HashNeRF: Accelerating NeRF Training with Multiresolution Hash Encoding

Ning Hsia
Columbia University
116th and Broadway, New York, NY 10027
nh2789@columbia.edu

Abstract

Novel view synthesis refers to the generation of images from previously unseen viewpoints using captured images from various angles. Neural Radiance Fields (NeRF) [5] represents a significant advancement in this field, generating high-quality images from a single continuous 5D coordinate input. However, the original NeRF model often requires extensive training time, limiting its real-time applicability. In this work, we introduce HashNeRF [6], a novel approach to do position encoding. This method enables the use of smaller networks without compromising quality, resulting in accelerated training of NeRF models. We also conduct new analyses on HashNeRF by testing it on a new dataset, making modifications to the original model, and inspecting its performance. Our research dives into the intricacies of HashNeRF, showcasing its unparalleled advantages in terms of training efficiency and rendering quality.

1. Introduction & Related Work

Novel view synthesis is the task of generating images of a specific subject or scene from a particular viewpoint, using only images captured from different viewpoints, which is crucial for various applications in virtual reality, gaming, video enhancement, etc. Neural Radiance Fields (NeRF) [5] represents a significant advancement in this field. NeRF takes a single continuous 5D coordinate (comprising 3D spatial location $\mathbf{x} = (x, y, z)$ and viewing direction $(\mathbf{d} = \phi, \theta)$) as input and produces volume density σ and view-dependent emitted radiance $\mathbf{c} = (r, g, b)$ at that spatial location. Views are synthesized by querying 5D coordinates along camera rays and applying classic volume rendering techniques [7] to project the output colors and densities into an image. However, the original NeRF model often requires hours to days to model a single scene for crisp and high-quality rendering, limiting its practicality for real-time applications. Therefore, speeding up training has become a focal point of NeRF research.

Several approaches have been proposed to address the challenge of training efficiency in NeRF models. DS-NeRF [2] leverages depth as an additional, cost-effective source of supervision to guide the geometry learned by NeRF, resulting in improved image rendering while training 2-3 times faster. DirectVoxGO [8] adopts a representation comprising a density and feature voxel grids with a shallow network for complex view-dependent appearance, enabling the NeRF model to converge rapidly from scratch. Plenoxels [3] even introduce a novel view-dependent sparse voxel model that achieves optimization to the same fidelity as NeRF without the use of neural networks.

In this work, we introduce a surprising approach HashNeRF proposed in Instant Neural Graphics Primitives with Multiresolution Hash Encoding [6]. This novel method of encoding position and viewing direction enables the use of a smaller network without sacrificing quality, thereby accelerating the training of new NeRF models. Furthermore, this encoding method can be applied to other tasks that utilize positional ncoding, including Neural Signed Distance Functions (SDF), Neural Radiance Caching (NRC), and Gigapixel images.

2. Methodology

Given a fully connected neural network $m = (y; \phi)$, which corresponds to the original NeRF model, HashNeRF proposes a multiresolution hash encoding method $y = enc(x; \theta)$, where x represents the 3D location coordinate and θ represents the trainable encoding parameters. Despite the apparent increase in the number of trainable parameters, this encoding method has been demonstrated to reduce the depth of the subsequent fully connected neural network while maintaining result quality. Consequently, this approach accelerates the training time of the NeRF model.

2.1. Background

According to Mildenhall et al. [5], the original NeRF model employs a positional encoding function known as $\gamma(p)$:

$$\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), \sin(2^1\pi p), \cos(2^1\pi p), \ldots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)) \quad (1)$$

where L is an integer chosen experimentally. This function γ applies sine and cosine transformations to each of the three coordinate values in x, which are normalized to lie in the range $[-1, 1]$. This mapping effectively expands the input space from $\mathbb{R}$ to $\mathbb{R}^2$, enabling better representation and fitting of the data.

As mentioned, HashNeRF introduces a novel encoding method that surpasses the performance of γ, leading to improved results in fewer iterations. This advancement in encoding technique plays a crucial role in enhancing the efficiency and effectiveness of the NeRF model.

2.2. Muitiresolution Hash Encoding

Multiresolution Hash Encoding is a technique used in to efficiently encode position. The trainable encoding parameters θ are organized into L levels, each containing up to T feature vectors with length F. Figure 1 illustrates an example with $L = 2$, $T = 8$, and $F = 2$ in a 2D world. At each resolution level, independent voxels are set and a hash table is used to store feature vectors. To establish the resolution for each level, it is essential to define both the coarsest ($N_{\min}$) and finest ($N_{\max}$) resolutions. These values set the range of resolutions spanned across the levels, with $N_{\min}$ representing the resolution of the coarsest level and $N_{\max}$ indicating the resolution of the finest levels. As for layers in the middle, the resolutions are computed by the following formula:

$$N_l := \lfloor N_{\min} \cdot b^l \rfloor \quad (2)$$

$$b := \exp\left(\frac{\ln N_{\max} - \ln N_{\min}}{L-1}\right) \quad (3)$$

To encode a given 3D position coordinate $\mathbf{x} \in \mathbb{R}^3$, we first identify the 8 surrounding voxels at resolution level l and assign each voxel to an entry in the hash table of that level. For coarser levels where the total number of voxels is less than T (i.e., $N_l + 1 \leq T$), each voxel corresponds to a unique entry. However, at finer levels, we need the following hash function to map voxel to index in the hash table:

$$h(x) := \left(\bigoplus_{i=1}^{3} x_i \pi_i\right) \bmod T \quad (4)$$

where $\bigoplus$ denotes the bit-wise XOR operation and π_i are unique large prime numbers. It may be quite confusing that hash collisions may occur without any resolving techniques. However, the author points out that the subsequent neural network can resolve the collision issue. In a 3D world, only a few points lie on the visible surface of a radiance field, while others reside in empty space and are less meaningful for the model. Consequently, the gradients of the former type of points are dominant and undergo larger changes, naturally optimizing the hash tables.

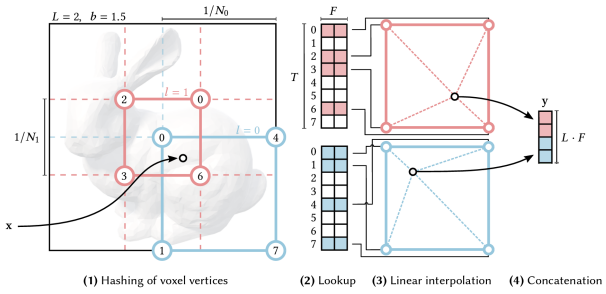

Figure 1. Illustration of the multiresolution hash encoding process (depicted in 2D for simplification). (1) For a given 3D position coordinate $\mathbf{x} \in \mathbb{R}^3$, we first identify the 8 surrounding voxels at resolution level l and assign each voxel to an entry in the hash table of that level. (2) We retrieve the associated feature vectors by looking up the corresponding entries in the hash table. (3) We then perform linear interpolation based on the relative position of $\mathbf{x}$. (4) Finally, we concatenate the results from each level to obtain the final encoding vectors. This figure is adapted from [6] with slight modifications.

Once we obtain the 8 resulting indices, we look up the corresponding entries in the hash table and retrieve the associated feature vectors. These vectors are then linearly interpolated based on the relative position of $\mathbf{x}$. After obtaining the results from each level, they are concatenated to form the final encoding. For instance, in a 3D world with $L = 2$ and $F = 2$ (T would not affect the length of the final encoding), the final encoding vector y would have a length of $2 \times 2 = 32$.

2.3. HashNeRF model

The HashNeRF model consists of two main components: a fully connected neural network (FCNN) with one hidden layer for predicting the volume density σ, followed by another FCNN with two hidden layers for predicting RGB colors, both 64 neurons wide. This indicates that the HashNeRF model has significantly fewer parameters compared to the original NeRF. Here are the specific details of how they are constructed:

Density FCNN

- The density FCNN has 1 hidden layer.
- The encoding vector $y = enc(x; \theta)$ is passed into the density FCNN.
- The output is a vector of length 16.

- The first value of the output represents predicted log-space density.

RGB FCNN

- The RGB FCNN has 2 hidden layers.
- The RGB FCNN takes as input the concatenation of two vectors: the 16-length output vector from the density FCNN, and the view direction projected onto the initial 16 coefficients of the spherical harmonics basis $\delta(\mathbf{d})$.
- The output is a vector of length 3, representing the RGB radiance values.

Despite the different model architecture from NeRF, HashNeRF is trained by backpropagating through a differentiable ray marcher guided by 2D RGB images captured from known camera poses, similar to NeRF's training process.

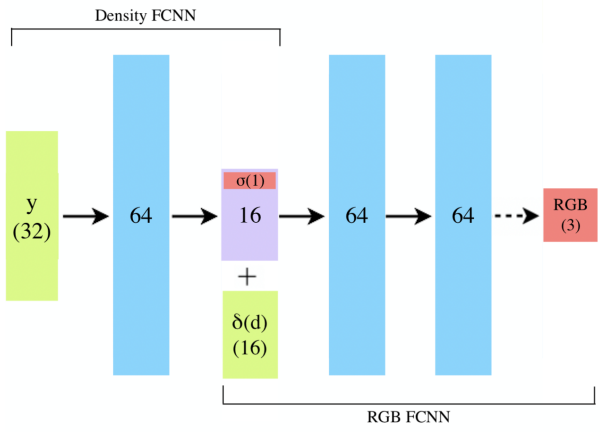

Figure 2. A visualization of HashNeRF model architecture. Input vectors are shown in green, hidden layers of FCNNs are shown in blue, the output vector of the density FCNN is shown in purple, and the output vectors are shown in red. All layers are standard fully-connected layers, with black arrows indicating layers where ReLU activations are applied. Dashed black arrows indicate straightforward output without any activation function, and "+" denotes vector concatenation. The number within each block denotes the vector's dimension.

3. Experimental Results & Discussion

We employed a PyTorch implementation of HashNeRF provided by Bhalgat et al. [1] to explore variations of the HashNeRF model..

3.1. Variations of HashNeRF model

We experimentally developed two variations of the HashNeRF model, namely **HashNeRF-3RGB** and **HashNeRF-DRGB**, to assess their performance compared to the original HashNeRF model. Each model has trained on the Local Light Field Fusion (LLFF) dataset [4] for 5000 iterations, and we used the peak signal-to-noise ratio (PSNR) to evaluate the results. **HashNeRF-3RGB**: The density FCNN is identical to that of the original HashNeRF model, while we employed three separate FCNNs, each with two hidden layers, to predict RGB channels individually. **HashNeRF-DRGB**: We utilized a single FCNN with three hidden layers to simultaneously output both the density and RGB radiance.

The results are shown in Table 1. We observed that

	Fern	Room	Leaves	Orchids
HashNeRF	23.29	25.00	19.55	18.99
HashNeRF-3RGB	23.61	25.46	19.90	19.33
HashNeRF-DRGB	23.73	25.41	20.12	19.41

Table 1. Peak signal to noise ratio(PSNR) of the original HashNeRF vs. HashNeRF-3RGB vs. HashNeRF-DRGB.

HashNeRF-3RGB outperformed the original NeRF model. This may be attributed to the slight increase in parameters and the separation of color channel FCNN models. The separate RGB FCNNs enable the model to capture the intricacies of each color channel independently, leading to improved reconstruction quality, particularly in scenes with complex lighting and color variations. Additionally, HashNeRF exhibited significantly better performance than HashNeRF in high-frequency areas (see Figure 3), which confirms our hypothesis. Similarly, HashNeRF-DRGB also outperformed the original NeRF. This could be attributed to the joint learning of representations for density and RGB radiance prediction, allowing HashNeRF-DRGB to better capture the complex interactions and dependencies between scene geometry and color channels. Like HashNeRF-3RGB, HashNeRF-DRGB also excelled in handling high-frequency areas compared to HashNeRF.

4. Conclusion

In this work, we introduced HashNeRF, a novel method for accelerating the training of Neural Radiance Fields (NeRF) models while maintaining rendering quality. By employing a multiresolution hash encoding approach, HashNeRF enables the use of smaller networks, leading to faster training times. Our experiments showed that HashNeRF variants outperform the original NeRF model, particularly in high-frequency areas and complex scenes. Overall, HashNeRF offers a promising avenue for efficient and high-quality novel view synthesis in various applications.

Figure 3. Fern rendering results after 5000 iterations of training . The result for HashNeRF-3RGB and HashNeRF-DRGB are better than that of HashNeRF in high frequency region (marked in red box). Visit here to see more visualized results. (left) Result for HashNeRF. (middle) Result for HashNeRF-3RGB (right) Result for HashNeRF-DRGB.

References

[1] Yash Bhalgat. Hashnerf-pytorch. https://github.com/yashbhalgat/HashNeRF-pytorch/, 2022. 3

[2] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 1

[3] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 1

[4] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 3

[5] 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, 2020. 1

[6] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 1, 2

[7] Thomas Porter and Tom Duff. Compositing digital images. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 253–259, New York, NY, USA, 1984. Association for Computing Machinery. 1

[8] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 1