Optional Homework
Advanced Topics in Neural Networks

Dragos Gabriel-Catalin

December 23, 2025

1 Introduction

The goal of this assignment is to replace a sequence of classical image transformations (resizing, grayscale conversion, horizontal and vertical flipping) with a neural network that performs the same transformation faster at inference time. The network receives RGB CIFAR-10 images of size $3 \times 32 \times 32$ as input and produces grayscale images of size $1 \times 28 \times 28$ as output.

The primary objective is not perfect reconstruction quality, but achieving faster inference compared to sequential CPU-based transformations.

2 Dataset and Ground Truth

The CIFAR-10 dataset was used as input. For each image, the ground truth output was generated using the following torchvision transformations:

- Resize to 28×28
- Conversion to grayscale
- Horizontal flip
- Vertical flip

These ground truth images are used as targets during training.

3 Model Architecture

A small convolutional neural network was chosen in order to keep inference fast and efficient. The model consists of two convolutional layers:

- A 3×3 convolution mapping 3 input channels to 16 feature maps, followed by ReLU
- A 3×3 convolution mapping 16 feature maps to 1 output channel

This architecture reduces the spatial resolution from 32×32 to 28×28 naturally, without requiring explicit resizing layers.

The simplicity of the model was intentional, as the task is deterministic and does not require deep or complex representations.

4 Loss Function

Mean Squared Error (MSE) loss was used for training:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|y_i - \hat{y}_i\|^2$$

This choice is appropriate because the task is an image-to-image regression problem where the goal is to approximate pixel intensities. Although MSE tends to produce slightly blurred outputs, this is acceptable given that visual fidelity is not the main objective of the assignment.

5 Training and Early Stopping

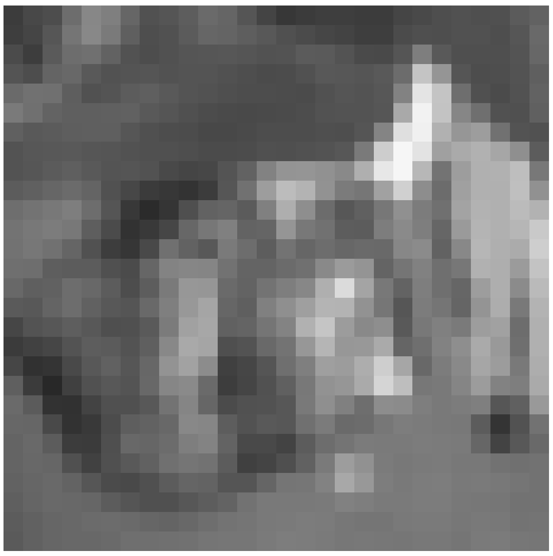
The model was trained using the Adam optimizer with a learning rate of 10^{-3}. Training was performed on CPU, although the pipeline supports GPU and MPS devices.

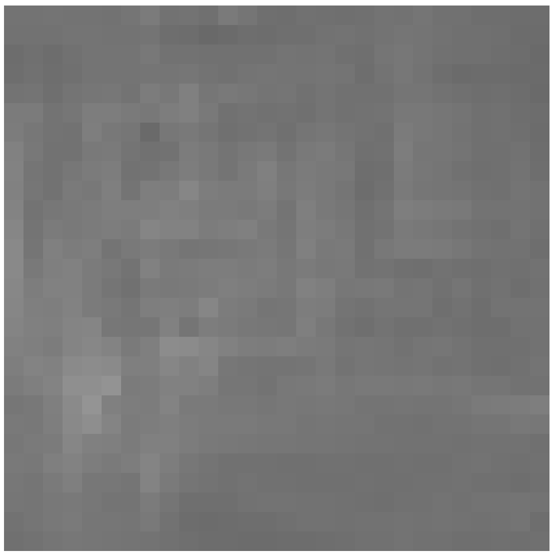
Early stopping was employed to prevent unnecessary training once the loss stopped improving. Training was stopped if the training loss did not decrease for 5 consecutive epochs. This criterion was chosen because the task is simple and overfitting is not desirable.

Training converged quickly, with the loss stabilizing around 0.05.

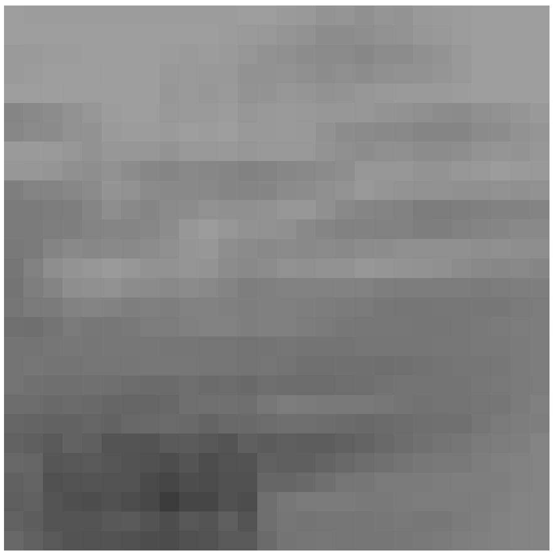
6 Qualitative Results

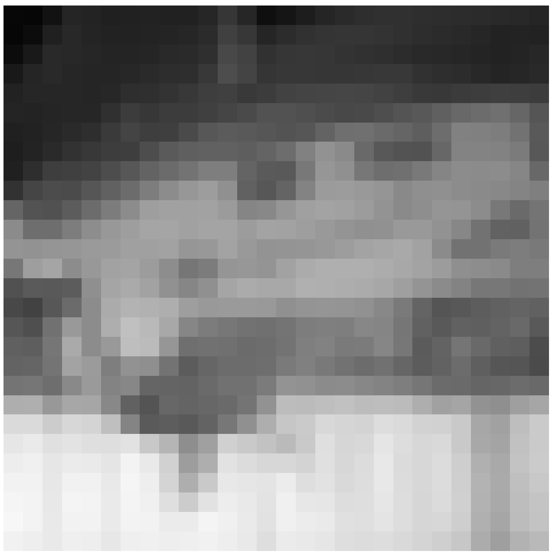
Figure 1 shows examples of images generated by the model compared to the ground truth transformations.


(a) Ground Truth


(b) Model Output

(c) Ground Truth


(d) Model Output


(e) Ground Truth

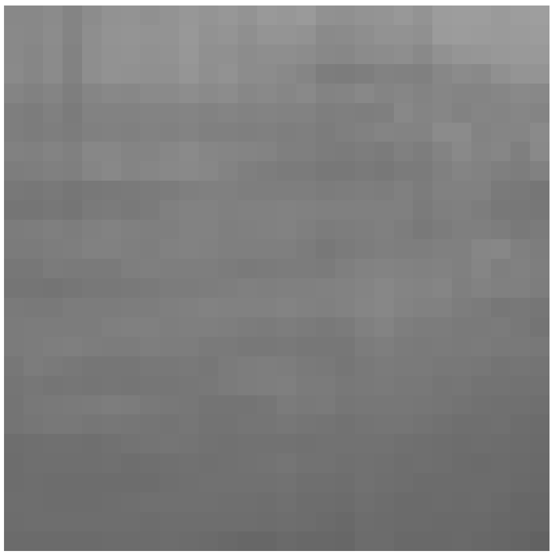
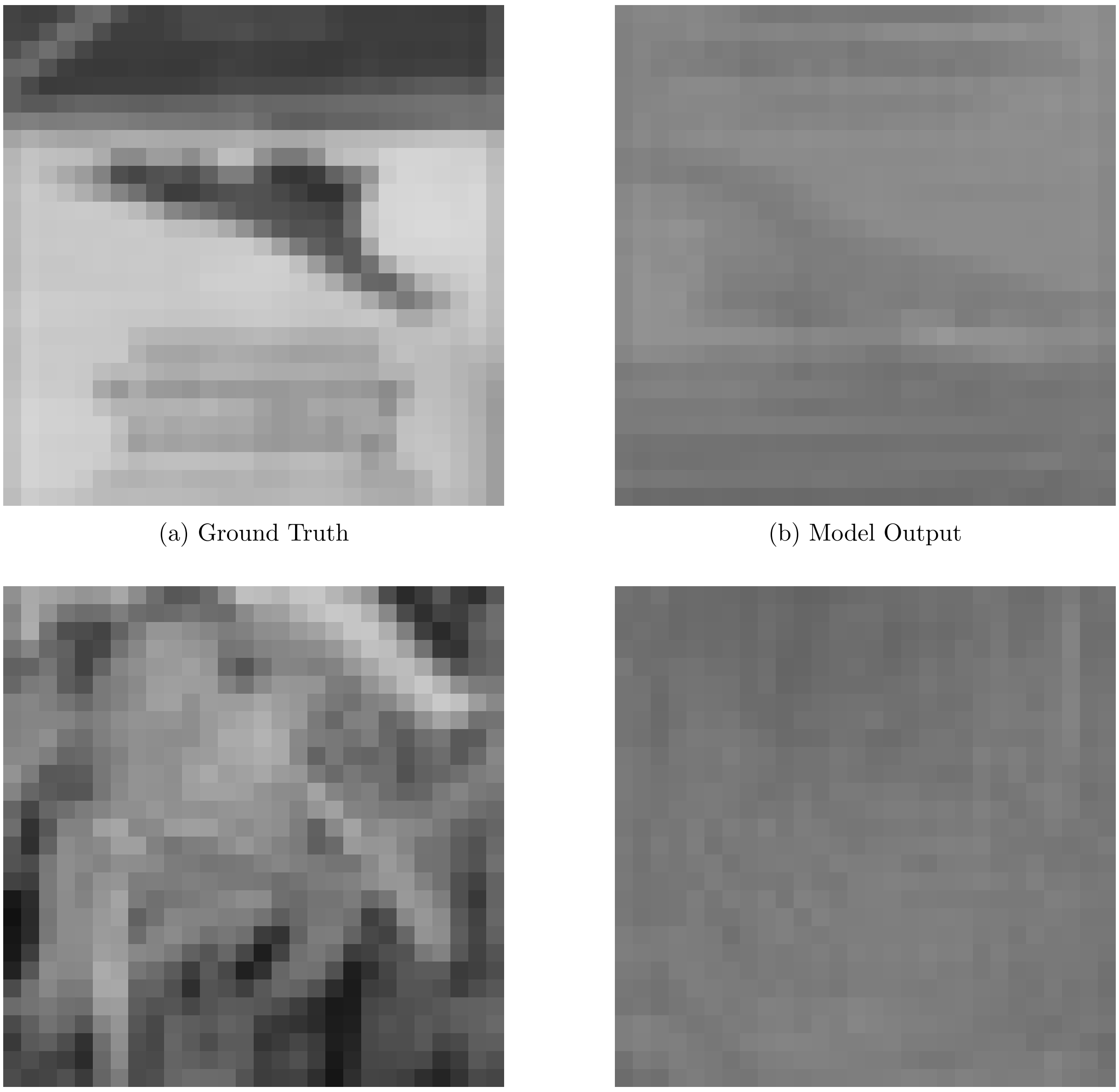

(f) Model Output

Figure 1: Comparison between ground truth images and model predictions.


(a) Ground Truth (b) Model Output

(c) Ground Truth (d) Model Output

Figure 2: Comparison between ground truth images and model predictions.

The model correctly captures the global structure and orientation of the transformed images. Fine details are smoothed due to the small model capacity and the use of MSE loss, which is acceptable for this task.

7 Inference Time Benchmark

The model's inference performance was evaluated using the `test_inference_time` function across multiple batch sizes on both CPU and GPU (CUDA). The baseline for comparison is the sequential torchvision transformations performed on the CPU.

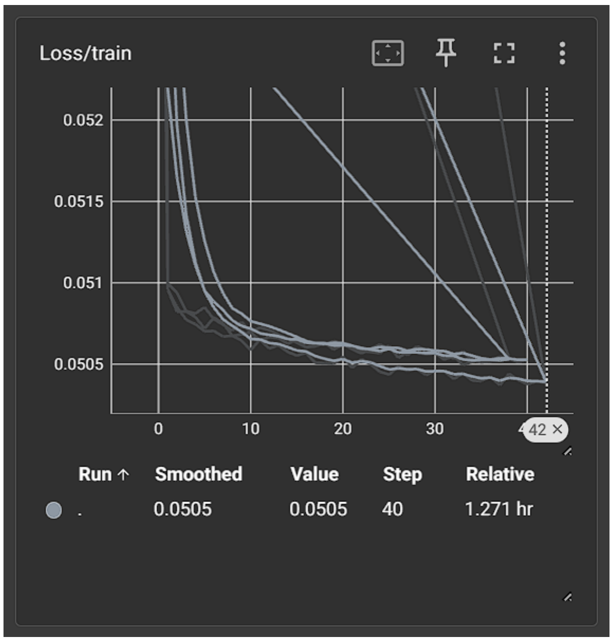
Device	Batch Size	Sequential (s)	Model (s)	Status
CPU	1	1.3212	1.6470	Slower
CPU	16	1.3212	0.3217	FASTER
CPU	64	1.3212	0.2257	FASTER
CPU	128	1.3212	0.3044	FASTER
CPU	256	1.3212	0.3533	FASTER
CUDA	1	1.3212	2.9161	Slower
CUDA	16	1.3212	0.2416	FASTER
CUDA	64	1.3212	0.0905	FASTER
CUDA	128	1.3212	0.0720	FASTER
CUDA	256	1.3212	0.0667	FASTER

Table 1: Inference time comparison on CPU and CUDA. Sequential CPU time remains the constant baseline.

7.1 Analysis of Performance and "Break-even" Points

Based on the benchmark data in Table 1, we identified the following performance characteristics:

- **Break-even Points:** On both CPU and CUDA, the model becomes faster than sequential transformations at a **batch size of 16**. For a batch size of 1, the model is slower due to the overhead of framework initialization and, in the case of CUDA, the latency of moving data from host memory to GPU VRAM.
- **CPU Scaling:** On the CPU, the model reached peak performance at a batch size of 64, achieving a speedup of approximately 5.85×.
- **CUDA Acceleration:** The GPU (CUDA) demonstrated massive parallelism benefits. At a batch size of 256, the inference time dropped to 0.0667 seconds, which corresponds to a speedup of roughly 19.8× compared to the sequential ground truth.



Run ↑	Smoothed	Value	Step	Relative
● .	0.0505	0.0505	40	1.271 hr

Figure 3: TensorBoard Loss Curve

The TensorBoard logs demonstrate successful training, with the MSE loss converging to approximately 0.0505. The rapid initial descent followed by a plateau after step 20 justifies the chosen early stopping patience, as the model reached a point of diminishing returns where the loss minimization became marginal .

These results confirm that the proposed CNN model is highly efficient for bulk image transformations, particularly when hardware acceleration is available.

8 Conclusion

A lightweight convolutional neural network was successfully trained to approximate a sequence of image transformations. Despite imperfect visual reconstruction, the model significantly outperforms sequential CPU-based transformations in terms of inference speed. This demonstrates that neural networks can be effectively used as fast approximators for deterministic image processing pipelines.