Project 1 Report

Komal Niraula, Mohammed Shipat Uddin, Sanjana Battula

Team Name: kms resnet
ECE-GY 7123 / CS-GY 6953
Spring 2025
Professor Chinmay Hegde

GitHub Repository Link:
`https://github.com/komalniraula/LiteResNet`

Overview

In this project, we designed and trained a custom ResNet-based architecture for CIFAR-10 classification while ensuring the total parameter count remained below 5 million. The CIFAR-10 dataset consists of 60,000 32×32 color images, evenly distributed across 10 classes, with 50,000 for training and 10,000 for testing. Our objective was to develop a model that achieved high classification accuracy while remaining computationally efficient. Through careful architecture design, advanced data augmentation, and adaptive learning rate scheduling, we developed a modified ResNet architecture with 4,919,626 parameters, achieving a test accuracy of 95.04%.

Methodology

Data Augmentation

The dataset used in this project was CIFAR-10, comprising 60,000 color images of 32×32 pixels categorized into ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. It was structured into five training batches and one test batch, with 50,000 images allocated for training and 10,000 for testing. Before training the model, an extensive dataset analysis was conducted to identify potential issues affecting feature learning and classification performance.

First, we loaded the dataset and performed normalization using the mean and standard deviation of CIFAR-10 data to stabilize learning and gradients(Thakkar, Tewary, and Chakraborty 2018). This adjustment accounted for the higher brightness and contrast variability in the test set, preventing excessive distortions. A test set analysis revealed that images exhibited diverse brightness and contrast levels, leading to refinements in the augmentation strategy. Excessive color jittering was limited to avoid unnatural variations, while positional augmentations such as cropping, flipping, and rotation were prioritized to improve generalization.

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Further analysis identified background complexity issues, particularly white backgrounds concentrated in the top-left corner (Figure 1). This posed challenges in learning meaningful object features, as confirmed by histogram analysis. Some images also exhibited blurry edges, making object differentiation more difficult. To mitigate these issues, weak edge detection using Sobel filters was applied to emphasize object boundaries. Additionally, contrast enhancement was selectively applied to weak-edge images, adjusting the contrast value to 0.1 to improve fine detail visibility without over-processing.

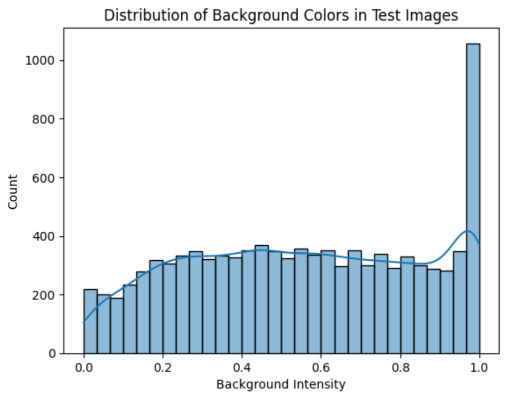


Figure 1: Histogram of background intensity in the final nolabel data

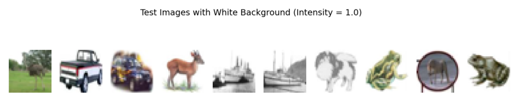


Figure 2: Examples of white background images in nolabel data

To enhance generalization and reduce overfitting, we employed a comprehensive augmentation pipeline. Initially, we considered separate transformations for structured objects (e.g., airplanes, cars, ships, and trucks) and natural objects (e.g., birds, cats). However, early evaluations revealed performance inconsistencies, prompting us to adopt a unified transformation strategy across all classes. Since structured objects like airplanes, cars, and ships should not be flipped vertically (as it would result in unrealistic transformations), we applied vertical flips with a low probability of 0.2. This ensured that vertical flips occurred only 20% of the time, minimizing the risk of unrealistic transformations while still providing some variability for natural objects. The final augmentation pipeline included random cropping with padding (4 pixels) to simulate different framing, horizontal flipping with a probability of 0.5, and minor rotations (5-15 degrees) to improve robustness. Additionally, we applied mild color jittering (brightness and contrast adjustments of 0.1) to maintain natural variations and occlusion-based augmentations such as Cutout (mask size of 12, probability 0.5) and random erasing (probability 0.2) to encourage the model to focus on object features rather than background elements.

By implementing these refinements, we ensured that the pipeline enhanced model robustness, improved generalization, and preserved meaningful object features while maintaining realistic transformations.

Model Architecture

The model architecture was based on a modified ResNet, designed to optimize efficiency while preserving high accuracy. The fundamental element of ResNet models is the residual block, which follows the formula:

$$\text{ReLU}(S(x) + F(x)) \tag{1}$$

Equation 1: Residual Block in a ResNet architecture.

Here, $S(x)$ represents the skip connection, allowing the input to bypass certain layers, while $F(x)$ is a block that applies convolutional operations.

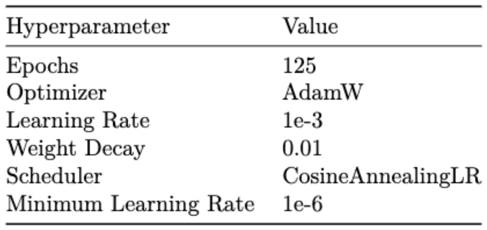
The model features three residual layers instead of four, reducing the total parameter count. The network was structured with BasicBlock residual units, with 7, 4, and 3 residual blocks in the respective layers, progressively increasing the number of channels from 64 to 128 and then to 256.

To further refine feature learning, Squeeze-and-Excitation (SE) blocks were integrated(Hu et al. 2019). These blocks dynamically recalibrate channel-wise feature importance, suppressing irrelevant features while emphasizing critical ones. This was particularly advantageous in addressing contrast variations and blurry edges within the dataset. The SE blocks also removed negative values to ensure that the network focused on positive, informative features.

Global average pooling was used instead of fully connected layers to reduce overfitting and enhance generalization. Additionally,batch normalization was incorporated to improve training stability and prevent overfitting.

Training Phase

Initially, training was conducted using the AdamW optimizer to speed up training and improve generalization(Zhou et al. 2024). However, early experiments without a learning rate scheduler resulted in significant accuracy fluctuations due to the absence of controlled learning rate adjustments. To address this, a cosine annealing learning rate scheduler was introduced, gradually adjusting the learning rate in a cosine pattern over 125 epochs, reaching a minimum of 1e-6(Shamaee and Hafshejani 2025). This approach smoothed learning and reduced loss variations. The scheduler updated the learning rate at the end of each epoch rather than every step, ensuring stability. Mixed precision training with PyTorch's Automatic Mixed Precision (AMP) (Micikevicius et al. 2018) was employed to optimize memory usage and accelerate training. The batch size was set to 64 to balance computational efficiency and convergence speed.

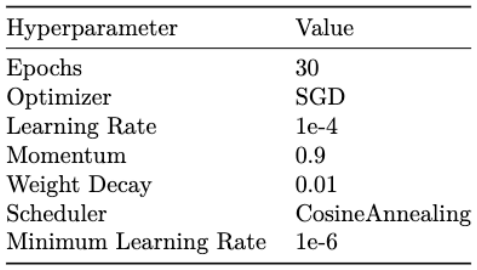


Hyperparameter	Value
Epochs	125
Optimizer	AdamW
Learning Rate	1e-3
Weight Decay	0.01
Scheduler	CosineAnnealingLR
Minimum Learning Rate	1e-6

Figure 3: Training Phase Hyperparameters

Fine-tuning Phase

Following initial training, fine-tuning was conducted using SGD instead of AdamW, offering more controlled weight updates for refining feature representations(Keskar and Socher 2017). Weight decay (0.01) and momentum (0.9) was applied, leveraging past gradient updates to maintain stable weight trajectories and reduce oscillations. Similar to initial training, a cosine annealing learning rate scheduler was employed to gradually decay the learning rate, starting from 1e-4 and decreasing to a minimum of 1e-6. To prevent excessive deviations from the pre-trained features, the first convolutional layer and the first residual layer were frozen, ensuring that low-level feature representations remained unchanged(Goutam et al. 2020). Only the second and third residual layers were updated, allowing deeper layers to adapt without affecting foundational learned patterns.



Hyperparameter	Value
Epochs	30
Optimizer	SGD
Learning Rate	1e-4
Momentum	0.9
Weight Decay	0.01
Scheduler	CosineAnnealing
Minimum Learning Rate	1e-6

Figure 4: Finetuning Phase Hyperparameters

Additionally, CutMix and MixUp were applied exclusively during fine-tuning, with 35% of images randomly selected for MixUp and another 35% for CutMix in every batch. However, cat and dog images were excluded from these transformations since the model showed frequent confusion between these classes. This strategy allowed the model to improve feature distinctions for challenging categories while still benefiting from augmentation-based regularization(Shen et al. 2024).

Evaluation and No-Label Set Prediction

Model evaluation was performed continuously throughout training by tracking validation accuracy and loss using CrossEntropyLoss. The best-performing model was saved based on the highest test accuracy recorded. No additional augmentations were applied during evaluation, ensuring that performance was assessed on the standard test set.

$$\mathcal{L}_{\text{CE}}(y,\hat{y}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{2}$$

Equation 2: Cross-entropy loss function.

For predicting the no-label dataset, test-time augmentation (TTA) was incorporated to enhance prediction stability. Each test image underwent seven transformations, including rotations, horizontal flips, and center cropping. The final classification was obtained by averaging predictions across these augmented versions, reducing variance and improving robustness. TTA helped mitigate biases introduced by individual transformations and reduced over-reliance on specific image features that may vary under different conditions (Shanmugam et al. 2021). By applying TTA, the model produced more stable and reliable predictions, particularly for images with challenging background variations or weak object boundaries.

Key Design Insights and Lessons Learned

The iterative design process provided several key insights that shaped the final model's performance. Initially, it was assumed that standard augmentations would be sufficient for generalization; however, early results showed that blurry images and white backgrounds negatively impacted feature learning. This led to the implementation of targeted techniques such as contrast enhancement and weak edge detection, significantly improving image clarity and classification performance. Additionally, diverse augmentation strategies played a crucial role—while standard augmentations helped prevent overfitting, techniques like CutMix and Cutout further enhanced generalization by encouraging the model to focus on object-specific features rather than background cues.

Another key insight was the importance of model depth. The performance consistently improved as more layers and blocks were added, highlighting the benefits of a deeper architecture in capturing complex patterns. The introduction of the Squeeze-and-Excitation (SE) block further boosted performance by adaptively recalibrating channel-wise feature responses, allowing the network to prioritize the most informative features. This architectural enhancement contributed to better feature extraction and improved classification accuracy.

By systematically refining augmentations and deepening the network, we achieved a well-optimized CIFAR-10 classifier, demonstrating that both data preprocessing and model architecture play crucial roles in improving generalization and robustness.

Results

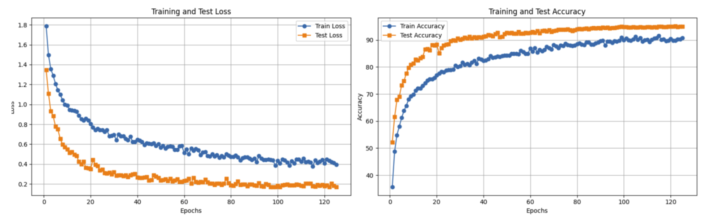


Figure 5: Loss and Accuracy Graphs

Parameter	Our.Model	ResNet18
Number of residual layers	3	4
Number of residual blocks	[7, 4, 3]	[2, 2, 2, 2]
Convolutional kernel sizes	[3, 3, 3]	[3, 3, 3, 3]
Shortcut kernel sizes	[1, 1, 1]	[1, 1, 1, 1]
Number of channels	[64, 128, 256]	[64, 128, 256, 512]
Average pool kernel size	8	4
Batch normalization	True	True
Squeeze and excitation	True	False
Data augmentation	True	False
Data normalization	True	False
Optimizer	AdamW	SGD
Learning rate (lr)	0.001	0.1
LR scheduler	CosineAnnealingLR	CosineAnnealingLR
Weight decay	0.01	0.0005
Batch size	64	128
Number of workers	4	16
Total number of Parameters	4,919,626	11,173,962

Figure 6: Configuration of our Residual Network Design that achieves a test accuracy of 95.04% on CIFAR-10

Our final model included 3 residual layers in which each layer had 7, 4, and 3 residual blocks respectively. The model has also include a squeeze and excitation (SE) block, with the final layout being in this order: SE block, residual layers, average pool, fully connected linear layer. More details with regards to the hyperparameters can be seen in figure 6. The total number of learnable parameters was 4.91 million. During training, the model history and graphs suggested that convergance happened around 80 epochs, and thereafter diminishing returns on loss and accuracy. After training, we achieved 95.04% accuracy on the validation set, and after fine-tuning we were able to achieve 91.125% accuracy on validation set and training set combined. The final trained model allowed us to get 85.144% on the no-label submission set.

References

[Goutam et al. 2020] Goutam, K.; Balasubramanian, S.; Gera, D.; and Sarma, R. R. 2020. Layerout: Freezing layers in deep neural networks. *SN Comput. Sci.* 1(5):295.

[Hu et al. 2019] Hu, J.; Shen, L.; Albanie, S.; Sun, G.; and Wu, E. 2019. Squeeze-and-excitation networks.

[Keskar and Socher 2017] Keskar, N. S., and Socher, R. 2017. Improving generalization performance by switching from adam to sgd.

[Micikevicius et al. 2018] Micikevicius, P.; Narang, S.; Alben, J.; Diamos, G.; Elsen, E.; Garcia, D.; Ginsburg, B.; Houston, M.; Kuchaiev, O.; Venkatesh, G.; and Wu, H. 2018. Mixed precision training.

[Shamaee and Hafshejani 2025] Shamaee, M. S., and Hafshejani, S. F. 2025. A Second Examination of Trigonometric Step Sizes and Their Impact on Warm Restart SGD for Non-Smooth and Non-Convex Functions. *Mathematics* 13(5):1–20.

[Shanmugam et al. 2021] Shanmugam, D.; Blalock, D.; Balakrishnan, G.; and Guttag, J. 2021. Better aggregation in test-time augmentation.

[Shen et al. 2024] Shen, L.; Yu, J.; Yang, H.; and Kwok, J. T. 2024. Mixup augmentation with multiple interpolations.

[Thakkar, Tewary, and Chakraborty 2018] Thakkar, V.; Tewary, S.; and Chakraborty, C. 2018. Batch normalization in convolutional neural networks — a comparative study with cifar-10 data. In *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*, 1–5.

[Zhou et al. 2024] Zhou, P.; Xie, X.; Lin, Z.; and Yan, S. 2024. Towards understanding convergence and generalization of adamw. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46(9):6486–6493.

This report was generated with some assistance from ChatGPT