ATNN - Optional Homework - Report

Marin Andrei-Vasile

December 24, 2025

1 Approach and Model Design

The goal is to learn a fast approximation of a horizontal and vertical flip combined with an RGB-to-grayscale conversion and a spatial size reduction from 32×32 to 28×28. Since the task is deterministic and simple, the model was intentionally kept minimal to favor inference speed over representational power.

1.1 Explored Model Variants

Several architectural ideas were explored before settling on the final model:

- **Resolution reduction with convolutions:** Convolutional layers with kernel sizes 3 (two stacked layers) and 5 (single layer) were tested to reduce the image resolution from 32×32 to 28×28. These layers were able to learn the resizing operation, but models using these layers failed to properly predict the pixel values on the edges of the image.
- **Channel reduction with 1×1 convolution:** A `Conv2d` layer with kernel size 1 was introduced to reduce the input from 3 channels to 1. This successfully learned an RGB-to-grayscale conversion and significantly reduced the computational cost of subsequent layers.
- **Fully linear mapping:** A model consisting of a single large linear layer mapping the flattened $3 \times 32 \times 32$ input directly to the flattened $1 \times 28 \times 28$ output was evaluated. While this approach worked functionally, it was computationally expensive and offered no practical speed advantage.
- **Output value constraints:** A sigmoid activation was tested at the output to enforce values in $[0, 1]$, but it introduced visible noise and unnecessary computation. Clamping the output values to $[0, 1]$ was also evaluated; while effective, it similarly added avoidable overhead.

Flattening and unflattening operations were implicitly used in all models involving linear layers.

1.2 Final Model

Based on these experiments, the final architecture was chosen for its simplicity and speed:

- A single `Conv2d` layer with kernel size 1 mapping 3 input channels to 1 output channel.
- A flattening step followed by a single fully connected layer mapping the flattened 32×32 grayscale image to a flattened 28×28 output.
- An unflattening step to produce a $1 \times 28 \times 28$ output image.

This design avoids explicit geometric operations and instead learns the transformation as a linear mapping, resulting in minimal inference overhead. The convolutional layer learns the RGB-to-grayscale conversion, while the linear layer learns the resizing operation and the horizontal and vertical flips.

2 Loss Function

Mean Squared Error (MSE) loss was used, as the task is a pixel-wise regression problem. Since pixel values are normalized to $[0, 1]$, the per-pixel error is small. Therefore, the reduction was set to `sum` instead of `mean` to provide a stronger training signal.

To discourage predictions outside the valid range, errors for pixels with values below 0 or above 1 were weighted by a factor of 2. Higher weights were avoided as they led to unstable training.

3 Early Stopping Criterion

Early stopping was applied based on validation performance. Training was stopped when the average (over the validation dataset) sum of absolute pixel errors per image dropped below half the intensity interval between two consecutive pixel values, multiplied by the number of pixels in an image. This criterion ensures that, on average, most pixels are mapped to the correct discrete intensity level, which is sufficient for a visually correct result.

The target validation error is given by the following formula: $0.5/255 \times (28 \times 28)$. This target can be lowered further to get better predictions.

4 Qualitative Results

The report includes six representative examples in Figure 1. Each visualization is organized into four columns: input image, ground truth output, model prediction, and absolute difference. To improve interpretability, the absolute difference image is normalized by the maximum absolute error per image, mapping zero error to black and maximum error to white.

The number of visible gray levels reflects the magnitude of the maximum error. Overall, the predictions closely match the ground truth, with only minor pixel-level deviations.

5 Inference Speed and Benchmarking

Due to the simplicity of the architecture (one convolutional and one linear layer), the model achieves very fast inference. With batch size 1 on CPU, the model is already on par with or slightly faster than applying the ground truth transformations sequentially.

A Weights & Biases hyperparameter sweep was conducted to find the fastest inference configuration on both CPU and GPU. The optimal setup for both devices used a batch size of 64, zero data loader workers (to avoid worker creation overhead), and `pin_memory=True`. Under these conditions, the original CPU transformations took approximately 2.3-2.5 seconds, while the trained model required about 0.45-0.5 seconds on CPU and 0.16-0.35 seconds on GPU, clearly outperforming the direct transformations.

The full results can be found in Figures 2 and 3.

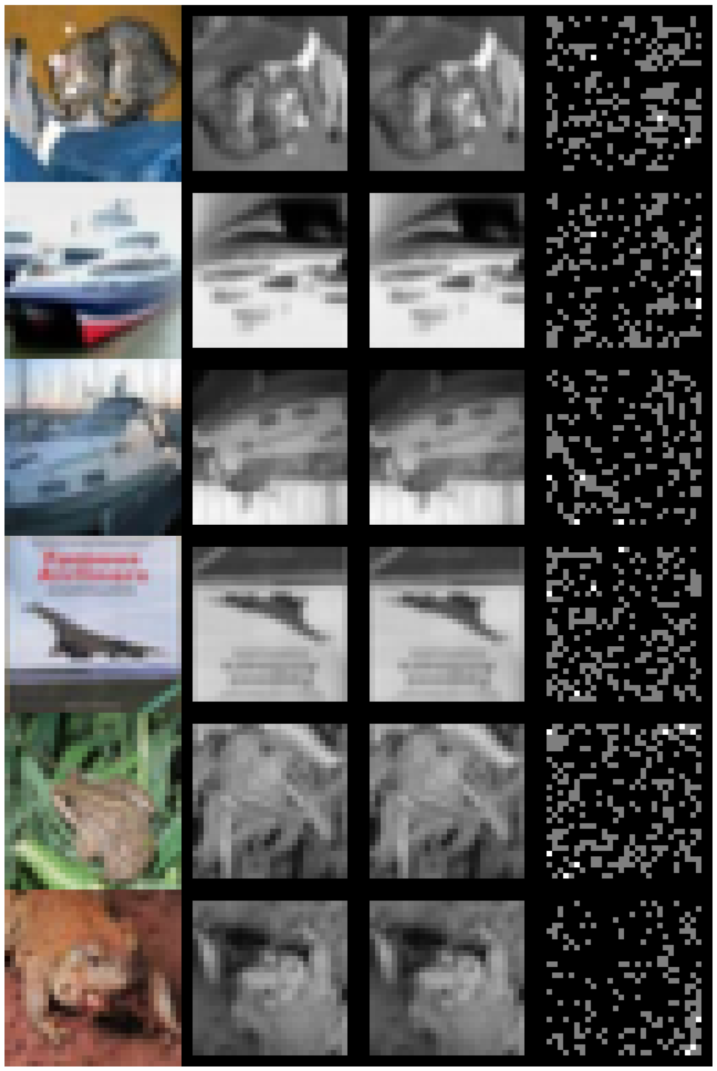
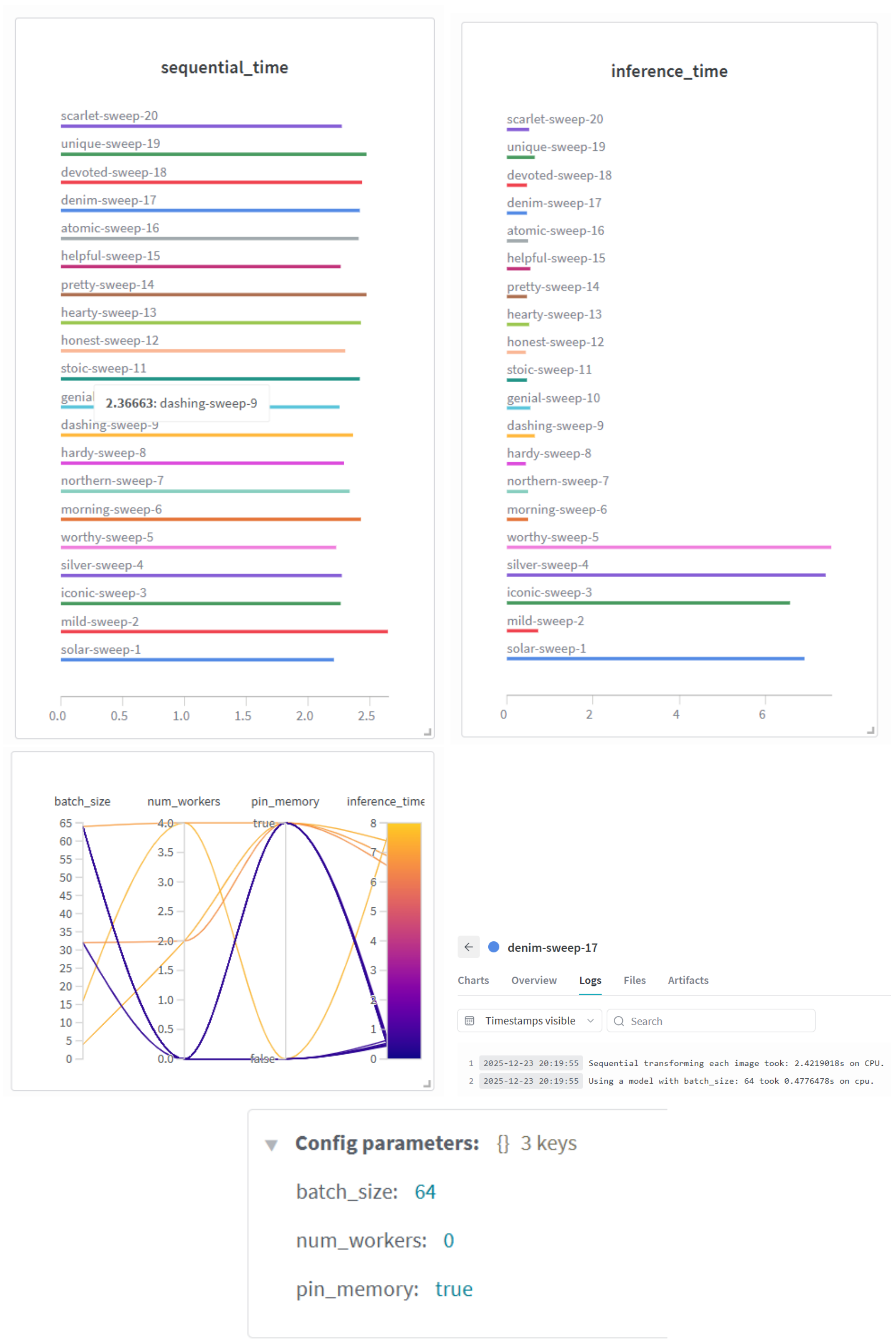


Figure 1: Model prediction comparison (input image, ground truth output, model prediction, absolute difference)

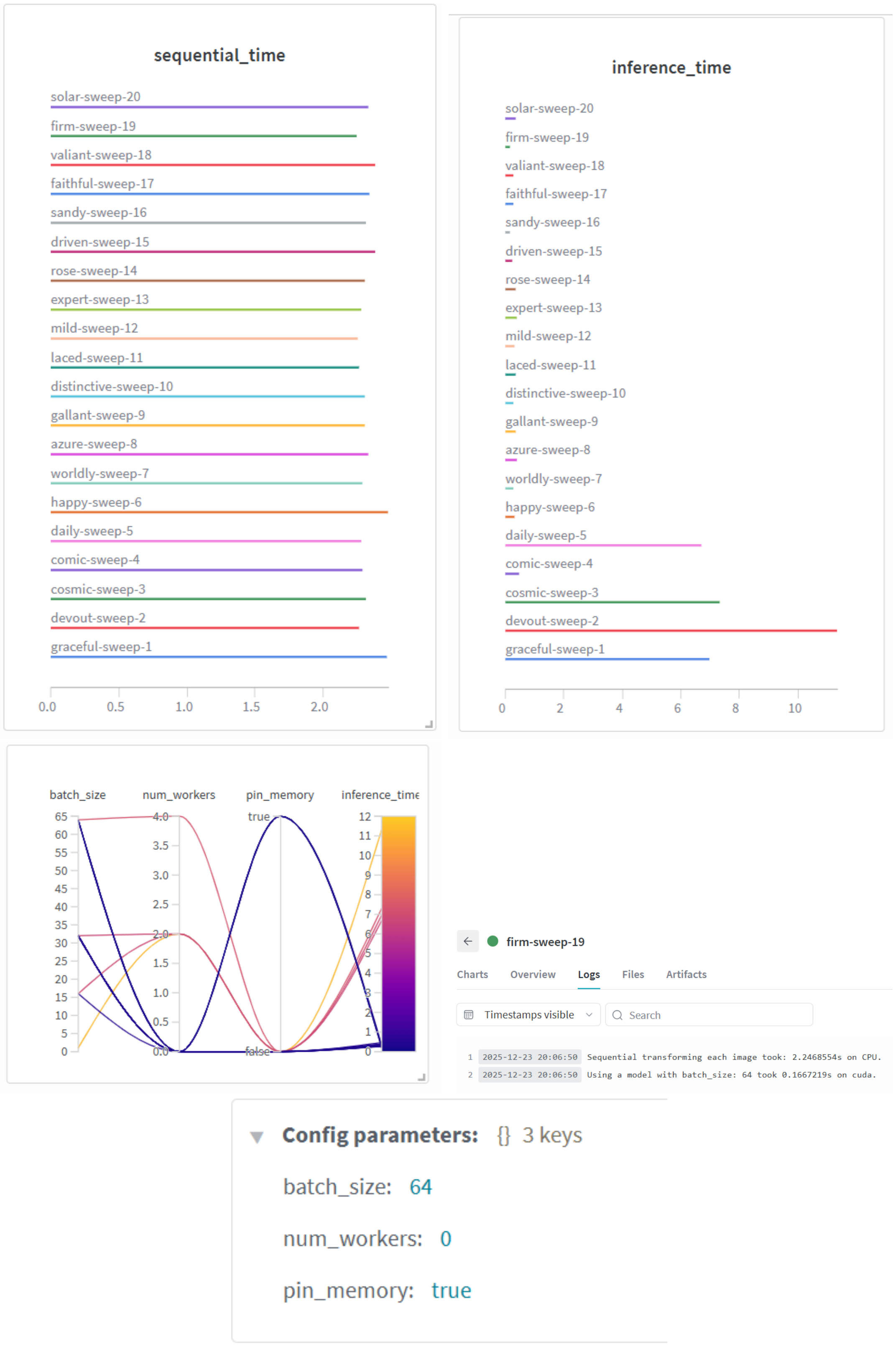


```
2025-12-23 20:19:55 Sequential transforming each image took: 2.4219018s on CPU.
2025-12-23 20:19:55 Using a model with batch_size: 64 took 0.4776478s on cpu.
```

Config parameters: {} 3 keys

batch_size:	64
num_workers:	0
pin_memory:	true

Figure 2: CPU hyperparameter sweep logs



```
1  2025-12-23 20:06:50  Sequential transforming each image took: 2.2468554s on CPU.
2  2025-12-23 20:06:50  Using a model with batch_size: 64 took 0.1667219s on cuda.
```

Config parameters: {} 3 keys

batch_size: 64

num_workers: 0

pin_memory: true

Figure 3: GPU hyperparameter sweep logs