ESE 3060 Final Project - Part 1

Ani Petrosyan, Lakshman Swaminathan

December 7, 2025

1 Hypothesis

Our goal is to reduce the wall-clock training time of `airbench94.py` on CIFAR-10 while staying as close as possible to the original 94.0% test accuracy. Our hypothesis is that (1) precomputing deterministic image transforms, (2) adding residual connections to the convolutional blocks, (3) tuning the batch size, learning rate, and number of epochs, and (4) optimizing data movement to the GPU will shorten training time by around 15–20% while keeping test accuracy within about 0.5–1.0 percentage points of the 94.0% baseline.

2 Methodology

To test this hypothesis, we started from the provided `airbench94.py` baseline and made a small set of targeted changes aimed at either (a) removing redundant computation or (b) improving convergence per unit time:

- **Static precomputation of image transforms.** Normalization, flipping, and padding are computed once in `CifarLoader.__init__` and stored in `self.proc_images`, instead of being recomputed every epoch. During training, the loader only does the remaining random crop on precomputed padded images.
- **Residual connections in `ConvGroup`.** Each `ConvGroup` now has a residual path: the pooled input is passed through a 1×1 convolution "downsample" layer and added back to the main convolutional path before the GELU activation. This is meant to improve gradient flow and let the network learn near-identity mappings more easily, so it can converge in fewer effective epochs.
- **Hyperparameter changes.** We increased the batch size from 1024 to 1536 and scaled the learning rate from 11.5 to 17.25 (1.5×, matching the batch-size change). We reduced the number of epochs from 9.9 to 8.0 to trade a small loss in optimization for faster overall runtime. These changes are intended to better use GPU memory and reduce per-epoch overhead while keeping training dynamics similar.
- **Optimized data loading and device transfers.** Images are converted to the final dtype/layout and moved to the GPU once, with non-blocking transfers and labels cast to `long` on the correct device. This removes per-batch conversion and host-to-device overhead that does not contribute to learning.

To evaluate these changes, we ran the baseline and the modified script on the same GPU and recorded (a) final CIFAR-10 test accuracy and (b) wall-clock runtime. The detailed experiment log, including commands and multiple runs, is provided in the separate appendix.

3 Results & Conclusions

In 5 runs, the baseline achieved an average of 94.01% test accuracy with a runtime of roughly 1m 54.381s (real category in the logs). Over 5 runs, the modified version ("modified" variant) ran in an average of 1m 48.074s, which is roughly an 5.3% reduction in wall-clock time, but its accuracy dropped slightly to around 92.81% (a loss of about 1.2 percentage points compared to the baseline).

These results partially support the hypothesis: the combination of precomputed transforms, residual connections, and hyperparameter / data-loading tweaks clearly reduces training time, and the accuracy remains very close to the original 94.0% target. However, the small accuracy drop suggests there is still a trade-off between speed and performance. In future iterations, we would tune the number of epochs and learning rate more finely (or add mild regularization changes) to see if we can recover the lost ∼1.2 percentage points while keeping most of the ∼5.3% speed-up.