Stress-Testing a Convolutional Neural Network on CIFAR-10

Zenith (m25csa032) Nishant Thapa (m25csa019) Nikhil Jha (R25CS0001)

Course: CSL7590 – Deep Learning

Code Repository

All code used to generate the results in this report is available at:

`https://github.com/zzethh/Stress-Testing-a-Convolutional-Neural-Network-on-CIFAR-10`

The repository contains training, evaluation, and explainability scripts, along with instructions to reproduce all experiments.

1 Objective

The goal of this assignment is to move beyond raw accuracy and critically analyze how Convolutional Neural Networks (CNNs) behave in practice. We aim to understand where the model fails, which visual cues it relies on, and how a single constrained modification affects robustness. Concretely, we train a baseline CNN and a modified version on CIFAR-10, then carry out failure analysis, explainability using Grad-CAM, and a controlled architectural improvement.

2 Dataset and Experimental Setup

2.1 Dataset

We use the CIFAR-10 dataset, which consists of 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 provided by CIFAR-10 is used; no external data, additional datasets, or pretrained models are used.

2.2 Model Architecture

Our baseline architecture is a custom CNN, denoted **FlexibleCNN**, composed of:
- An initial 3×3 convolution with 64 channels and batch normalization;
- Three subsequent convolutional stages with 64, 128, and 256 channels, respectively, each formed by several 3×3 blocks with batch normalization;
- Strided convolutions for spatial downsampling;
- Global average pooling followed by a fully connected layer to 10 output logits.

This design gives reasonable capacity for CIFAR-10 while remaining lightweight under the epoch budget.

2.3 Training Protocol

All experiments use PyTorch with the following common setup:
- Optimizer: SGD with momentum 0.9 and weight decay 5×10^{-4};
- Learning rate schedule: One-Cycle LR with maximum learning rate 0.01;
- Epochs: 40 (below the 50-epoch guideline);
- Batch size: 64;

- Fixed random seed: 2026, applied to Python, NumPy, and PyTorch, with deterministic CuDNN configuration;
- Train–validation split: a deterministic 90/10 split of the official training set, implemented via a fixed permutation, ensuring reproducible validation curves.

2.4 Data Preprocessing and Augmentation

To encourage robustness while respecting the assignment constraints, we use:
- **Training transform** (applied identically to baseline and modified models): random crop with padding, random horizontal flip, mild color jitter, and normalization with CIFAR-10 channel means and standard deviations.
- **Validation and test transforms**: only normalization (no augmentation), to provide an unbiased estimate of generalization.

These choices are part of the baseline training protocol; the only constrained modification we study is architectural (residual connections), as described next.

3 Baseline and Constrained Modification

3.1 Baseline Model

The baseline FlexibleCNN uses purely feed-forward convolutional blocks without residual shortcuts. It is trained from scratch on CIFAR-10 using the protocol in Section 2.3. This serves as our reference for both performance and failure behavior.

3.2 Constrained Modification: Residual Connections

To improve robustness in a controlled way, we apply exactly one change: we enable residual shortcuts within each convolutional block, making FlexibleCNN behave more like a lightweight residual network. Concretely, when spatial dimensions or channel counts differ, a learned 1×1 projection is used and the block output becomes

$$\mathrm{ReLU}\big(\mathrm{ConvBN}_2(x) + \mathrm{shortcut}(x)\big).$$

All other aspects (dataset, transforms, optimizer, learning rate schedule, seed, and epoch budget) are kept identical between baseline and modified models. This allows a clean comparison focused on the effect of residual connections.

4 Training Dynamics and Overall Performance

4.1 Training and Validation Curves

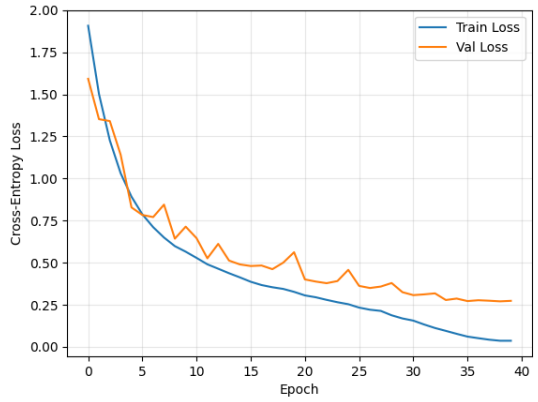
Figures 1 and 2 summarize the training behavior of both models using loss and accuracy curves.

Both models converge smoothly with no severe overfitting; the residual model tends to slightly outperform the baseline in later epochs.

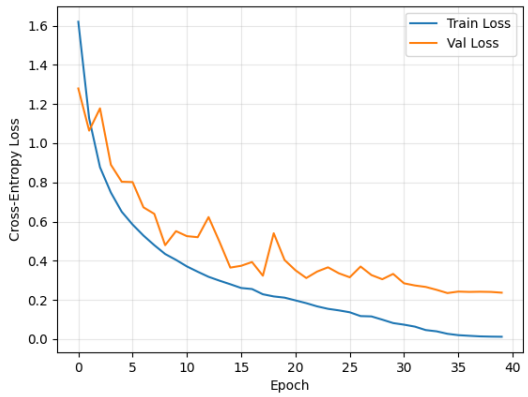
4.2 Final Accuracy and Confusion Patterns

Using the best validation-checkpointed models, we evaluate on the held-out CIFAR-10 test set:
- Baseline: 92.06% test accuracy;
- Modified (residual): 93.41% test accuracy.

Figure 3 shows confusion matrices, per-class accuracy, and a direct comparison of validation accuracy trajectories.

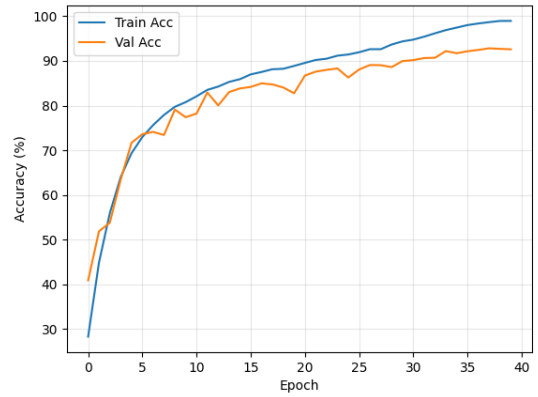


(a) Baseline loss (train/val).

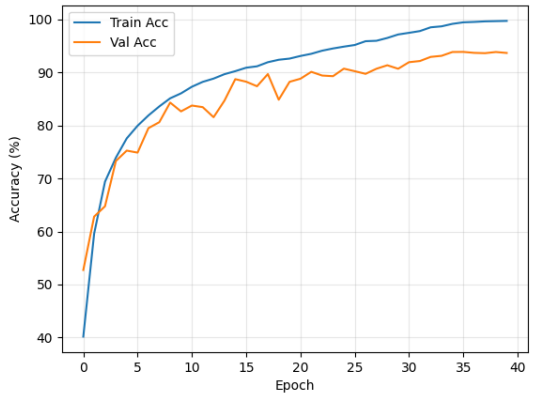


(b) Residual loss (train/val).

Figure 1: Training and validation loss vs. epoch for the baseline and residual CNNs.



(a) Baseline accuracy (train/val).



(b) Residual accuracy (train/val).

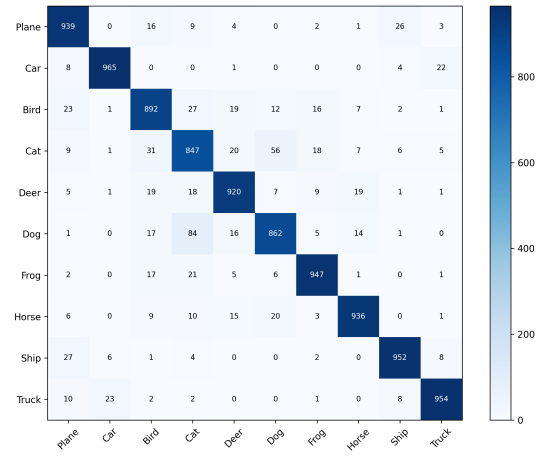
Figure 2: Training and validation accuracy vs. epoch for the baseline and residual CNNs.

5 Failure Case Discovery

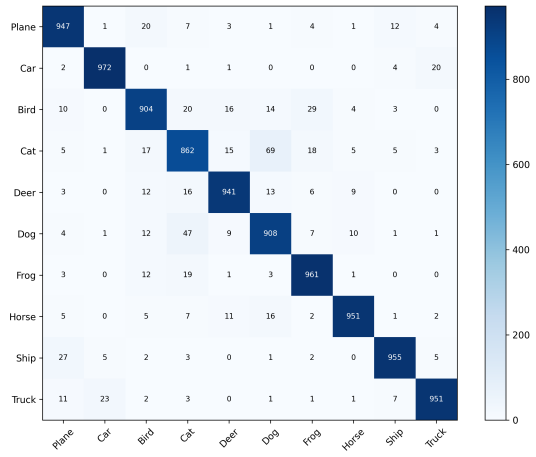
5.1 High-Confidence Misclassifications

We collect failure cases where the model is confidently wrong. For each test image, we record the ground-truth label, predicted label, softmax confidence of the predicted class, and test index. We sort misclassifications by confidence and visualize high-confidence failures for both models (Figure 4).

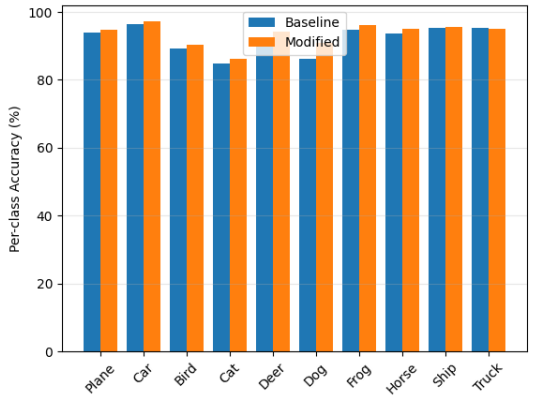
From the ranked baseline failures, we select three canonical cases (indices stored in the code outputs) to analyze in depth. These cover different error types, such as visually similar categories and misleading backgrounds. For each case we document: the true vs. predicted label and confidence, whether the error is due to genuine ambiguity, and any spurious correlates (e.g., dominant background or color).



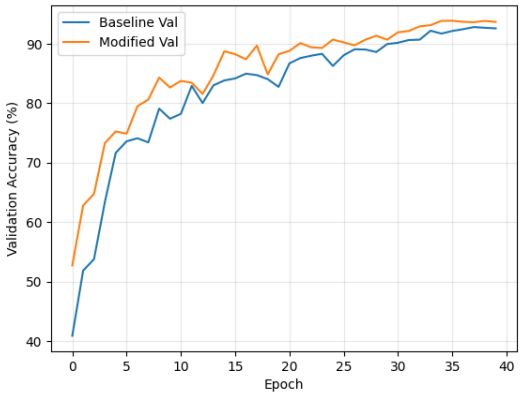
(a) Baseline confusion matrix.



(b) Residual confusion matrix.



(c) Per-class accuracy (baseline vs residual).



(d) Validation accuracy comparison.

Figure 3: Overall performance comparison. The residual model improves both overall and per-class accuracy and shows slightly more stable validation accuracy in later epochs.

6 Explainability via Grad-CAM

6.1 Method

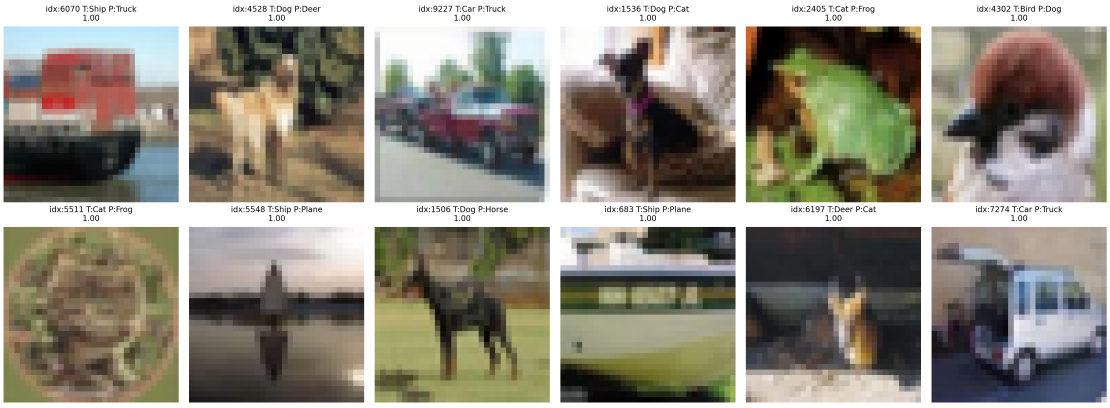
To understand which regions drive the model's decisions, we apply Grad-CAM on the last convolutional layer. For each chosen failure case (three high-confidence mistakes selected from the baseline ranking), we compute Grad-CAM maps for both the baseline and residual models. This yields coarse heatmaps indicating where gradients for the predicted class are concentrated.

6.2 Baseline vs Residual Attention Patterns

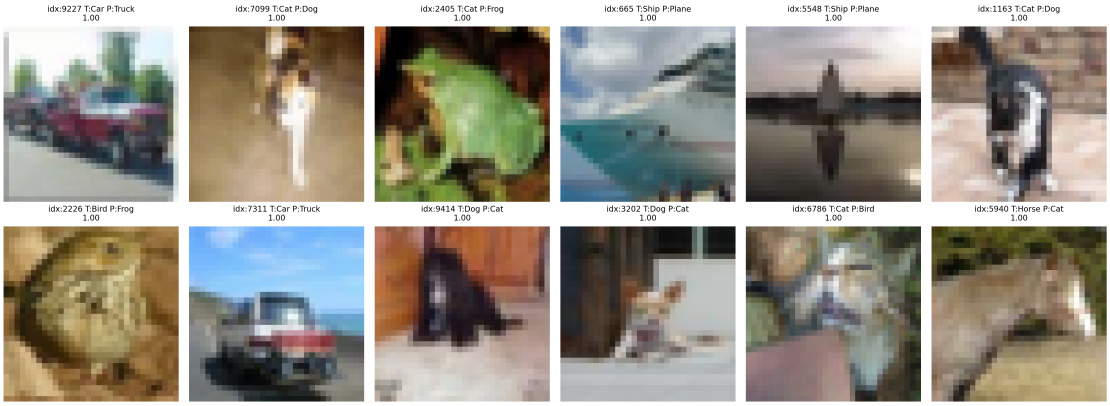
Figure 5 shows side-by-side Grad-CAM visualizations for the three canonical failure cases, using a compact 3-row comparison. Each row corresponds to a single test image, with three columns for the baseline (image, heatmap, overlay) and three for the residual model.

Across cases, we observe that:

- The baseline model often attends to large background regions or global color/texture cues;
- The residual model tends to produce slightly more localized activations around the main object, consistent with its modest accuracy gains;



(a) Baseline high-confidence failures.



(b) Residual high-confidence failures.

Figure 4: High-confidence failure grids. Each image is annotated with test index, true class, predicted class, and predicted-class confidence.

- Some failures remain persistent: both models can be misled by similar visual shortcuts, indicating that architectural changes alone do not fully resolve these issues.

7 Effect of the Constrained Improvement

7.1 Quantitative Impact

Enabling residual connections yields:
- An absolute test accuracy improvement of approximately 1.3 percentage points;
- Fewer misclassifications across several classes, as seen in the confusion matrices;
- Slightly more stable validation accuracy in late epochs, indicating improved generalization.

7.2 Impact on Identified Failure Cases

For the three canonical failure cases, we compare predictions and Grad-CAM maps before and after the modification:
- In some cases, the residual model corrects the label and shifts attention more squarely onto the object;
- In others, the prediction remains incorrect but the attention map becomes less diffuse, suggesting more focused but still misleading evidence;

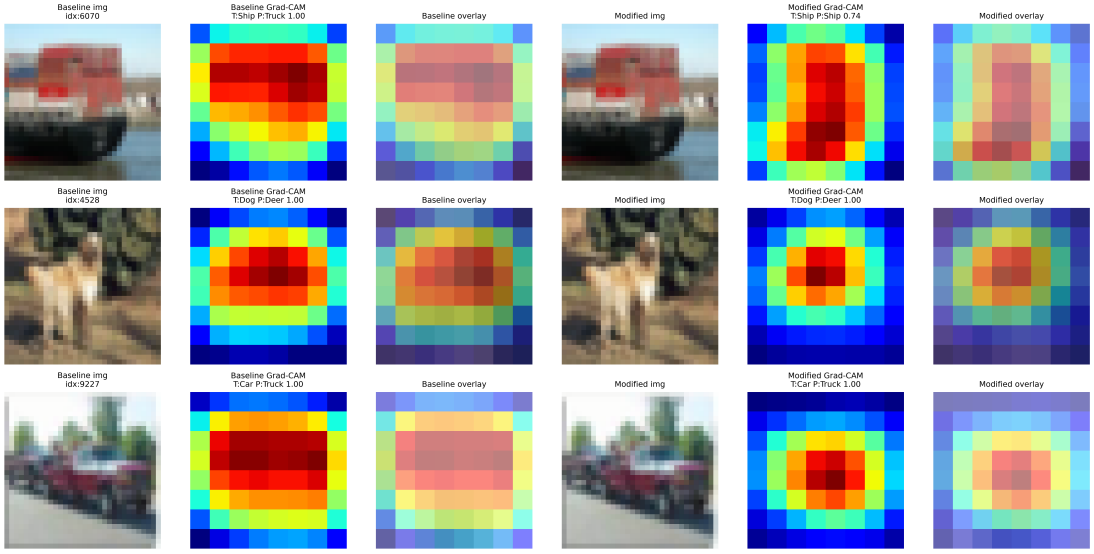


Figure 5: Grad-CAM comparison for three canonical failure cases. For each case (row), we show the original image, Grad-CAM heatmap, and overlay for the baseline model (left triplet) and residual model (right triplet). Often the residual model focuses more tightly on the object, though some failures remain dominated by background or texture.

- A few failures are unchanged in both prediction and attention, highlighting limitations of architectural tweaks and the need for data-centric or task-level changes for further robustness.

8 Reflection and Insights

Several aspects of CNN behavior were surprising:

- The model can reach over 92% test accuracy while still producing very confident misclassifications on apparently simple images;
- Backgrounds, colors, and textures sometimes dominate the Grad-CAM maps, revealing reliance on shortcuts rather than true object understanding;
- A relatively small architectural change (residual connections) can improve both accuracy and the apparent quality of attention, but does not fundamentally change the failure modes.

From a deployment perspective, the most concerning failures are those where the model is confidently wrong on visually plausible, non-adversarial images, especially when explanations suggest it is focusing on irrelevant regions. In safety-critical settings, we would not fully trust this model without additional safeguards (e.g., calibrated uncertainty estimates, human-in-the-loop checks, or robustness-oriented data curation). For non-critical applications, the residual model's improved performance and more interpretable attention patterns make it the preferable choice.

9 Conclusion

We have stress-tested a CIFAR-10 CNN by examining its training dynamics, confusion patterns, high-confidence failures, and Grad-CAM explanations, before and after a single constrained architectural modification. While residual connections provide measurable benefits, the persistence of structured, confident errors underscores the importance of looking beyond top-line accuracy and using systematic failure analysis and explainability to assess whether a model is truly reliable.