Neural Radiance Fields for Semantic Segmentation

Outline

Phuoc Thang Le

26.01.2023

Abstract

The early days of deep learning saw significant progress in traditional 2D image recognition tasks such as classification, detection, and instance segmentation. As the field has matured, research in deep learning-based computer vision has turned towards tackling fundamental 3D computer vision problems, including synthesizing new views of an object and reconstructing its 3D shape from images. Many approaches to these problems have taken a traditional machine learning approach, attempting to reconstruct 3D geometry from images through a set of training iterations. However, a new approach called Neural Radiance Fields (NeRF) has recently been introduced and has gained attention in the field. This work will cover the basic concepts of the original NeRF proposal [2] and attempt to use the modified architecture to solve the problem of 3D semantic segmentation.

1 Introduction

1.1 What is NeRF?

Neural radiance fields (NeRF), proposed by Mildenhall et al. [2] are neural networks that can generate realistic 3D views of a scene based on a set of 2D images. NeRF are trained to reconstruct the input images using a rendering loss function, by interpolating between the images to create a single, complete scene. The NeRF network maps the 5D input of viewing direction and spatial location to the 4D output of opacity and color, using volume rendering to generate new views. While NeRF can be computationally intensive, particularly for complex scenes, there have been recent developments that significantly improve its performance.

The architecture of NeRF consists of a series of fully-connected layers that take the 5D input and process it through multiple steps to produce the 4D output. The

network is trained using a rendering loss, which measures the difference between the generated views and the input views, and adjusts the network weights to minimize this difference.

1.2 How does NeRF work?

1.2.1 Input and Output

Neural Radiance Fields (NeRF) are designed to generate novel views of complex 3D scenes based on a small set of input views. The network optimizes a continuous volumetric scene function using these input views and is able to produce new views of the scene as a result. The input for NeRF can be a set of static images. Its input is a 3D location $l = (x; y; z)$ and 2D viewing direction $(\theta; \phi)$ and the ouput is an emitted color $c = (r; g; b)$ and volume density (α). The whole set-up can be depicted in Figure 1.

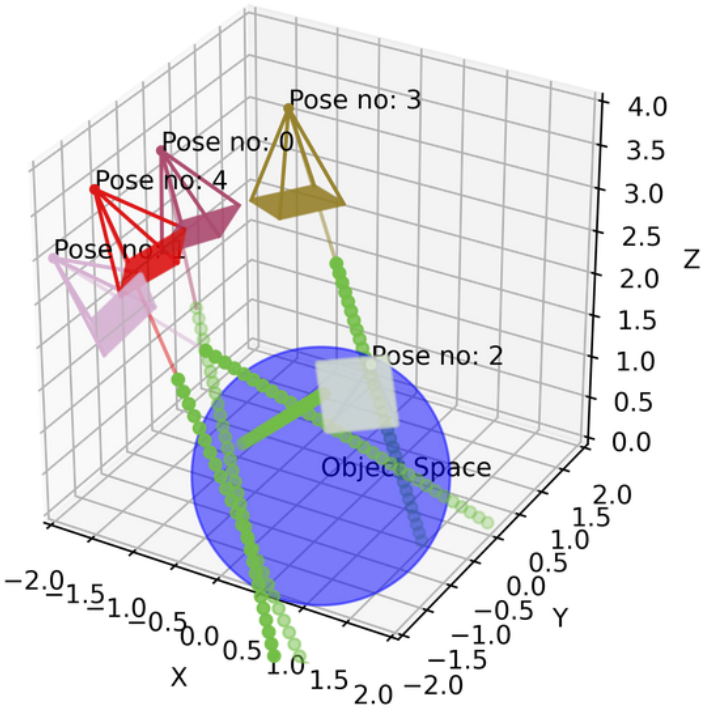

Figure 1: The blue object to be rendered is located at (0,0,0) and the camera is positioned at different locations. The distance between the camera and the canvas (the 2D image) is known as the focal length. By using each camera position and ray direction point, we can draw a ray vector and sample along it (green).[1]

1.2.2 Positional encoding

According to the empirical research conducted by Mildenhall et al., it has been demonstrated that the networks exhibit a bias towards learning low frequencies, resulting in poor rendering performance at high-frequency variations in color and geometry as demonstrated in Figure 2. Previous research has shown that representing inputs in a higher-dimensional space can enhance a neural network's capacity to learn intricate functions [3].

Positional encoding is a technique that uses high frequency functions to encode input data and map it to a higher-dimensional space. In the context of Neural Radiance Fields (NeRF), positional encoding is applied to both the location coordinates and view directions of the input data before it is input into the multi-layer perceptron (MLP) network. For example, a scalar position $x \in$ will be encoded in multiresolution sequence of $L \in$ sine and cosine functions.

- $enc(x) = (\sin(2^0 x), \sin(2^1 x), \sin(2^2 x), ..., \sin(2^{L-1} x),$
  $\cos(2^0 x), \cos(2^1 x), \cos(2^2 x), ..., \cos(2^{L-1} x))$

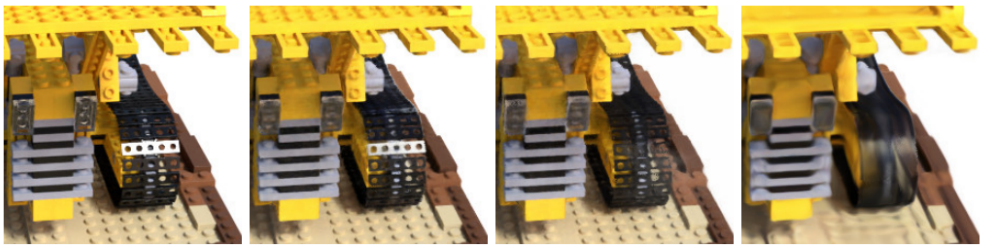

Figure 2: From the right: 1) The image that we desire to generate 2) The rendered image with positional encoding 3) Disregard this image; not in the scope of this writeup 4) The rendered image without the positional encoding. The absence of positional encoding results in bleak, over-smoothed rendered images. [2]

1.2.3 View synthesis and image-based rendering

Based on the given pose from the input, we determined the "rays" from the camera, through every pixel of the image plane. Every ray is described by two vectors:

- v_o, a vector that specifies the origin of the ray. Note that $v_o = (x_o, y_o, z_o)$ with (x_o, y_o, z_o) is the fixed position of the camera
- v_d, a normalized vector that specifies the direction of the ray.

From these rays, we sample uniformly a bundle of query points a long the rays. The output would be a set of 3D query points $\{x_p, y_p, z_p\}$ with shape $Height \times Width \times N \times 3$, with N is the number of sampling points. These points will be put in small batches and given to NeRF. Then for each point, NeRF will give a 2-dimensional output consisting of a color value and density.

2 Research Questions

In this work, I will attempt to modify NeRF to solve the task of semantic segmentation. The input is a 3D location $l = (x; y; z)$ and 2D viewing direction $(\theta; \phi)$ and the output is an binary value $g \in \{0, 1\}$ with 1 encodes the existence of the object and 0 otherwise.

Further, we will explore how the size of the inputs affect the performance of the network.

3 Approach

3.1 Data set

The dataset utilized for the experiment consists of 100 binary masks of a cello, each measuring 50px by 50px, accompanied by their corresponding transformation matrices. These matrices will later be utilized to compute the pose and focal length of the camera. A sample of 25 pictures of the masks are show in Figure 3.

3.2 Model Architecture

In order to transform the model's purpose from performing input data to RGB space to becoming a segmentation model, modifications were made to the way the model renders the volume and in the number of output of the neural radiance field.

3.2.1 Changing the rendering method

Initially, the outputs of NeRF were two-dimensional, with one dimension indicating color and the other indicating density. However, given that the task at hand is a segmentation task, a single output value would suffice to predict the mask as either 1 or 0 in a specific voxel. As a result, it is possible to modify NeRF to produce only a single output at the final layer.

However, a problem arises with this approach. In traditional volume rendering methods, density and color values are required to be given separately. In an attempt to overcome this issue, the color values were set to be uniformly as one, allowing the network to only learn the density. Under this approach, if the density is close to 0, the color result should be 0, otherwise it should be 1. However, this method did not result in any success, as the network was unable to learn any useful patterns, even after 5000 epochs.

Therefore, an alternative, simpler rendering method is proposed in which the maximum value of the array is chosen. Specifically, the function tf.reduce_max is implemented, which is a differentiable operation and TensorFlow automatically tracks the gradients through it when computing gradients with respect to the model's parameters during training.

Figure 3: examples of 25 masks in the input

```
gray_map = tf.reduce_max(sigma_a, axis = -1)
```

3.2.2 Changing the number of output channel in NeRF

An alternative, more intuitive approach is to maintain the rendering method and treat the binary mask as a RGB image. By doing so, the network will process the image as if it were a RGB image, and the gray value can be obtained by averaging the RGB values. The result is shown in Figure 5.

By the same logic, it is possible to utilize multiple channels for color in the images and average them later to obtain the gray values. These values are retained as placeholders for the network to learn about the color characteristics in the images. At a minimum, NeRF can be run with two outputs, one for the gray value and one for the density. Through experimentation, it was found that NeRF with different numbers of color channels reached the same average peak signal-to-noise ratio (PSNR) after 3000 epochs. NeRF with 100 channels performs slowly at the

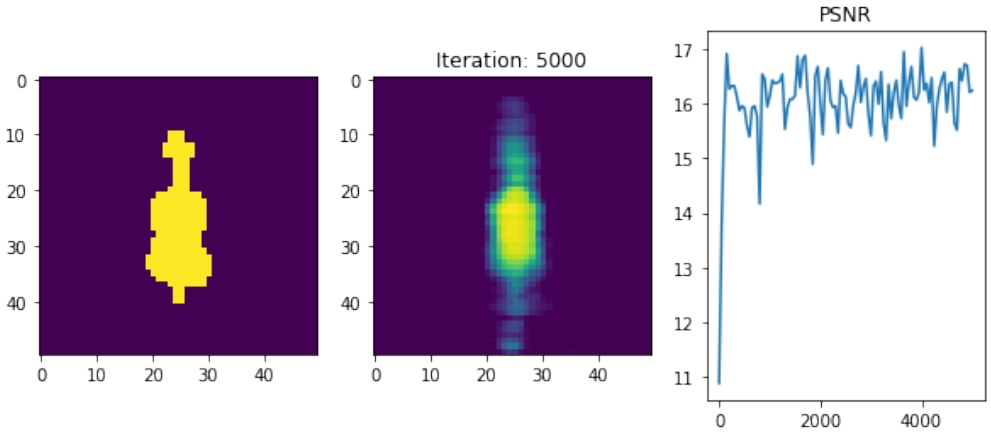

Figure 4: Result after 5000 iteration with 1 channel NeRF

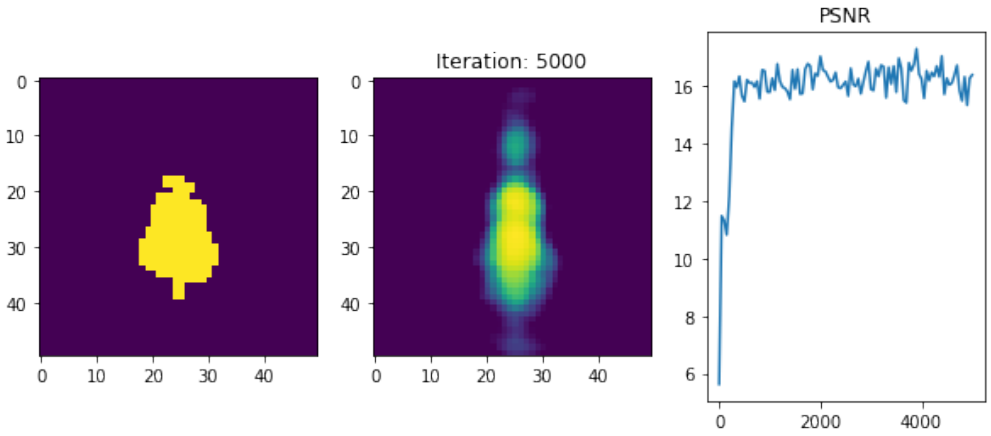

Figure 5: Result after 5000 iteration with 4 channels NeRF

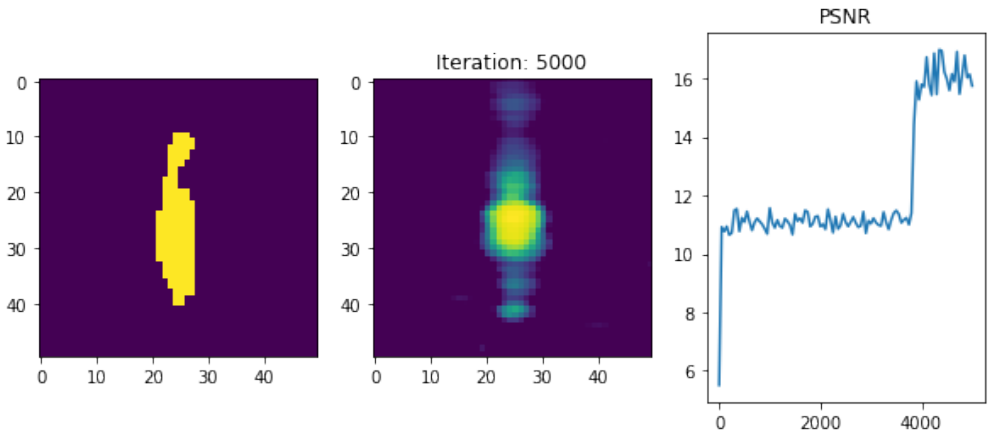

Figure 6: Result after 5000 iteration with 100 channels NeRF

beginning but jump fast after a certain epochs as in Figure 6

```
channel_map = tf.reduce_sum(weights[...,None] * channel_map_raw, -2)
gray_map = tf.reduce_mean(channel_map, -1) #gray map is the average of
    all other channel values
```

4 Conclusion

In this study, the fundamental principles of the NeRF architecture are presented. The objective of the research, which is to develop a NeRF-based model that can be applied to the segmentation task, has been successfully accomplished.

References

[1] Deep Dive into NeRF (Neural Radiance Fields), howpublished = `https://dtransposed.github.io/blog/2022/08/06/nerf/`, note = Accessed: 2023-01-26.

[2] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[3] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.