COS526 Assignment 2 Report

Jiatong Yu

Implementation

In this assignment, we partially implemented the model architecture, rendering, and the positional embedding.

For model architecture, we implemented the equivalence of the original paper, where skip connections are supported. In the default setting, we implemented 8 identical MLP layers that is supposed to be learning the latent representation of the radiance field. Then three additional layers are added to support feature mapping, predict volume density, and RGB color.

The positional encoder module also strictly follows the paper's describtion, where we have $\sin$ and $\cos$ functions that map the absolute position into a higher dimension feature, as shown by effective by most contemporary architectures. It was a challenging part to reverse-engineer what we wanted for log sampling, since the paper does not go into details about it.

The rendering part takes model's output and transform it into an actual rendering, where we considered weight, RGB-color, disparsity, etc. for a successful rendering. During the implementation, I was interested in seeing how adding the random noise before $\alpha-$compositing influence the training result. Due to high training cost, we were not able to run a comparison, but it would be a natural future work for us to explore.

Results

First we use the LEGO and FERN rendering task to show training dynamics. The images below are selected from the same angle for training iterations at $10,000$, $35,000$, and $100,000$. As shown in Figure 1, for the LEGO rendering task, the rendering quality improved significantly from iteration 10000 to iteration 35000, while later steps iteratively refine and output rendering with higher clarity. In Figure 2, we also selected some views from the last iteration rendering.

Figure 1: **Training Checkpoints.** Generated LEGO image at 10000 iteration (left), 35000 iterations (middle), and 100000 iterations (right).

Figure 2: **Final Iteration Multivew.** Generated LEGO images at the final iteration, with multiple views.

The FERN rendering task behaves very similarly to the LEGO task, despite taking a longer number of iterations before a satisfactory result is rendered. As in LEGO's Figure 1, iteration 35000 is of acceptable quality. For FERN, a similar quality image occurs around 8000 iterations, and further iterations build on its clarity. We report the FERN training outputs in Figure 3 and Figure 4.

Figure 3: **Training Checkpoints.** Generated FERN image at 10000 iteration (left), 75000 iterations (middle), and 175000 iterations (right).

Figure 4: **Final Iteration Multivew.** Generated FERN images at the final iteration, with multiple views.