Deep Learning Spring 2025: CIFAR 10 classification Kaggle challenge 1: Deep Learning SP25

Rujuta Amit Joshi[1], Shashank Dugad[2], Anshi Shah[3]

[1]New York University
[2] New York University
[3] New York University

rj2719@nyu.edu, sd5957@nyu.edu, ans10020@nyu.edu

GitHub Project Repository: DLProject1

Abstract

In this project, we present a custom ResNet architecture for classifying images in the CIFAR-10 dataset under a Kaggle competition setting that required using a separate, custom test dataset and balancing a parameter count (under 5 million) with strong accuracy. Our network balances depth and parameter efficiency, stacking *residual connections* to mitigate vanishing gradients. We use data augmentation (random cropping, horizontal flipping, AutoAugment) and a multi-step learning rate schedule to boost generalization. We also detail each component of our approach: the overall data pipeline, residual block design, training hyperparameters, how learning rate scheduling stabilizes training and final results. On the private leaderboard for the custom test dataset, we achieve 83.62% accuracy, placing us 88th overall (an improvement of 15 places). Our findings reaffirm that well-structured skip connections and data augmentation achieve high performance on relatively small images like 32×32 in CIFAR-10, without an excessively large parameter footprint.

Introduction

Deep learning has transformed the field of computer vision over the last decade, enabling significant advances in image classification, object detection, and more. One of the most popular datasets used to benchmark such progress is CIFAR-10, introduced by Krizhevsky [1]. It comprises 60,000 color images (each 32×32 pixels) distributed across ten object classes. While CIFAR-10 is smaller in resolution than many modern benchmarks (e.g., ImageNet's 224×224 images), it remains a valuable testbed for novel architectures and regularization strategies due to its balanced class distribution, challenging intra-class variability, and comparatively modest dataset size.

1.1. Motivation and Challenges

Despite its seemingly small size, CIFAR-10 poses non-trivial hurdles:

- **Limited Resolution:** Each image is only 32×32, which can constrain the amount of discriminative detail available to the model. As a result, networks require sufficient depth or specialized mechanisms to learn fine-grained features.
- **Data Efficiency:** With just 50,000 training samples (5,000 per class), deeper architectures can overfit if not properly regularized. Techniques like data augmentation, weight decay, and careful learning rate scheduling are key to maintaining generalization.
- **Vanishing Gradients in Deeper Networks:** As CNNs grow deeper, gradient signals tend to become weaker as they propagate back through many layers. This challenge led to the development of ResNets [2], where identity (skip) connections preserve gradient flow even with 50 or more layers.

1.2. Residual Networks

He et al. [2] proposed residual connections- an identity mapping that circumvents the standard sequence of convolutions in each block. Instead of learning a direct mapping $\mathcal{F}(x)$, a residual block learns $\mathcal{F}(x) + x$, where x is the block's input. This architecture effectively reduces vanishing gradients, making deeper models trainable and often leading to superior accuracy. ResNets have yielded state-of-the-art results on many benchmarks (e.g., ImageNet), and they can also be adapted to smaller-scale datasets like CIFAR-10.

1.3. Custom Test Dataset Context

A unique aspect of this project is the requirement to predict on a given custom test dataset (50% disclosed) rather than the standard 10,000-image CIFAR-10 test set (60,000 total images (32×32×3), labeled into 10 classes with ~6,000 im-

ages perclass). Each image in the custom test set is unlabeled, containing only an ID. We, participants, had to train on the publicly available CIFAR-10 images, then generate a CSV of predictions for the new test set. A private leaderboard, computed from a subset of these test images, ensured models are genuinely evaluated on unseen data and are not inadvertently overfitted to the commonly used public test set.

1.4 Our Proposed Contribution

- **Custom ResNet:** We tailored the number of blocks and channels such that the model remains under 5 million parameters yet retains sufficient depth for high accuracy.
- **Data Augmentation:** We applied random cropping, horizontal flipping, and AutoAugment (CIFAR-10 policy) to enrich training samples.
- **Multi-Step LR Schedule:** A carefully timed learning rate decay at epochs 30, 60, 80, and 90 helped the model converge effectively.
- **Final Performance:** Achieving 83.62% on the private leaderboard, placing us at rank 88- a significant improvement over baseline trials.

1.5 Challenges

One of the primary challenges was striking the **right balance in data augmentation**- ensuring that transformations like AutoAugment, random cropping, and horizontal flips enhanced model robustness **without distorting essential features**. Designing a resilient ResNet-like architecture within the **strict 5M parameter limit** required careful trade-offs in the number of layers and channels. Additionally, **fine-tuning hyperparameters**, such as the learning rate schedule, momentum, and weight decay, was critical for stable convergence. **Managing computational resources**, including training time and GPU memory, further compounded these challenges, all while ensuring that the model maintained a **good balance between underfitting and overfitting**.

Methodology

Our approach combines a custom ResNet architecture with strategic data augmentation and a well-calibrated training schedule:

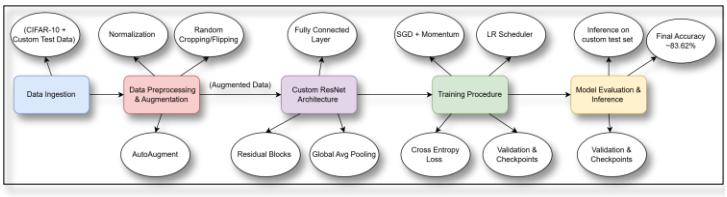

Fig. 1. System Architecture Diagram

2.1. Data Preprocessing & Augmentation

- **Normalization***:* We compute per-channel mean and standard deviation on the 50,000 training images. Each pixel is normalized accordingly, centering and scaling the data. Standardized images using μ = (0.4914, 0.4822, 0.4465) and σ = (0.247, 0.243, 0.261)
- **Random Cropping (with padding = 4):** A 4-pixel zero padding on each border, followed by a 32×32 crop, simulates shifting objects within the frame.
- **Horizontal Flip:** A 50% probability flip addresses left-right symmetry.
- **AutoAugment:** Dynamically applies transformations (e.g., brightness, cutout, shear, rotate) chosen from a curated set, significantly diversifying the training data.

In preliminary experiments, we found that advanced augmentation (AutoAugment) improved validation accuracy by ~5–6% beyond basic random cropping and flipping.

Pros:

- Robust augmentations improve model generalization and reduce overfitting.
- Horizontal flips and random crops are lightweight yet effective for CIFAR-10.

Cons

- Excessive augmentation may lead to slower convergence or confusion if transformations become too aggressive.
- AutoAugment can increase training time due to more complex transformations.

2.2. Custom ResNet Architecture

We adopted a residual learning framework to handle deeper layers effectively:

- **Initial Convolution:** 3×3 kernel, stride=1, from 3→39 channels, followed by batch normalization and ReLU.
- **Residual Layers:**
  a) Layer 1: 4 blocks, 39→39 channels, stride=1 for the first block.
  b) Layer 2: 4 blocks, 39→78 channels, stride=2 in the first block.
  c) Layer 3: 3 blocks, 78→156 channels, stride=2 in the first block.
  d) Layer 4: 2 blocks, 156→305 channels, stride=2 in the first block. Each block has two 3×3 convolutions plus an identity skip path. Where channel dimensions change, a 1×1 convolution is used on the skip path to match shapes.
- **Global Average Pooling:** Averages spatial dimensions into a single vector of size 305.
- **Fully Connected Layer:** Maps 305→10 outputs (one for each CIFAR-10 class).
- **Parameter Count:** The model consists of a total of 4,738,952 parameters (<5 Million), all of which are trainable, with no non-trainable parameters.

Pros

- Residual connections help avoid vanishing gradients, allowing deeper networks to train effectively.
- Batch Normalization speeds convergence and stabilizes training.

Cons

- Deeper networks require more computation and can overfit without proper regularization.
- Parameter constraints (under 5M) required careful selection of channel sizes.

2.3. Training Procedure

We used PyTorch with GPU P100 usage:

- Optimizer: SGD with momentum = 0.9 and weight decay=1e-4.
- Initial Learning Rate: 0.1, decayed by 0.1 at epochs 30, 60, 80, and 90.
- Batch Size: 256.
- Number of Epochs: 100.
- Loss Function: Standard cross-entropy between model outputs and one-hot labels.

Validation: A 5,000-image validation split was used to track accuracy and loss each epoch. The checkpoint with the highest validation accuracy was chosen for final inference on the custom test dataset.

Pros

- SGD with momentum is a standard, robust choice for image classification.
- MultiStepLR at carefully selected epochs ensures stable and thorough training.

Cons

- Fixed scheduling can be suboptimal if the chosen epochs do not align perfectly with the loss plateau.
- Hyperparameter tuning (e.g., exact epoch milestones) can be time-consuming.

Lessons Learned

- **Residual Connections** significantly improved training stability and final accuracy compared to non-residual architectures.
- **Data Augmentation Balance** is necessary as Although AutoAugment and random crops boosted performance, too much augmentation can require re-tuning of learning rates and other hyperparameters.
- Staying under a **strict parameter count** (<5 million) forced a more thoughtful architecture design, ensuring that each layer was sized to balance representational power and efficiency.
- **Learning Rate Scheduling:** Multi-step drops worked well, but alternative strategies (e.g., Cosine Annealing) could be explored for potentially smoother convergence.
- Tracking the train- val loss gap was crucial. If overfitting was detected, we adjusted augmentation intensity or increased regularization.

Results

3.1. Leaderboard Outcome

After training for 100 epochs, we generated predictions on the custom test dataset and submitted the CSV. The private leaderboard reported:

- Accuracy: 83.62%
- Rank: 88th *(improvement of 15 ranks from our earlier baselines)*

3.2. Training Dynamics

- **Loss vs. Epoch**: Training loss steadily declined over the first 60 epochs, then dropped more gently after each learning rate reduction. Validation loss followed a similar trend, indicating no major overfitting. (Fig. 2.)
- **Accuracy vs. Epoch**: Training accuracy reached ~95% by epoch 70, while validation accuracy peaked around 83-84% toward the last 20 epochs. The incremental boosts after each LR drop highlight the importance of multi-step scheduling. (Fig. 1.)
- **Train Loss & Validation Loss:** The training and validation loss decrease steadily, indicating effective model learning. Both losses stabilize after 40 epochs, showing minimal overfitting and good generalization. (Fig. 2.)

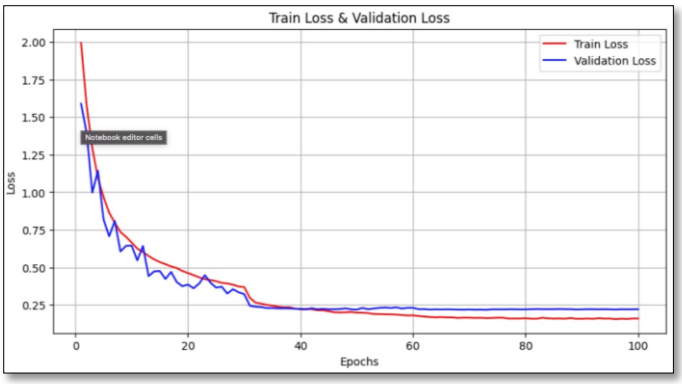

Fig. 2. Train Loss vs. Validation Loss

- **Train Accuracy & Validation Accuracy**: Accuracy improves rapidly in early epochs and stabilizes after 40 epochs. Validation accuracy closely follows training accuracy, ensuring strong generalization. (Fig. 3.)

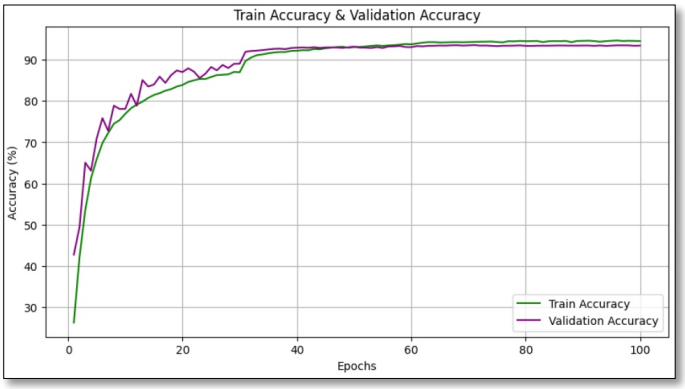

Fig. 3. Train Accuracy vs. Validation Accuracy

3.3. Comparisons

- **Non-Residual CNN:** A simpler CNN (no skip connections) plateaued at ~75%. Thus, the skip-based architecture confers substantial benefits for deeper feature extraction.
- **Parameter Efficiency:** Our ~4.7M-parameter model performed on par with some larger networks tested internally, showing effectiveness of carefully designed residual blocks.

Conclusion

We developed a custom ResNet that efficiently classifies CIFAR-10 images via residual connections, data augmentation, and a staged learning rate decay. Notably, we adhered to the competition requirement to submit predictions solely for a custom test dataset, revealing our model's genuine generalization capacity which resulted in 83.62% private leaderboard accuracy, rank 88 overall.

- Residual Connections facilitated deeper networks on small 32×32 images, mitigating training instabilities.
- AutoAugment accounted for a major accuracy jump beyond basic random transformations.
- Multi-Step LR Decays at carefully chosen epochs sustained progressive improvements to the final accuracy.

4.1. Future Directions

- **More Sophisticated Augmentations**: Techniques like Mixup or CutMix might further improve generalization.
- **Optimizer Variants**: AdamW, RAdam, or other adaptive methods may converge faster or yield incremental gains.
- **Layerwise LR Tuning**: Different schedules per residual layer could tailor learning dynamics, particularly in the earliest or deepest blocks.

We conclude that a well-structured ResNet architecture, paired with aggressive augmentation and a multi-step learning rate approach, remains a robust strategy for competitive performance on small-scale image classification tasks like CIFAR-10.

Code Repository

The complete implementation, including code and training details, is available in our GitHub repository: DLProject1. The notebook is designed to be executed on Kaggle for ease of use and reproducibility.

References

[1] A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images.* University of Toronto, 2009.
[2] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition.* In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.