Stress-Testing Convolutional Neural Networks

CIFAR-100 • ResNet-18 from Scratch • Failure Analysis & Constrained Improvement

Deep Learning Assignment 01 | TEAM: 19

1 What This Report Is About

We trained a ResNet-18 on CIFAR-100 from scratch, watched it fail on nearly 40% of test images, and then dug into *why*. The goal here isn't just to report a number—it's to understand the model's decision-making, figure out what goes wrong when it's confidently incorrect, and see if a single, well-motivated tweak can make things meaningfully better. Spoiler: it can. But the interesting part isn't the accuracy bump—it's what the model's mistakes reveal about how CNNs actually "see" images.

2 Why CIFAR-100 and ResNet-18?

The dataset. CIFAR-100 gives us 100 classes crammed into 32×32 pixels—that's absurdly small. You can barely tell a lobster from a crab at that resolution, and that's exactly the point. It forces the model into situations where it *has* to rely on texture and colour rather than shape, which makes the failure cases really informative. We used 45k/5k/10k for train/val/test splits.

The architecture. ResNet-18 is the sweet spot here: deep enough to learn 100 classes, small enough to overfit visibly (which we want to study), and its skip connections let us train from scratch without the gradient issues that plague VGG at this resolution. We swapped the initial 7×7 conv for a 3×3 and removed max-pooling—standard practice for 32×32 inputs. No pretrained weights; everything learned from scratch.

3 Baseline: A Competitive Starting Point

Training setup: 50 epochs, SGD (momentum 0.9, weight decay 5×10^{-4}), LR = 0.1 decayed $10\times$ at epochs 25 and 40. Augmentation: random crop + horizontal flip. Loss: cross-entropy. Seed 42, deterministic CuDNN. This isn't a deliberately weak baseline—weight decay and LR scheduling give it every chance to generalise.

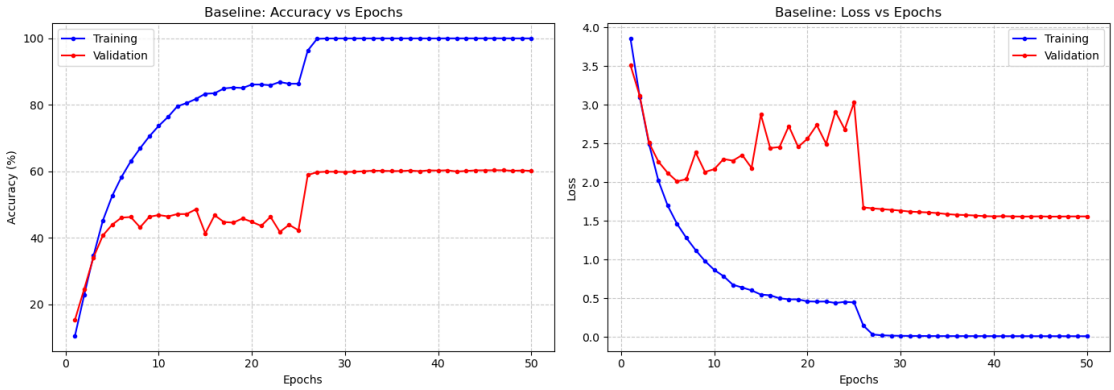

Figure 1: Baseline training curves. That ∼40% gap between train and val accuracy is hard to miss—the model basically memorised the training set. Validation loss actually *increases* after epoch 25 while training loss approaches zero.

Baseline Result

Test accuracy: 60.69%. Training hit 99.8% while validation flatlined at ∼60%—a 40-point gap. The model memorised the training data instead of learning transferable features. The diverging loss curves (Figure 1) confirm this is textbook overfitting.

4 Where Does It Go Wrong?

Out of 10,000 test images, 3,931 were misclassified—a 39.3% error rate. We picked five cases that were not just wrong, but *confidently* wrong (89–99.9% confidence). These are the scariest kind of errors: the model has no doubt, and it's completely off.

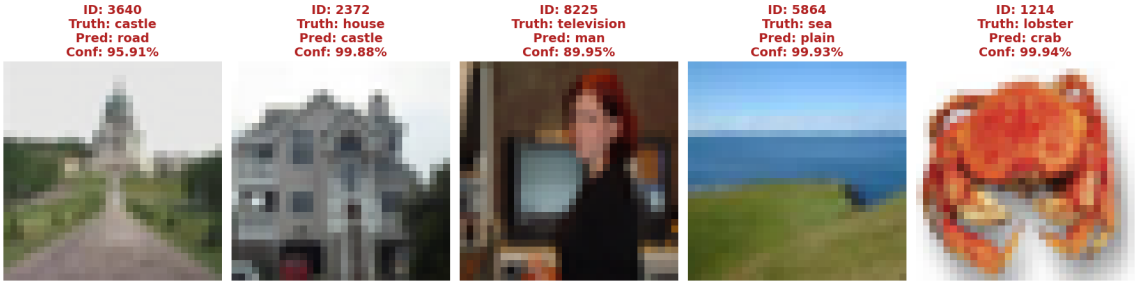

Figure 2: Our five hand-picked failure cases. Every single one is predicted with high confidence on the wrong class. Two of these (IDs 2372, 5864) get fixed later by CutMix; the rest get their confidence knocked down substantially.

What We Think Is Happening (Hypotheses)

ID	True	Pred	Conf	Our Hypothesis
3640	Castle	Road	95.9%	The gravel path takes up 2/3 of the image; the castle is tiny at the top. The model just sees "road texture" and runs with it.
2372	House	Castle	99.9%	A large stone house that honestly does look like a castle. At 32×32, you lose the modern windows and scale cues that would give it away.
8225	TV	Man	89.9%	There's a person standing right in front of the TV. The model locks onto the person—bigger, more features, easier to classify.
5864	Sea	Plain	99.9%	The bottom half is all green grass; the sea is a thin blue strip up top. Model sees green and calls it a plain.
1214	Lobster	Crab	99.9%	Same red colour, same shell texture, same roundish shape. At this resolution, even humans would struggle to tell them apart.

Table 1: What we think went wrong in each case. These hypotheses were confirmed by Grad-CAM (Section 5).

5 Opening the Black Box: Grad-CAM

To check whether our hypotheses hold up, we ran Grad-CAM on `layer4`—targeting the *wrong* predicted class to see exactly what the model was looking at when it made each mistake.

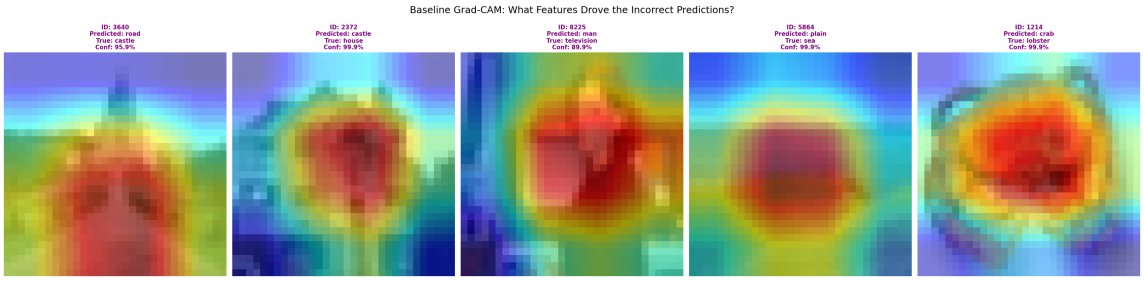

Figure 3: Grad-CAM heatmaps for all five failures. Red = where the model is paying attention. The pattern is clear: it fixates on whatever texture dominates the image spatially, not on the actual object of interest.

The Pattern We Found

Every single heatmap tells the same story: **the model fixates on the spatially dominant texture and ignores everything else.** Castle→road? Attention is on the ground. TV→man? All on the person. Sea→plain? The green foreground gets everything. For lobster→crab, attention is spread uniformly—there simply aren't any discriminative features to find at this resolution. This is the CNN texture bias problem in action.

6 The Fix: CutMix Augmentation

Why CutMix?

The Grad-CAM analysis pointed directly at the problem: **spatial texture bias**. The model latches onto whatever texture takes up the most space. CutMix attacks this head-on: during training, we randomly cut a rectangular patch from one image and paste it onto another, mixing the labels proportionally. Now the model can't just look at one dominant region—it has to pay attention to *everything*, because any part of the image might contain relevant class information.

We also used label smoothing (0.1) as part of the CutMix setup. This isn't a second trick—it's a natural fit. CutMix already creates soft, blended labels (say, 70% castle + 30% road for a partially pasted image), and label smoothing complements this by gently regularising the output probabilities. Soft inputs deserve soft outputs. Everything else—optimiser, LR, schedule, weight decay, epochs—stays *exactly* the same as the baseline.

CutMix config: Beta distribution $\alpha = 1.0$, applied with probability 0.5 per batch. Patch coordinates sampled uniformly; λ adjusted for boundary clipping.

What Changed

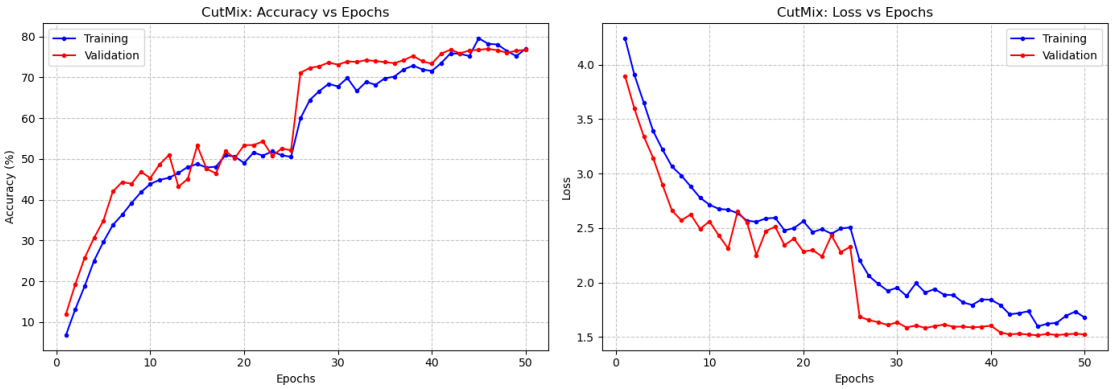

Figure 4: CutMix training curves. The train-val gap collapsed from ∼40% to under 3%. Training accuracy topping out at ∼75% instead of 99.8% isn't a weakness—it means the model can't cheat by memorising anymore.

CutMix Result

Test accuracy: 75.70% — a **+15.01 point jump**. But the numbers beneath the headline are arguably more important:

What We Measured	Baseline	CutMix	Change
Test Accuracy	60.69%	75.70%	+15.01%
Total Wrong (/10k)	3,931	2,430	−1,501
Train–Val Gap	∼40%	∼3%	Collapsed
Wrong with ≥90% confidence	315	44	−86%
Wrong with ≥99% confidence	54	1	−98%
Average confidence on wrong preds	47.8%	40.8%	−7%

Table 2: The full picture. CutMix didn't just improve accuracy—it fundamentally changed how the model handles uncertainty.

7 Head-to-Head: Baseline vs. CutMix

Did It Fix Our Five Problem Cases?

ID	Failure	Before	After	What Happened
3640	Castle→Road	95.9%	63.5%	Still wrong, but way less sure (−32%)
2372	House→Castle	99.9%	62.4%	**Fixed!** Now correctly predicts house
8225	TV→Man	89.9%	37.2%	Still wrong, but barely confident (−53%)
5864	Sea→Plain	99.9%	49.6%	**Fixed!** Now correctly predicts sea
1214	Lobster→Crab	99.9%	92.1%	Still wrong, barely budged (−8%)

Table 3: Two spatial-bias failures got fully corrected. Two more had their confidence dramatically reduced. The lobster/crab case barely moved—that's a resolution problem, not a training problem.

Grad-CAM: Before and After

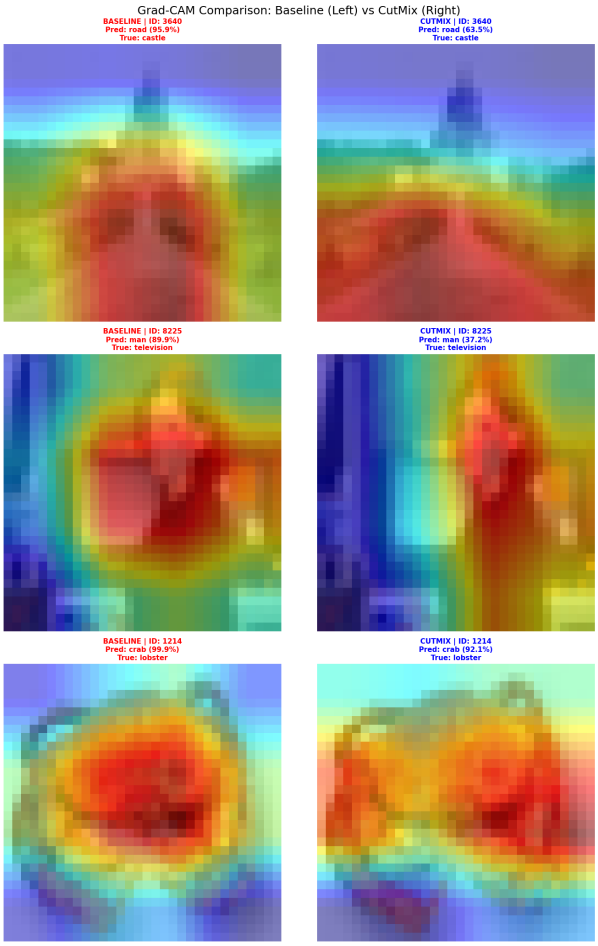

Figure 5: Baseline (left) vs. CutMix (right) Grad-CAM for the three cases still wrong. The CutMix model spreads its attention more—less tunnel vision on one texture region. Most obvious for the castle/road and TV/man cases.

The Confidence Story

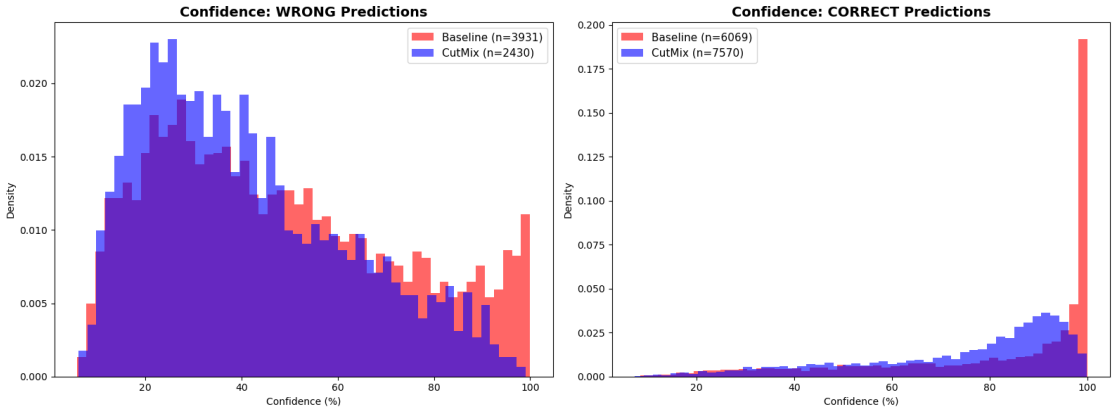

Figure 6: This is our favourite plot. Left: when the model is wrong, CutMix pushes confidence way down. Right: when it's right, confidence stays high. That's *exactly* what you want—selective calibration.

Look at the left panel of Figure 6: the baseline has this alarming spike near 100% confidence for *wrong* predictions—54 images where the model was dead certain and dead wrong. CutMix virtually eliminates that tail (down to just 1). Meanwhile, the right panel shows that confidence on *correct* predictions stays high. The model learned to be humble when it's unsure, without losing confidence when it actually knows the answer.

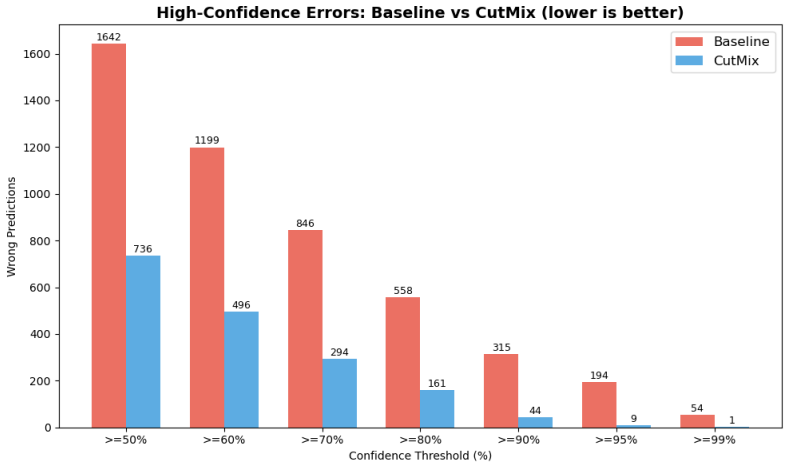

Figure 7: Wrong predictions by confidence threshold. The higher the stakes, the bigger the improvement: 86% reduction at $\geq$90%, 98% at $\geq$99%. The most dangerous errors are almost gone.

Class-by-Class: Who Benefited?

Net sample migration: 1,938 corrected, 437 regressed, **+1,501 net gain**. Even among the 1,993 samples still wrong in both models, 57.5% showed lower confidence—better calibrated even when still incorrect.

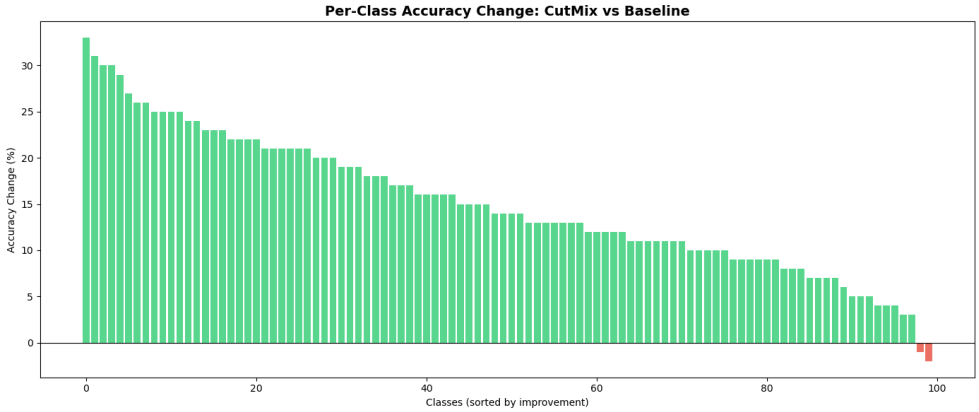

Figure 8: Per-class accuracy change across all 100 CIFAR-100 classes. Green = improved, red = regressed. **98 out of 100 classes got better or stayed the same.** Only oak tree ($-$2%) and maple tree ($-$1%) dipped—visually similar classes that likely just swapped confusion boundaries. Top gainers: otter (+33%), camel (+31%), mouse (+30%).

8 What We Learned

The Surprising Part

Honestly, the most eye-opening thing wasn't the accuracy improvement—it was discovering that the baseline was *this* overconfident. 54 images predicted wrong with $>$99% certainty. That's not a model making reasonable mistakes; that's a model that has learned texture shortcuts and treats them as absolute truths. Grad-CAM showed us exactly how: "road texture" $\rightarrow$ "must be a road" at 96% confidence, even when there's a castle right there in the image.

What surprised us equally was that CutMix fixed this so effectively. One augmentation trick knocked those 54 catastrophic errors down to 1, while adding 15 points of accuracy. The overfitting and overconfidence turned out to be the same problem—both caused by the model locking onto dominant textures.

The Cases That Keep Us Up at Night

The lobster$\rightarrow$crab failure (still 92% confidence after CutMix) tells us something important: **some errors can't be fixed by better training.** At 32×32, the features that separate a lobster from a crab—antenna length, tail shape, claw proportions—simply don't exist in the pixel data. You'd need higher resolution. No augmentation strategy can recover information that was never there.

The TV$\rightarrow$man case raises a different concern: what happens when the "correct" object isn't the most visually prominent one? The model classified by saliency, not by the label. That's a fundamental limitation of single-label classification, not something training tricks alone can solve.

Would We Actually Deploy This?

At 75.7% on 100 classes? Not without guardrails. But here's the thing—CutMix *makes guardrails possible*. Because wrong predictions now cluster at low confidence, a simple threshold (reject anything below 70%) becomes a viable safety net. The baseline's overconfidence made that impossible—its wrong answers looked just as confident as its right ones. We'd also want special handling for known hard pairs (lobster/crab, oak/maple) and higher-resolution

fallback for critical decisions.

The Takeaway

If there's one thing this project drove home, it's that **accuracy is only half the story**. A model that's 60% accurate but knows when it's guessing is arguably more useful than one that's 70% accurate but acts certain on everything. CutMix gave us both—higher accuracy *and* honest uncertainty. A model that knows what it doesn't know is worth a lot more than one that's always confident.

Reproducibility: Seed 42, deterministic CuDNN. Full code: `DL_Assignment01_Complete.ipynb`. Hardware: NVIDIA L40S (Institute server @172.25.0.141), PyTorch, CUDA 12.8. *Team:* b22cs055, b22ee064, p25am002, p25am003