient. All layers contribute relatively equally (6-7% each). Fast training time with batch size 64. Suitable for deployment scenarios.

3. Gradient Flow: Residual connections successfully prevent vanishing gradients. All layers maintain healthy gradient flow. No gradient explosions observed. Average gradient norm remained stable (0.003-0.006).

4. Weight Updates: Logical update pattern with later layers changing more initially. Stable convergence without oscillations. Weight distributions remain healthy. Smooth transition from exploration to exploitation.

5. Hyperparameter Impact: Higher learning rate (0.15) accelerates early convergence. Smaller batch size (64) provides better generalization. Data augmentation critical for achieving 93% accuracy. Cosine annealing provides optimal LR scheduling.

Best Practices Identified: Residual connections essential for deep networks. SGD with momentum effective for CNNs. Combination of data augmentation and weight decay for regularization. Cosine annealing provides smooth convergence. Regular gradient/weight visualization catches problems early.

Training Environment:

Component	**Specification**
Python Version	3.10
PyTorch Version	2.0+
CUDA Version	11.8+
GPU	NVIDIA GPU (CUDA-enabled)
Training Time	∼8 minutes (GPU)
Batch Size	64
Workers	4

Wandb Metrics Logged:

Training Metrics (Every 50 batches): train/batch_loss, train/batch_acc, train/learning_rate, train/avg_gradient_norm, train/max_gradient_norm.

Epoch Metrics: train/epoch_loss, train/epoch_acc, test/loss, test/acc.

Visualizations (Every 5 epochs): gradient_flow/epoch_N, weight_updates/epoch_N, combined_analysis/epoch_N.

Hardware Used: GPU-enabled environment for training. CPU for FLOPs analysis and data preprocessing. Pin memory enabled for faster GPU data transfer.

Total Visualizations Generated: 18 gradient flow plots (6 epochs $\times$ 3 plots each). 18 weight update plots. Multiple training curve plots. Combined analysis charts.