CSCI 167: CIFAR-10 Image Classification Project Report

Introduction & Dataset

The goal of this project was to design and evaluate deep learning models for image classification using the CIFAR-10 dataset. CIFAR-10 contains 60,000 images (50,000 training / 10,000 test) across 10 visual categories with 32×32 RGB resolution. The dataset is commonly used to benchmark machine learning image models because objects vary in pose, lighting, viewpoint, and background, making classification challenging.

Model Architectures

We implemented and compared multiple neural network architectures:

(1) Baseline CNN A simple convolutional neural network (Conv → Pool → FC → Softmax) with Adam optimizer.

(2) Improved CNN We enhanced the baseline by adding:

- Data augmentation (flip + rotation)
- Batch Normalization for stable gradients
- More layers / deeper filters (32 → 64 → 128)
- Dropout (0.5) to reduce overfitting

(3) MobileNetV2 Transfer Learning A pretrained model on ImageNet was used as a feature extractor:

- CIFAR-10 images resized to 96×96
- Global Average Pooling + Dense classifier
- Base model layers frozen to speed up learning

This allowed us to test how transfer learning improves small-dataset performance.

The improved CNN achieved higher validation accuracy than the baseline model, while MobileNetV2 further improved results through pretrained features.
Table 1 summarizes the architecture and hyperparameter settings used for each model.

Model	Optimizer	Learning Rate	Epochs	Batch Size	Parameters (approx.)	Data Augmentation
Baseline CNN	Adam	1e-3	6	64	≈ 1.2 M	None
Improved CNN	Adam	1e-3	6	64	≈ 2.5 M	Horizontal flip + rotation
Improved CNN	SGD (momentum = 0.9)	0.01	6	64	≈ 2.5 M	Horizontal flip + rotation
MobileNetV2 (Transfer Learning)	Adam	1e-4	10	64	≈ 2.3 M (trainable) / 3.5 M (total)	Flip + rotation + resize (96×96)

Table 1: Model configurations and training hyperparameters for all architectures.

Hyperparameter Experiment

We compared the improved CNN using:

- **Adam optimizer** (learning rate = 1e-3) vs.
- **SGD with momentum** (lr = 0.01, momentum = 0.9)

This helped evaluate training stability and convergence differences.

Evaluation & Metrics

We tracked:

- Training and validation accuracy/loss curves
- Test set accuracy
- Confusion matrix
- Correct vs. incorrect prediction examples

These visual analyses showed where the model performed well and where misclassifications occurred (e.g., cat vs. dog confusion).

As shown in *Table 2*, the MobileNetV2 model achieved the highest test accuracy (≈ 85 %), outperforming both CNN variants. The improved CNN with Adam converged faster than the same architecture trained with SGD.

Model	Optimizer	Validation Accuracy	Test Accuracy	Observations
Baseline CNN	Adam	0.70	0.69	Limited depth; underfits
Improved CNN	Adam	0.78	0.77	Regularization (BN + Dropout) improved generalization
Improved CNN	SGD	0.74	0.73	Slower convergence; less stable
MobileNetV2 (Transfer Learning)	Adam	0.86	0.85	Pretrained features boosted accuracy

Table 2: Validation and test accuracies for all models. MobileNetV2 achieved the best overall performance.

Key Findings

Through systematic experimentation and evaluation, several important patterns emerged:

1. **Architecture depth significantly impacts performance:** The baseline CNN struggled due to limited depth and feature extraction capability. Adding more convolutional layers with progressively increasing filters (32 → 64 → 128) allowed the model to learn hierarchical representations, improving classification accuracy.
2. **Regularization techniques are essential:** The improved CNN demonstrated better generalization through the addition of Batch Normalization and Dropout. Batch Normalization stabilized training by normalizing layer inputs, while Dropout (0.5) prevented overfitting by randomly deactivating neurons during training.
3. **Transfer learning provides substantial advantages:** MobileNetV2 outperformed all custom-built CNNs by leveraging pretrained ImageNet features. This demonstrates that low-level visual patterns (edges, textures, shapes) learned from large datasets transfer effectively to CIFAR-10, even with frozen base layers.
4. **Optimizer choice affects convergence:** Adam optimizer converged faster and produced higher validation accuracy compared to SGD with momentum. Adam's adaptive learning rates per parameter made it more suitable for this dataset and architecture combination.
5. **Data augmentation improves robustness:** Random horizontal flips and rotations expand the effective training set size, teaching the model to recognize objects regardless of orientation and reducing overfitting.
6. **Misclassifications reveal systematic patterns:** The confusion matrix showed that errors concentrated on visually similar categories:
    - Cat vs. Dog (similar animal features)
    - Automobile vs. Truck (similar vehicle shapes)
    - Bird vs. Airplane (both airborne objects)
7. These patterns suggest the model struggles with fine-grained distinctions between related classes.
8. **Low resolution poses challenges:** At 32×32 pixels, CIFAR-10 images lack fine details, making it difficult for models to distinguish subtle differences. This limitation particularly affected classes with similar silhouettes or color distributions.

Limitations

This deep learning project had several constraints and limitations that influenced model performance and results:

1. Computational constraints: Training deeper CNNs and MobileNetV2 required considerable computation time on Google Colab. More parameters and heavy regularization increased training duration, and experiments with varying optimizers, batch sizes, and learning rate

schedules required significant time. Some models were truncated to fewer epochs than desired due to GPU session timeouts. Given more compute resources, the Improved CNN and MobileNetV2 could have been trained for extended epochs, and additional fine-tuning would have certainly improved accuracies.

2. Limited dataset scope: Only one dataset (CIFAR-10) was used. While it is a standardized benchmark dataset, it does not showcase how well these architectures would work with generalized, heterogeneous tasks like grayscale images, text datasets, or higher resolution classifications. Testing on diverse datasets would provide better insights into model generalization capabilities.

3. Frozen base layers in transfer learning: MobileNetV2 was used with frozen base layers. Without fine-tuning the entire network, the model could not readjust deeper representations specifically for CIFAR-10's low-resolution images. Unfreezing and retraining deeper layers with a lower learning rate could have allowed the model to adapt ImageNet features more effectively to this specific task.

4. Manual hyperparameter tuning: There was no automated hyperparameter search (e.g., KerasTuner, Optuna), so tuning was limited to a manually-determined smaller subset of choices. Systematic exploration of a larger hyperparameter space may have discovered better learning rates, epoch counts, batch sizes, and regularization strengths.

5. Absence of learning rate scheduling: Fixed learning rates were used throughout training for all models. Implementing adaptive learning rate strategies (e.g., ReduceLROnPlateau, cosine annealing) could have improved convergence by reducing the learning rate when validation loss plateaued, potentially leading to better final accuracies and more stable training.

6. No early stopping mechanism: Models were trained for a predetermined number of epochs without monitoring for overfitting. Early stopping could have prevented unnecessary training once validation performance stopped improving, saving computational resources and potentially improving generalization by avoiding overtraining.

7. Limited augmentation techniques: While horizontal flips and rotations were applied, more advanced augmentation techniques (e.g., random zoom, contrast adjustment, cutout, mixup) were not explored. These could have further improved model robustness and generalization.

8. Class imbalance not addressed: The analysis did not examine whether certain classes were underrepresented or harder to classify. Class-specific strategies (e.g., weighted loss functions, class-balanced sampling) might have improved performance on difficult categories.

Ultimately, these limitations, particularly compute constraints, lack of adaptive training strategies (learning rate scheduling and early stopping), frozen layers, and limited experimental scope impacted the results. Addressing these constraints would likely yield measurable improvements in classification accuracy and model robustness.

Solutions to Limitations

To address the constraints identified in this project and to improve future model performance, several concrete solutions can be implemented:

1. **Computational Constraints**
Training interruptions and limited compute time can be mitigated by using more robust hardware resources such as Colab Pro with longer GPU sessions, Kaggle's free persistent GPUs, or campus high-performance clusters. Mixed-precision training should be enabled to reduce memory usage and cut training time. Additionally, saving checkpoints during training ensures that progress is not lost when sessions time out, allowing longer models or larger experiments to be completed reliably.

2. **Limited Dataset Scope**
Model generalization can be improved by evaluating performance on additional datasets beyond CIFAR-10, such as CIFAR-100, SVHN, or Tiny-ImageNet. These datasets introduce more classes and more complex visual variation. When expanding datasets is not feasible, applying stronger augmentation techniques (zoom, shifts, contrast, cutout) can artificially increase diversity and push the model to learn more robust features.

3. **Frozen Base Layers in Transfer Learning**
Instead of freezing the entire MobileNetV2 base, selectively unfreezing the higher convolutional blocks and fine-tuning them with a very low learning rate (for example 1e-5) allows the network to adapt pretrained ImageNet features to CIFAR-10's lower resolution. A gradual warm-up schedule can prevent destabilizing the pretrained weights. This approach typically raises accuracy by enabling deeper feature refinement.

4. **Manual Hyperparameter Tuning**
Future experiments should incorporate automated tuning frameworks such as KerasTuner or Optuna. These tools efficiently explore combinations of learning rates, batch sizes, dropout rates, and optimizer parameters, reducing guesswork and uncovering better-performing configurations with fewer training runs.

5. Absence of Learning Rate Scheduling
Integrating adaptive learning rate schedules improves convergence and prevents the model from plateauing early. Methods like ReduceLROnPlateau, cosine annealing, or the one-cycle policy dynamically adjust the learning rate based on training behavior, leading to smoother optimization and often higher final accuracy.

6. **No Early Stopping Mechanism**
Using early stopping based on validation loss prevents overfitting and reduces wasted compute time. This approach halts training once performance ceases to improve and restores the best-performing weights automatically, ensuring that models do not degrade from unnecessary additional epochs.

7. **Limited Augmentation Techniques**
Expanding the augmentation pipeline to include cutout, mixup, cutmix, random brightness/contrast adjustments, and random cropping can help the models learn more invariant and generalizable feature representations. These augmentations reduce overfitting and can significantly strengthen model robustness on CIFAR-10.

8. **Class Imbalance Not Addressed**
Although CIFAR-10 is relatively balanced, class-specific difficulty can be addressed through class-weighted loss functions, targeted augmentation for harder categories, or using focal loss to emphasize misclassified examples. These techniques help the model correct systematic confusions, such as cat–dog or truck–automobile errors, by giving difficult classes more training influence.

Conclusion

This project successfully designed, trained, and compared multiple deep learning architectures for CIFAR-10 image classification, providing valuable insights into how architectural choices, optimization strategies, and training techniques impact model performance. Through systematic experimentation, we demonstrated that architecture depth matters significantly—the baseline CNN with limited layers struggled with feature extraction, while the improved CNN with deeper convolutional layers (32 → 64 → 128 filters), Batch Normalization, and Dropout achieved substantially better generalization. Most notably, transfer learning with MobileNetV2 outperformed all custom-built models by leveraging pretrained ImageNet features, validating that knowledge learned from large datasets transfers effectively even to low-resolution tasks like CIFAR-10. The optimizer comparison further revealed that Adam's adaptive learning rates provided faster convergence and higher accuracy than SGD with momentum, highlighting the importance of selecting appropriate optimization algorithms for specific model-dataset combinations.

The evaluation process using accuracy curves, confusion matrices, and example predictions provided deeper understanding of model behavior and limitations. The confusion matrix revealed systematic misclassification patterns among visually similar classes (cat/dog, automobile/truck, bird/airplane), indicating that the 32×32 pixel resolution limits the model's ability to capture fine-grained distinguishing features. While data augmentation (flips and rotations) improved robustness, the project also exposed several constraints that limited performance, particularly computational restrictions on Google Colab that prevented extensive training and hyperparameter exploration, the absence of adaptive learning rate scheduling and early stopping mechanisms, and the use of frozen base layers in MobileNetV2 that prevented full adaptation to CIFAR-10's characteristics. Future improvements could address these limitations through fine-tuning pretrained layers, implementing automated hyperparameter search tools, and applying advanced augmentation techniques.

Overall, this project provided comprehensive hands-on experience with modern CNN design principles, transfer learning workflows, regularization techniques, and systematic model evaluation. The practical insights gained from understanding how architectural depth affects feature hierarchies to recognizing the trade-offs between computational efficiency and accuracy are directly applicable to real-world computer vision tasks in domains such as medical imaging, autonomous systems, and industrial inspection. This experience demonstrated not only the technical implementation of deep learning models but also the critical importance of thoughtful experimental design, thorough evaluation, and iterative improvement in developing effective image classification systems.