Importance Sampling of Views in NeRF

Albert Ge
Harvard University *
albertge@g.harvard.edu, albertge@mit.edu

Abstract

View synthesis is a deeply studied problem in which new viewpoints of a given scene are generated. A recent breakthrough in the field optimizes a neural network to learn a radiance field representation (NeRF), and achieves state of the art rendering of novel views. Furthermore, there have been several advancements in rendering speed, to reduce time per iteration and speedup the overall training and inference speed. Our project specifically investigates an importance sampling scheme of viewpoints, which focuses on optimizing hard-to-learn/tricky viewing directions, and attempts to reduce the total number of iterations needed to achieve photorealistic results. While our project does not improve the synthesis quality, we report on which views are particularly difficult to optimize, opening further exploration on sampling techniques for NeRF.

1. Introduction

View synthesis is a longstanding problem in which a new views of a scene can be constructed from a limited set of pictures from a given point of views. A recent breakthrough in the field of view synthesis utilizes optimizes a neural radiance field representation [6]. The authors propose representing a view of a position of a scene as a 5-dimensional vector, where $\mathbf{x} = (x, y, z)$ represents the 3D location, and $\mathbf{d} = (\theta, \phi)$ represent the viewing direction of that location. This vector is input into a model which predicts the RGB color value of the scene $\mathbf{c}$, and volume density σ.

One unique feature about NeRF is its incredible sample efficiency relative to other deep learning techniques. In the field of computer vision, especially object recognition, it is well-established that large datasets such as Imagenet [9] are necessary to achieve model convergence of neural networks, with over 1000 images per class. NeRF only requires on the order a hundred of images of a scene to generate photorealistic results - an order of magnitude less data per subject.

In this project, we further investigate this sample efficiency idea by evaluating which training views may be more "informative" for our model to train on. This is a borrowed idea from statistics, known as importance sampling: certain input values may have more impact on the model parameter being estimated. Our method learns the informativeness of each view by learning the weighting scheme of each view, such that the model will sample more frequently from views it has trouble reconstructing. Our work shows that this importance sampling method is able to generate good reconstruction of novel views, although it is not able to significantly improve the baseline. Additionally, we report on the most informative views for our model by examining their weights.

2. Related Work

Existing work has shown that an autoencoder architecture can learn an embedding of 3D shapes, by mapping a 3D location to a signed distance function [2]. However, this requires a large quantity of ground truth 3D shapes to train the network. Additionally, these methods do not generalize well to complex scene geometry, and the embeddings only learn a simple smoothed representation of the scene. Other work shows that high quality view synthesis is possible using interpolation techniques between sampled viewpoints, but this can only be achieved through densely-sampled views [4].

NeRF combines the best of both worlds, by using a neural network to encode the 5-dimensional radiance field of the scene, and only requires a sparse sampling of views to achieve the same result. NeRF's hierarchical sampling technique is actually a form of importance sampling, as it samples locations likely of interest along the ray when rendering, to reduce rendering time. Interestingly, NeRF uses a uniform sampling method when choosing a training to view optimize. Thus, our project focuses on importance sampling of the input views themselves: the flat 2D images and their associated poses. Our scientific novelty is in implementing a method which will automatically learn weights with which to sample from our training set, which will give us a measure of which views are "better" at capturing the

*Cross-registered student

scene geometry.

Separate work by Katharopoulos and Fleuret [3] has been shown on the use of importance sampling for deep neural networks. Their paper derives a method to focus computation on samples that would significantly update the model parameters. They reformulate this problem as a variance reduction of the training estimates; in other words, as the model trains, it gradually will become more confident in its predictions, and these predictions will vary less over time. When variance is sufficiently reduced (past a threshold), the model can estimate the importance of each sample, and sample from the data according to this metric.

However, their paper imposes the constraint that the dataset size is large, and that would be infeasible to compute the importance score for all training data. Their solution is to sample enough times to estimate the model's total variance reduction, and then only set the weights once. In our project, we can relax both these constraints; our dataset size is around 100 images, so we can compute the importance score for the entire set; furthermore, this allows us to reweight multiple times over the course of training.

3. Approach

3.1. Baseline NeRF

Our baseline model for comparison is directly taken from an original implementation of NeRF, previously written in Pytorch [11]. For comprehension, we elucidate the features of this baseline model.

The model is a simple MLP with 8 fully-connected layers, ReLU activation, and 256 units per layer. The model takes as input the positional encoding of the 5D input. There is a skip connection at the fifth layer, which concatenates the input with the fifth layers' activation. At the end of the MLP, there is an additional layer with no activation that outputs the volume density, and a 256-dimensional feature vector. This feature vector is concatenated with the original positional encoding, and passed through a final fully connected layer with sigmoid activation outputs the RGB radiance of the view of the scene. This final layer ensures a consistent multivew of the scene - only the position $\mathbf{x}$ is used to predict the volume density σ, whereas the color $\mathbf{c}$ can be predicted using both the position and viewing direction.

Positional encoding is an idea similar to the one used in the Transformer architecture [10]: to map the 5D inputs into a high-dimensional space. This purportedly helps the MLP model generalize better, as it is more suited for data with high-frequency variation [7]. Formally, we define the positional encoding as

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

To estimate the pixel value from a particular viewing direction, the authors take inspiration from classic volume rendering techniques, which can be summarized by the following equation for the expected color of a pixel, $C(\mathbf{r})$:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

$$T(t) = \exp\Big(-\int_{t_n}^{t_f} \sigma(\mathbf{r}(s))ds\Big)$$

Here, t_n and t_f are the near and far bounds of the camera ray, and $T(t)$ represents the probability that the ray travels from t_n to t without hitting any particle. Integrating across the bounds of the ray gives an average of how the position looks from that viewing direction.

Finally, the authors implement hierarchical sampling by utilizing two neural networks to represent scenes: one "coarse" and one "fine" network. The "coarse" network is trained using a coarse sampling of locations along the ray. The "fine" network is trained using a sampling of locations where there is density as predicted by the coarse network. The loss of the networks is the total squared error between rendered and true pixel colors.

3.2. Importance Sampling of Views

We reference the derived formula for variance reduction from Katharopoulos and Fleuret [3], which can be stated as

$$\frac{1}{\tau} = 1 - \frac{1}{\sum_{i=1}^{B} g_i^2}||g - u||_2^2$$

Where $\frac{1}{\tau}$ is the variance reduction, B is the batch size, and $||g - u||_2^2$ is the squared L_2 distance of the sampling distribution g to the uniform distribution u. Our project, for simplicity, replaces g_i with the model importance score, L_i. The paper calculates the moving average of τ with a momentum hyperparameter a_τ, which we also implement in our methods.

Algorithm 1 describes a high-level overview of our implemented method. We begin with training data $X = \{(x, y, z, \theta, \phi)\}$, and batch size B. The weights of each sample in our training set, so that samples for training will be drawn from a uniform distribution. We first begin our training loop by sampling a batch with size B of our training data. NeRF is trained successively on each data point, then we will compute the variance reduction τ for this given batch.

Once τ crosses the threshold value τ_{th}, we will recalibrate the weights according to the model scoring of each training sample. Our model scoring is simply the mean rendering loss of each sample (higher loss means the model has more difficulty reconstructing that pose, so it should be sampled more often). Additionally, we will reset our variance reduction, so that in subsequent iterations the model will reconverge according to the new weighted distribution. This idea allows the model to adjust the importance weights over time.

Algorithm 1 NeRF with Importance Sampling

Inputs $X = \{(x, y, z, \theta, \phi)\}, B$
$\tau \leftarrow 0$
$w_i \leftarrow \frac{1}{|X|}, \forall x_i \in X$
repeat
  if $\tau \leq \tau_{th}$ **then**
    x sampled B times from X with weights w
    $L = \mathcal{L}(\text{NeRF}(x_i)), \forall x_i \in X$
    NeRF $\leftarrow$ sgd_step(∇L)
    $\tau \leftarrow a_\tau \tau + (1 - a_\tau)(1 - \frac{1}{\sum L_i^2}||L_i - \frac{1}{B}||_2^2)^{-1}$
  else
    $w_i \leftarrow \mathcal{L}(\text{NeRF}(x_i)),\ \forall x_i \in X$
    $\tau = 0$
  end if
until convergence

3.3. Experiments

We implement this algorithm on a fully-featured NeRF model (includes hierarchical sampling, positional encoding). This model is trained on the lego bulldozer dataset, which is split into 100 training images, 25 test images, and 13 validation images. Each image has size $(400 \times 400 \times 3)$, and is associated with a (4×4) camera pose. This pose represents homogenous camera-to-world transformation matrix (which places the camera in the position and orientation relative to the world view).

Our model is trained for 100000 iterations, which takes approximately 9 hours on a Tesla T4 GPU. For the model, we use the Adam optimizer with learning rate 5×10^{-4}, and $N = 64$ samples along the ray. For the importance sampling method, we use a threshold of $\tau_{th} = 1.2$ and momentum $\alpha_\tau = 0.9$. To evaluate the convergence of our models, we record the training PSNR (peak-signal-to-noise ratio [1] , or the negative log loss). Every 25000 iterations, we render holdout views on the test set to provide a qualitiative measure of our model results. We also trained the baseline model, using the same hyperparameters and for 100000 iterations.

During training, we noticed that the NeRF model is sensitive to poor initialized states. Thus, our experiments include two runs of the baselined model to estimate its overall performance. In the interest of time, we were unable to generate a similar average estimate of NeRF with importance sampling; thus, we only declare a single best run for importance sampling.

4. Results

Fig. 1 shows the results of PSNR for our varying model runs. Our importance sampling method is able to learn complex scene geometry of the lego bulldozer quite well. However, it does not perform the baseline best result. Our importance sampling method consistently reaches PSNR values higher than 25, but the baseline best is able to achieve PSNR values above 30. Fig. 3 and Fig. 4 provide a qualitative comparison between the these two methods. Even by iteration 25000, we observe that the scene geometry is clearer in the baseline, while it is still blurry for the importance sampling method. By iteration 100000, the different in image quality is most clear: The bumped ridges of the lego toy contrast well with respect to the lighting.

The importance sampling run does seem to outperform the baseline average, and preliminary future experiments suggest that the importance sampling is on average most consistent in its training process than the baseline average.

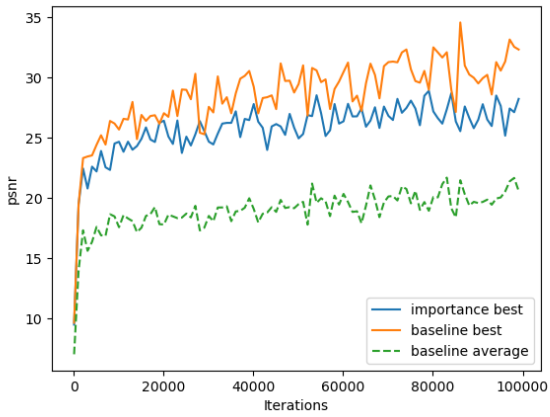


Figure 1. Training PSNR over number iterations. Baseline average refers to the average PSNR across three different runs of the baseline model. Due to time constraints, however, an average of importance sampling runs was not collected.

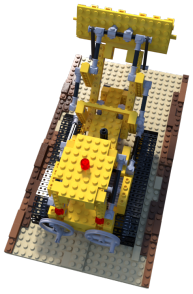
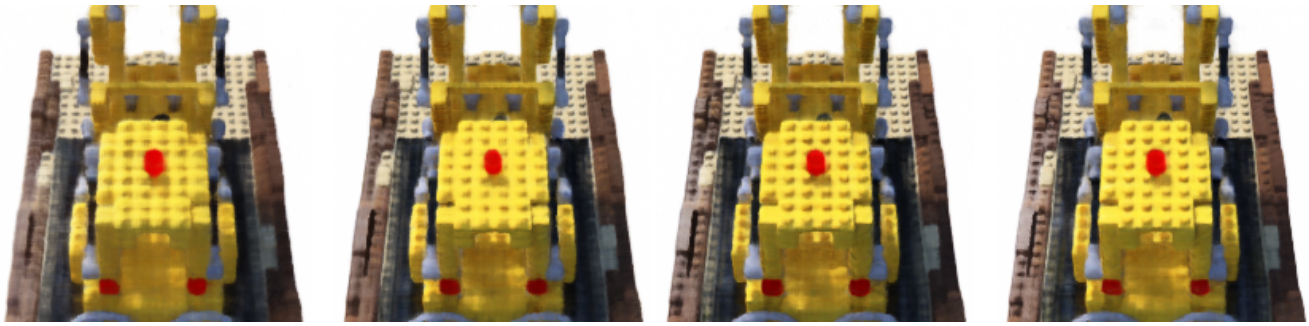


Figure 2. Training image 12, which is difficult for NeRF to reconstruct.

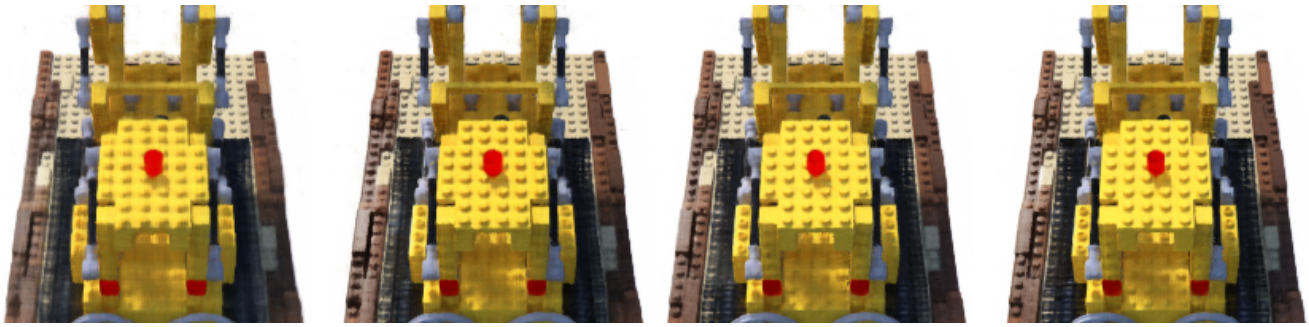
4.1. Which view is most informative?

We report an interesting finding in which one view in our training set is consistently difficult for NeRF to reconstruct, and has a high sampling rate. Fig. 2 illustrates this pose,



(a) Iteration 25000 (b) Iteration 50000 (c) Iteration 75000 (d) Iteration 100000

Figure 3. Importance sampling



(a) Iteration 25000 (b) Iteration 50000 (c) Iteration 75000 (d) Iteration 100000

Figure 4. Baseline Best

Iteration #	Importance Score
10000	0.0165
50000	0.0169
100000	0.0171

Table 1. Importance Score of Fig. 2

and Tab. 1 shows its importance score at certain iterations.

5. Discussion

In this section we outline possible theories as to why importance sampling did not seem to improve over the baseline image quality. Of primary concern is the sensitivity of NeRF to initialization. The variance of PSNR across baseline runs is quite large, and results in drastically different rendering qualities. Time permitting, more experiments should have been run in order to get a better estimate of robust each of these two methods are to initialization.

Additionally, the reweighting scheme may have affected the model quality more than anticipated. Tab. 1 shows that training image 12 is sampled with a frequency of 0.0171, which is almost twice the frequency over a uniform distribution (0.01). More informative (harder to reconstruct) training views are sampled most frequently, and we fully expect the model's training performance to be worse, but we did not expect test performance to suffer as well. Further experiments adjusting the variance reduction threshold may provide a more comprehensive understanding of how the model reacts to the weighting policy.

6. Conclusion

We devise a method combining importance sampling for deep neural networks with a neural radiance field representation of a 3D scene. Our method implements importance sampling on the training views themselves, rather than on the rendered points in the 3D space. While we do not demonstrate an clear improvement over the baseline NeRF model, we consider possible routes in quantifying sample importance in NeRF models.

6.1. Future Work

Possible future directions for the reweighting scheme have been briefly discussed in the Discussion section, such as changing varying parameters to observe the effect on the model. An alternative approach to measure sample importance is to consider how much a new view contributes to the certainty of the RGB and density of particular point. To illustrate this idea, two views with similar viewing angles (*i.e.* largely overlapping frustums) would not contribute much to

the certainty of a 3D point, whereas views with different viewing angles should greatly increase the model's confidence of that location. Thus, instead of the model learning the importance scores, each view's importance can be a function of the both its location and viewing direction.

Other areas of interest are on improving sampling techniques of the rendering itself. The original paper already uses hierarchical sampling as one means, but there is also exciting work to further sample on locations of interest, such as empty space skipping and early ray termination [8] [5]. This projects represents a step toward understanding how to apply different sampling techniques to improve both training and inference time for neural graphics rendering.

References

[1] M. Ghanbari. Scope of validity of psnr in image/video quality assessment. *Electronics Letters*, 44:800–801(1), June 2008. 3

[2] Chiyu Max Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes, 2020. 1

[3] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling, 2018. 2

[4] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996. 1

[5] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields, 2020. 5

[6] 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

[7] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2018. 2

[8] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2021. 5

[9] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2014. 1

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 2

[11] Lin Yen-Chen. Nerf-pytorch. https://github.com/yenchenlin/nerf-pytorch/, 2020. 2