Representing Scenes as Neural Radiance Fields for View Synthesis

Presenter: Adarsh S Menon 2017ME10563

Paper: Mildenhall et al., Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020.

Report by: Abhay Pratap Singh Rathore, 2019CS50414

This paper addresses the long-standing problem of view synthesis with a novel approach of using convolution-free deep neural networks. The view synthesis problem is the construction of a view just from a set of images as inputs. The paper presents a new scene representation that is claimed to synthesise high-resolution photorealistic images. This new representation is memory efficient and suitable for optimisation directly from images.

At the time when this paper was published, there was some work done on representing the 3D scene using weights of a neural network. However these representations didn't turn out very successful in representing realistic scenes with the same fidelity as triangle meshes or voxel grids. Works were done on representing 3D scenes using signed distance fields or level sets. This representation is then followed by a differentiable rendering pipeline for optimization. These techniques can represent highly detailed scenes but were limited to simple shapes and low geometric complexity. The paper also mentioned about mesh-based scene representation. These representations are limited due to various reasons like, fixed topology of base mesh and hard-to-optimize gradient descent due to local minima. Another class of method was to use volumetric representations like Voxels. Though these representations worked very well in the past, the memory requirement of high-resolution scenes is very high. The NeRF representation solves this problem by capturing the continuous resolution of the scene by representing it using the parameters of the neural network.

Scene Representation: NeRF represents a static scene as a continuous 5D function that outputs radiance (color and brightness) for each direction (θ, φ) at each point (x, y, z) in space, along with a density value that controls how much light is absorbed by a ray passing through that point. The paper suggests that the representation should maintain consistency across multiple views by limiting the network's prediction of volume density σ to location x only, while allowing the prediction of RGB color c to consider both location and viewing direction.

Rendering Process: To render a scene from a specific viewpoint, NeRF follows a three-step process:

1. It traces camera rays through the scene, generating a set of 3D sample points.

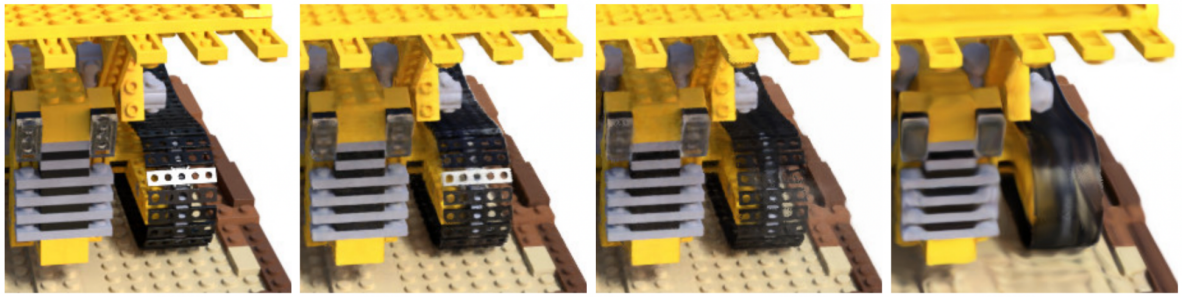
2. These points and their corresponding 2D viewing directions are fed into the neural network to produce colors and densities for each point.
3. Classical volume rendering techniques are then used to combine these colors and densities into a 2D image. They used the density parameter to calculate the alpha of the discretized segment using an (1 - exp(-sigma)). And the colors of each of these sample points were then combined using these alphas.

Optimization: NeRF is trained using gradient descent to minimize the error between observed images and the views rendered by the network.

Handling High-Resolution Scenes: NeRF employs positional encoding to address convergence issues and enable the representation of higher-frequency functions for complex scenes. Although neural networks can represent arbitrary functions, directly using 5D input is insufficient to represent high frequency scenes. Based on recent research by Rahaman et al. [35], it seems that deep networks have a tendency to learn lower frequency functions. They discovered that by mapping the inputs to a higher dimensional space using high-frequency functions before passing them to the network, better fitting of data that contains high-frequency variation can be achieved. The paper uses the following positional encoding,

$$\gamma(p) = \left(\sin\left(2^0\pi p\right), \cos\left(2^0\pi p\right), \cdots, \sin\left(2^{L-1}\pi p\right), \cos\left(2^{L-1}\pi p\right)\right)$$

Where gamma is applied to each coordinate of the 5D input.



Ground Truth | Complete Model | No View Dependence | No Positional Encoding

The above are the results that show the importance of positional encoding in the view dependence component of the NeRF.

Hierarchical volume sampling: Currently uniformly sampling N points randomly leads to unnecessary sampling of points in the free space. The solution for this is done by training two networks (coarse and fine) instead of one. The procedure is to sample Nc points and calculate the weight wi on the ray for each point such that the color Cc of the ray is Cc = SUM{Ci * wi} (using `coarse` network). These weights are proportional to the opacity of the point. Normalizing this weight gives a pdf function along the ray. A new Nf number of points are sampled on the same ray using inverse sampling on this pdf. Nf and Nc points are then taken and used to calculate the final color using the `fine` network. This idea is somewhat similar to importance

sampling. A separate network is trained for each scene. The main required data is the input images and the camera parameters. The loss function is,

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

Which is just a sum of L2 error for coarse and fine networks on the sampled points.

Advantages: NeRF can represent complex real-world scenes and objects accurately. By storing the scene in a neural network's parameters, it overcomes the storage limitations of voxel grids. NeRF outperforms existing view synthesis methods quantitatively and qualitatively, including those using neural 3D representations and deep convolutional networks. The paper shows many results to compare the contributions of NeRF compared to the existing models for scene representation. They presented results on both synthetic and real images.

Summary of View Synthesis Model Comparisons:

Tested Methods: The paper evaluates Neural Volumes (NV), Scene Representation Networks (SRN), and Local Light Field Fusion (LLFF) for view synthesis.

Training Approach: NV and SRN train separate networks for each scene, while LLFF uses a single network trained on a diverse dataset.

Performance: NeRF consistently outperforms NV and SRN in all scenarios, and it surpasses LLFF in most metrics, despite using LLFF's input images for training.

Limitations of Other Methods: SRN tends to produce smoothed geometry and texture, limiting its ability for view synthesis by selecting only a single depth and color per camera ray. NV, while capturing detailed volumetric information, struggles to represent fine details at high resolutions due to its use of a large 128^3 voxel grid. LLFF encounters difficulties when there is a significant disparity between input views, leading to incorrect geometry estimations and perceptual inconsistencies.

Trade-offs: A significant practical trade-off between these methods is the trade-off between time and space. Single-scene methods like NV, SRN, and the proposed method take longer to train (at least 12 hours per scene), while LLFF offers fast processing (under 10 minutes). However, LLFF's storage requirements are substantial due to its large 3D voxel grids, whereas the proposed method is highly memory-efficient, requiring only 5 MB for network weights, which is even less memory than the input images alone for a single scene.

Discussion in class

There were some discussions after the presentation.

Some questions were raised on the use of NeRF if the training time is very long and no editing can be done. Since the release of this original NeRF model, there has been a large progress in the area of NeRFs. Mainly, the time to train a NeRF has been reduced to a few minutes. I cannot find the exact research papers for this. But, I asked one of my friends who is working with NeRF in his project. He told me that it generally takes less than 10 minutes to train a NeRF model on a new scene using approximately 10 to 20 images.

Though there were discussions that NeRF models cannot be deformed, I found a paper, https://arxiv.org/abs/2205.04978, online, which claims the deformation of NeRF models. There was also some discussion on Drag GAN and if something similar can be done on NeRFs. Another question about whether camera translation can be performed in NeRF. For example, a castle can be modelled using NeRF, allowing users to freely roam around. Personally, I think this might be possible if we add more parameters to the input of the model. We can add the whole camera parameters like camera position, look at, up direction etc. This will need more input images to train the model.