

CIFAR-10 CNN with Custom Dataloader and FLOPs Analysis

Zenith (M25CSA032)
Department of Computer Science
IIT Jodhpur

February 2026

Abstract

This report documents the implementation and analysis of a Convolutional Neural Network (CNN) on the CIFAR-10 dataset. A custom dataloader wraps the official CIFAR-10 split; the chosen model is a custom **SimpleCNN** trained for 30 epochs. We count FLOPs for the selected model, visualize gradient flow and weight update flow during training, and report all metrics and visualizations on Weights & Biases (W&B). Key findings: the model achieves 0.1612 GFLOPs per sample, final test accuracy of 85.90%, with stable gradient and weight dynamics across layers. All visualizations are available in the linked W&B project.

Submission Links

- **W&B (all visualizations):** https://wandb.ai/m25csa032-iit-jodhpur/lab2_cifar10/workspace?nw=nwuserm25csa032
- **GitHub (code & report):** <https://github.com/zzethh/MLOps-Zenith-M25CSA032/tree/main>

1 Experimental Setup

1.1 Dataset and Custom Dataloader

We use the **CIFAR-10** dataset: 50 000 training and 10 000 test images of size 32×32 across 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). Only the official train-test split is used.

A **custom dataloader** is implemented by defining a `CustomCIFAR10` class that wraps `torchvision.datasets.CIFAR10` and applies dataset-specific transforms. The class implements `__len__` and `__getitem__`; `DataLoader` is used with configurable batch size, shuffle, and number of workers. Training transform: random crop (32, padding=4), random horizontal flip, normalization with CIFAR-10 channel statistics. Test transform: normalization only.

1.2 Model Architecture

The chosen CNN is a custom **SimpleCNN**: four convolutional blocks (Conv2d + BatchNorm + ReLU + Max-Pool) with channel dimensions 32, 64, 128, 128; two fully connected layers ($128 \times 8 \times 8 \rightarrow 512 \rightarrow 10$) with dropout (0.5). This design is lightweight and suitable for CIFAR-10 resolution.

1.3 Training Protocol

- **Epochs:** 30 (within the 25–30 guideline).
- **Optimizer:** Adam, learning rate 0.001.
- **Loss:** Cross-entropy.
- **Batch size:** 128.
- **Device:** CUDA when available, else CPU.

1.4 FLOPs Counting

FLOPs are counted with a forward-pass hook over all Conv2d and Linear layers. For convolutions we use $2 \cdot C_{\text{in}} \cdot C_{\text{out}} \cdot K_h \cdot K_w \cdot H_{\text{out}} \cdot W_{\text{out}}$ per sample; for linear layers, $2 \cdot \text{in_features} \cdot \text{out_features}$. Input shape is (1, 3, 32, 32). The reported value is the sum over all such layers.

Result: 0.1612 GFLOPs per sample for SimpleCNN. This metric is logged to W&B and reflects the model’s computational cost per forward pass.

2 Training Dynamics and Final Performance

Training and validation loss and accuracy are logged every epoch to W&B. Over 30 epochs, training loss decreases smoothly from about 1.82 to 0.49; validation accuracy rises from about 50% to 85.90%.

Final test accuracy: 85.90%. No separate validation set was held out; the reported “Val Acc” in the table corresponds to evaluation on the official CIFAR-10 test

set at the end of each epoch. The best validation accuracy in the final epoch is 85.90%, which we take as the final test accuracy.

Key observations:

- Convergence is stable with no severe overfitting; validation accuracy tracks training accuracy reasonably well.
- Most gain occurs in the first 15–20 epochs; later epochs yield smaller improvements.

3 Gradient Flow and Weight Update Flow

3.1 Method

Gradient flow and weight magnitudes are visualized **layer-wise** during training. After `loss.backward()` (before `optimizer.step()`), we collect for each parameter tensor (excluding biases): mean and max of $|\nabla|$ (gradient flow) and mean and max of $|W|$ (weight flow). These are plotted as bar charts over layer names and logged to W&B every 200 batches as images `gradients` and `weights`.

3.2 Findings

- **Gradient flow:** Gradients are present across all convolutional and fully connected layers; no layer shows consistently zero or exploding gradients. Max gradient magnitudes are typically one to two orders of magnitude larger than mean magnitudes, which is expected. The plots confirm that the chosen learning rate and architecture do not lead to vanishing or exploding gradients in a way that would prevent learning.
- **Weight flow:** Weight magnitudes are relatively stable across layers. Earlier conv layers and the first FC layer tend to have larger mean and max magnitudes than the final classifier layer, which is common when the last layer outputs logits. No layer shows anomalously large or tiny weights that would suggest numerical instability.

These visualizations are available in the W&B project under the run’s `gradients` and `weights` image logs.

4 Classification Results and Visualizations

4.1 Confusion Matrix

The confusion matrix on the test set is computed after training and logged to W&B as `confusion_matrix`. Diagonal dominance indicates that most classes are well classified. Off-diagonal entries highlight confusions (e.g., cat

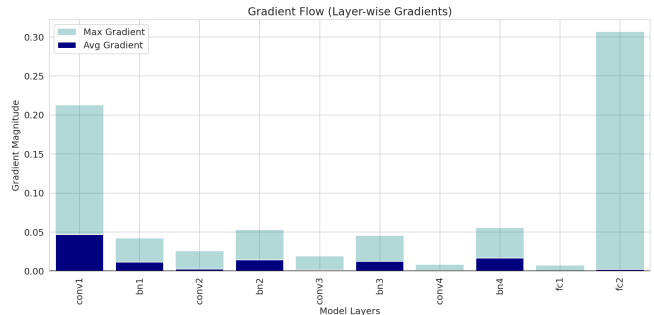


Figure 1: Gradient flow: layer-wise mean and max $|\nabla|$ (logged to W&B as `gradients`).

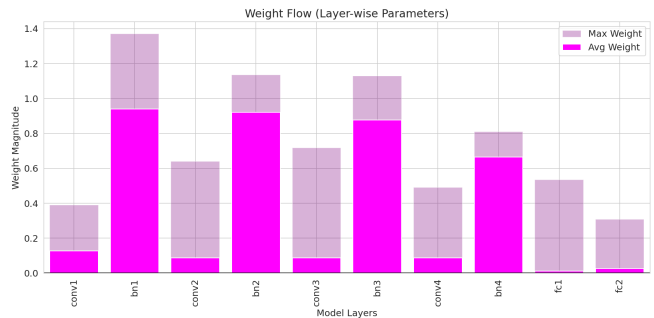


Figure 2: Weight flow: layer-wise mean and max $|W|$ (logged to W&B as `weights`).

vs dog, or automobile vs truck), which are consistent with known CIFAR-10 difficulty.

4.2 Per-Class Accuracy

Per-class accuracy is derived from the confusion matrix (diagonal divided by row sums) and logged as `class_accuracy`. Observations:

- Classes such as airplane, automobile, ship, and truck typically achieve higher accuracy.
- Classes with fine-grained visual similarity (e.g., cat/deer/dog) show relatively lower per-class accuracy, matching the confusion matrix.

4.3 Sample Predictions

A grid of sample test images with predicted and actual labels is logged to W&B as `predictions`. Green indicates correct and red incorrect predictions. This provides a qualitative check that the model learns meaningful features and that errors are concentrated on ambiguous or difficult examples.

5 Conclusion

This assignment implemented a CNN on CIFAR-10 with a custom dataloader, FLOPs counting, and gradi-

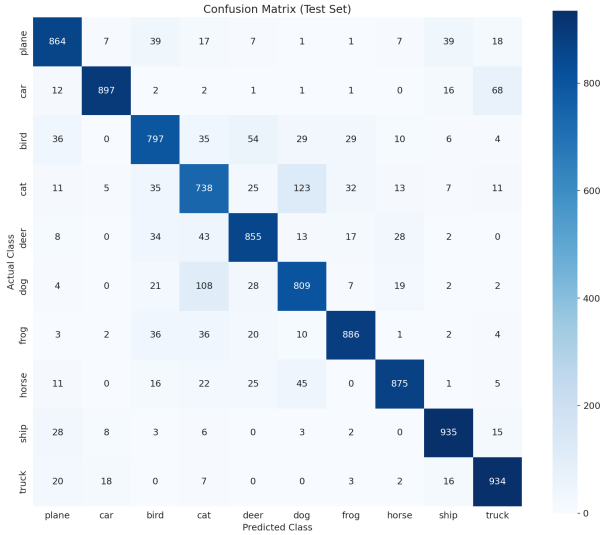


Figure 3: Confusion matrix on the CIFAR-10 test set (W&B: `confusion_matrix`).

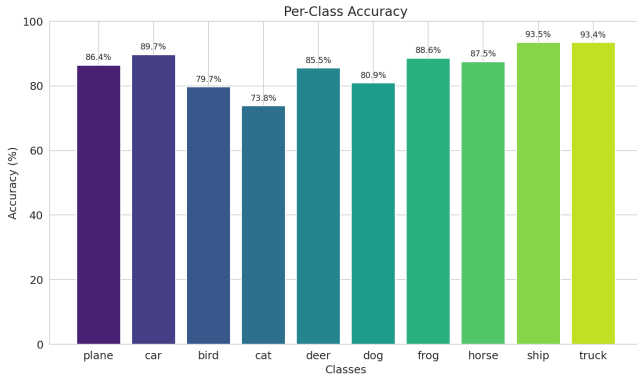


Figure 4: Per-class accuracy (W&B: `class_accuracy`).

ent/weight flow visualization, with all metrics and figures reported on W&B.

- **Model & dataset:** SimpleCNN on CIFAR-10 with official split; custom CustomCIFAR10 + DataLoader for training and evaluation.
- **FLOPs:** 0.1612 GFLOPs per sample, logged to W&B.
- **Training:** 30 epochs; final test accuracy 85.90%; stable training and validation curves.
- **Gradient and weight flow:** Layer-wise visualizations confirm healthy gradient propagation and stable weight magnitudes; all plots available in the W&B workspace.
- **Reproducibility:** Code and instructions are in the GitHub repository; trained model weights are not pushed, per assignment instructions. All visualizations are centralized in the provided W&B link.

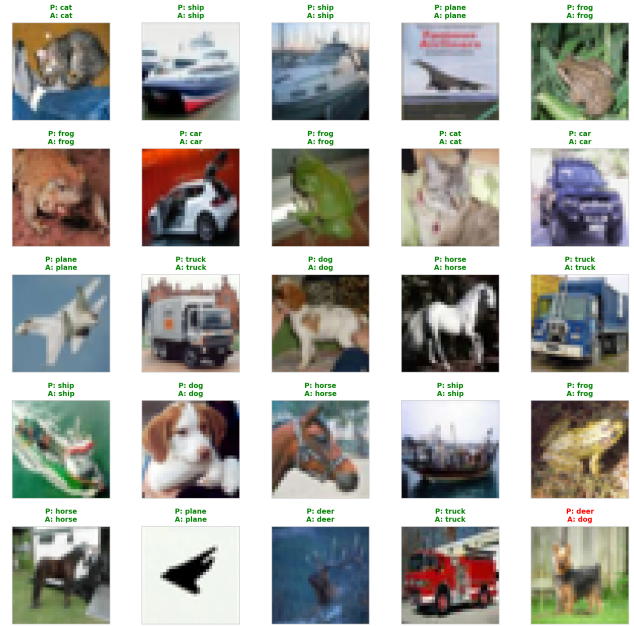


Figure 5: Sample predictions: green = correct, red = incorrect (W&B: `predictions`).