Optional Assignment: Advanced Topics in Neural Networks

Isac Lucian-Constantin

December 18, 2025

1 Model Architecture and Design

1.1 Chosen Architecture

It is used a very small model called `TransformNet`. It follows the same steps as the ground-truth pipeline:

- **Resize:** the input RGB image ($3 \times 32 \times 32$) is resized to 28×28 using bilinear interpolation.
- **Grayscale:** a single 1×1 convolution maps 3 channels to 1 channel (it learns how to combine R, G, B into grayscale).
- **Flip:** flip the image horizontally and vertically using `torch.flip`.
- **Clamp:** the output is clamped to $[0, 1]$ to keep valid pixel values.

1.2 Why This Architecture

The model is simple and fast:

- The geometric part (resize + flips) is hard-coded: it is known and always the same.
- The grayscale conversion is learned: that is the only part that can be learned as a simple mapping from RGB to one channel.
- The model is tiny, so it trains quickly on CPU.

1.3 Parameter Count and Size

The model has only one learnable layer: a 1×1 convolution from 3 channels to 1 channel. This means it has only 3 trainable weights (no bias), so it is extremely small.

2 Loss Function Choice

2.1 Selected Loss

The Mean Squared Error (MSE) is used between the model output and the ground truth:

$$\mathcal{L} = \text{MSE}(\hat{y}, y).$$

2.2 Motivation

MSE is a good choice because it is simple, stable during training, and directly measures how close the predicted pixels are to the target pixels.

3 Early Stopping Criterion

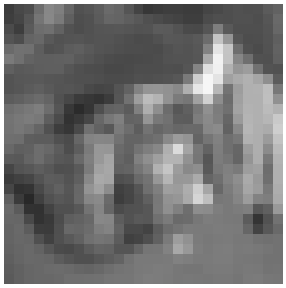
3.1 Early Stopping Criterion

- The model is saved when the validation loss improves by at least `min_delta = 1e-4`.
- If validation loss does not improve for `patience = 4` epochs, stop training.
- At the end, save the best weights.

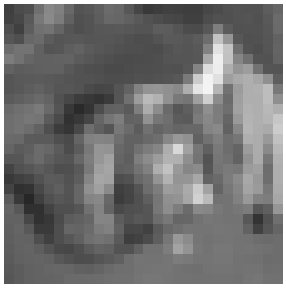
3.2 Motivation

Early stopping prevents wasting time when the model is not improving anymore. Since the model is small and converges fast, this helps finish training sooner while keeping the best validation performance.

4 Model vs. Ground Truth



(a) Ground truth #1



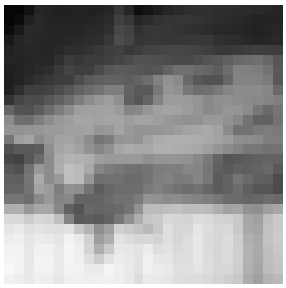
(b) Model output #1

Figure 1: Comparison example #1 (GT vs. model).

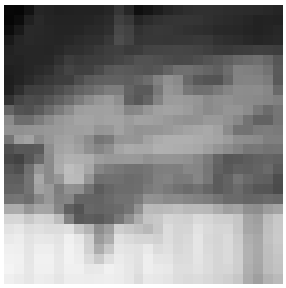
(a) Ground truth #2

(b) Model output #2

Figure 2: Comparison example #2 (GT vs. model).



(a) Ground truth #3



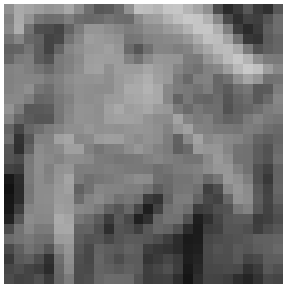
(b) Model output #3

Figure 3: Comparison example #3 (GT vs. model).

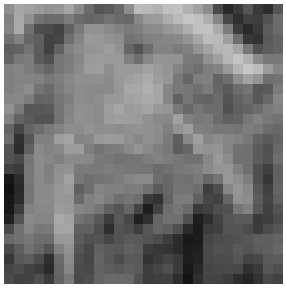
(a) Ground truth #4

(b) Model output #4

Figure 4: Comparison example #4 (GT vs. model).



(a) Ground truth #5



(b) Model output #5

Figure 5: Comparison example #5 (GT vs. model).

5 Runtime Benchmarking

5.1 Benchmark Setup

Measured the time needed to apply the ground-truth pipeline (resize + grayscale + hflip + vflip) sequentially on CPU over the entire test set and for the model, it was set in evaluation mode and measured the forward-pass time over the same test set using a `DataLoader`. Test parameters:

$$\texttt{batch\_size} \in \{16, 32, 64, 128, 256, 512\}.$$

$$\texttt{device} \in \{\texttt{cpu}, \texttt{cuda}\},$$

and also evaluated the impact of `shuffle` (`True`/`False`) and `drop_last` (`True`/`False`). The `num_workers` is set to 0 in all runs since it seems that increasing it results in worse outcomes.
Sequential: applying the ground-truth transformations sequentially on CPU over the test set took **0.7399 s**.

Device	Batch size	Shuffle	Drop last	Time (s)
cuda	16	True	True	0.4121
cuda	32	True	True	0.1712
cuda	64	True	True	0.1000
cuda	128	True	True	0.0711
cuda	256	True	True	0.0666
cuda	512	True	True	0.0597
cuda	16	True	False	0.2652
cuda	32	True	False	0.1332
cuda	64	True	False	0.1186
cuda	128	True	False	0.0692
cuda	256	True	False	0.0538
cuda	512	True	False	**0.0503**
cuda	16	False	True	0.2528
cuda	32	False	True	0.1517
cuda	64	False	True	0.1036
cuda	128	False	True	0.0887
cuda	256	False	True	0.0785
cuda	512	False	True	0.0504
cuda	16	False	False	0.2913
cuda	32	False	False	0.1337
cuda	64	False	False	0.0923
cuda	128	False	False	0.0966
cuda	256	False	False	0.0752
cuda	512	False	False	0.0681
cpu	16	True	True	0.3416
cpu	32	True	True	0.2136
cpu	64	True	True	0.1371
cpu	128	True	True	0.1181
cpu	256	True	True	0.0872
cpu	512	True	True	0.0739
cpu	16	True	False	0.2935
cpu	32	True	False	0.1903
cpu	64	True	False	0.1561
cpu	128	True	False	0.1157
cpu	256	True	False	0.0796
cpu	512	True	False	0.0869
cpu	16	False	True	0.2909
cpu	32	False	True	0.1823
cpu	64	False	True	0.1393
cpu	128	False	True	0.0977
cpu	256	False	True	0.1142
cpu	512	False	True	0.0864
cpu	16	False	False	0.3691
cpu	32	False	False	0.2315
cpu	64	False	False	0.1688
cpu	128	False	False	0.1110
cpu	256	False	False	0.1091
cpu	512	False	False	**0.0723**

Table 1: Inference-time benchmark for `TransformNet` using `test_inference_time` with varying device, batch size, `shuffle`, and `drop_last`. The sequential CPU baseline for ground-truth transforms is 0.7399 s.

6 Training Details

Trained `TransformNet` on CIFAR-10 training images using the provided `CustomDataset`. Each input is the original RGB image ($3 \times 32 \times 32$), and the target is produced by the ground-truth pipeline (resize to 28×28, convert to grayscale, then horizontal and vertical flip).

The training set was split into **90% training** and **10% validation**. Training was done with **AdamW** optimizer - learning rate $2 \cdot 10^{-3}$, batch size **256**, for a maximum of **30** epochs. The loss function was **MSE**.

Used a **device** parameter so the same training code can run on CPU or GPU (`cuda` if available, otherwise `cpu`). During training the epoch time, training MSE, and validation MSE were logged using TensorBoard.

7 Experiment Tracking (TensorBoard)

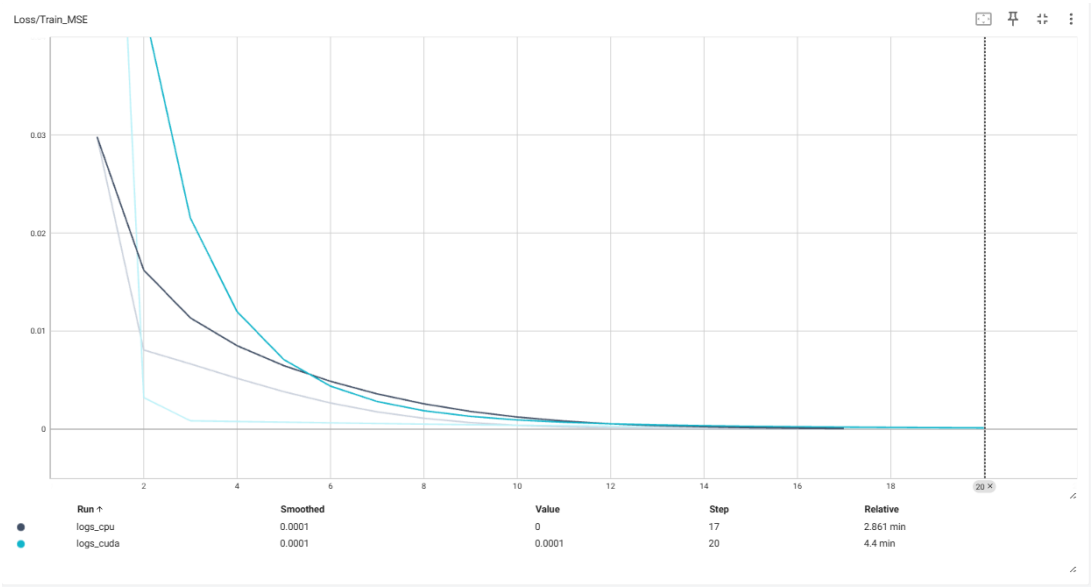


Figure 6: Comparison of training loss on `CPU` and `CUDA`

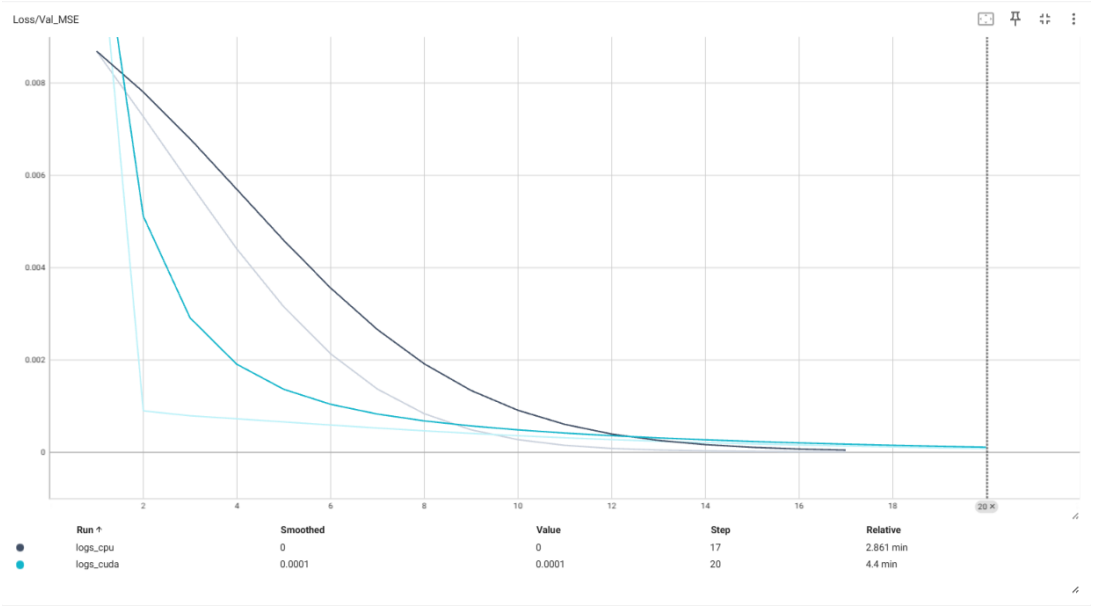


Figure 7: Comparison of validation loss on `CPU` and `CUDA`

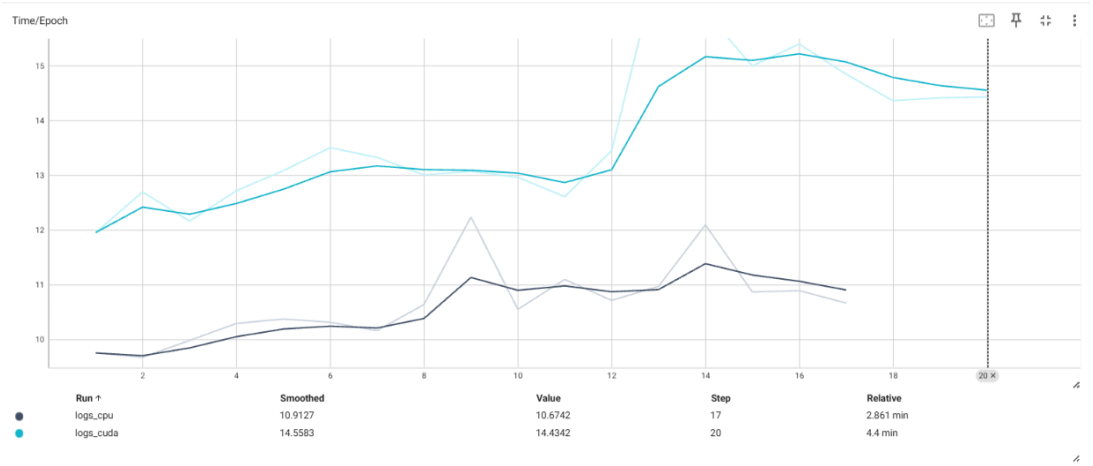


Figure 8: Comparison of elapsed time on `CPU` and `CUDA`