High School Student Institute of Mathematics and Informatics

Summer Research School
Symposium
2022

Pruning NeRFs

Author(s):

Erik Paskalev
NPMG "Academician Lyubomir Chakalov"
erik.paskalev@gmail.com

Scientific Advisor(s):

Victor Kolev
Stanford University
vkolev@stanford.edu

Abstract

In this paper, we explore the viability of pruning as a method for speeding up Neural Radiance Fields (NeRFs). The pruning method used is based on the Lottery Ticket Hypothesis. Unfortunately, the results imply pruning has the opposite effect on inference time, increasing it by almost half a second. However these results cannot be taken at face value, as the data used was a fraction of that of the original and the model size we trained was much smaller then that of the original paper. This leaves much to be desired from

1 Introduction

One of the fastest growing fields of Computer Vision today is Novel View Synthesis. Novel View Synthesis is solving the problem of generating 2D images of a 3D scene from a set of original 2D images. The amazing part is that most solutions don't keep a precise 3D model like most computer games and CAD (Computer Aided Design) software solutions do. Within the Novel View Synthesis field, the NeRF neural net architecture (short for Neural Radiance Field) has been a major breakthrough and has seen a meteoric rise in research, funding, and published papers.

Our work aims to improve upon the original implementation of the NeRF neural architecture, by speeding up the inference time of the model through a universal optimization method known as pruning as per the Lottery Ticket Hypothesis. [2] The original NeRF model aims to solve one of the long-standing problems in Computer Vision view synthesis - the creation of a target image given a camera position, angle, and focal length.

2 Related work

The original paper that first introduced the idea of NeRFs is described in [5]. The original idea was to create a model of the 3D scene. That model needs an objective for the neural network to optimize. The one chosen in the paper is to minimize the error between the rendering of a model and a set of "test set" images. Each problem/scene is represented by a 5D input function/field that outputs RGB color and volume density (radiance). The 5D of the input includes the obvious 3D coordinates and two parameters for each 3D point and two angles describing the viewing direction at a given point. The neural network itself is a bit uncommon as well. It is fully connected and has no convolutional layers. This architecture is called a multi-layer perceptron.

More recently there has been significant progress in the optimization of the NeRF. For example, FastNeRF [3] has been able to achieve real-time photo-realistic image generation around 3000 times faster than the original. The way this is achieved is by addressing the worst bottleneck of NeRF, which resides in the NeRF's volumetric scene representation. More than 100 neural network calls are required to render a single image pixel, which translates into several seconds being required to render low-resolution images on high-end GPUs. For a trained NeRF model, a bounding box V that covers the entire scene captured by the NeRF is defined. Then, a set of points on the ray are sampled uniformly. Similarly, they uniformly sample l values for each of the ray direction coordinates (θ, ϕ) = d with $\theta \in h_0$, πi and $\phi \in h_0, 2\pi$i. The cache is then generated by computing F for each combination of sampled p and d. This makes the bottleneck memory rather than computation and allows you to balance between the 2.

3 Neural Radiance Field

NeRFs represent a continuous scene as a 5D vector-valued function whose input is a 3D location x = (x, y, z) and 2D viewing direction (θ, ϕ), and whose output is an emitted color c = (r, g, b) and volume density σ. In practice, we express direction as a 3D Cartesian unit vector d. We approximate this continuous 5D scene representation with an MLP network F_θ* : (x, d) $\rightarrow$ (c,σ) and optimize its weights θ to map from each input 5D coordinate to its corresponding volume density and directional emitted color. The neural network itself consists of 8 layers with 256 neurons per layer with a RELU activation between each layer and outputs σ and a 256-dimensional feature vector. This feature vector is then concatenated with the camera ray's viewing direction and passed to one additional fully-connected layer (using a ReLU activation and 128 channels) that output the view-dependent RGB color.[5]

This is the core model that generates novel views. However, it isn't enough for state-of-the-art performance. There are two improvements that were used in the original paper: Positional encoding and Hierarchical volume sampling. Positional encoding helps the model represent high-frequency functions and hierarchical sampling is a procedure that allows us to efficiently sample the high-frequency representation.

3.1 Positional encoding

The original paper noted that directly operating on the raw 5D input causes the neural network to underperform at representing the high-frequency variations in color and geometry. This is consistent with findings from [6], which found that neural networks are biased towards learning low-frequency functions. This problem can be improved by mapping the input data to higher dimensional spaces. These observations were utilized to show that reformulating the function F_θ into a composition of two functions $F_\theta^{'} \circ \gamma$ where γ is the positional encoding function.

3.2 Hierarchical sampling

Hierarchical sampling solves a very important problem - slow rendering. Rendering is slow because the NeRF samples many times along each rendered ray - no matter how much free space (no objects) or hidden objects (behind the first visible object) exist in the 3D scene. The hierarchical sampling method is inspired by ideas from [4]. It is using a hierarchical representation that samples unevenly and the sampling frequency is proportional to the estimated effect on each rendered ray.

4 Pruning

Pruning is an optimization technique that works by removing a certain percentage of all weights in a neural network. There exist many different kinds of pruning strategies. However, the one we used is based on the Lottery Ticket Hypothesis [2]. The Lottery Ticket Hypothesis states that for dense layers, most of the predictive power of the neural network is caused by a small subset of it. This means that we can prune the majority of the weights with the lowest values, thus reducing memory and computation time, while retaining satisfactory performance. The specific pruning implementation that the original paper found to be most optimal does pruning on a pre-trained model. Next, 90-95 percent of weights in the network are pruned. The final step is to re-training the new 10x-20x smaller network. This retains much of the performance while making the network faster and less memory intensive.

5 Results

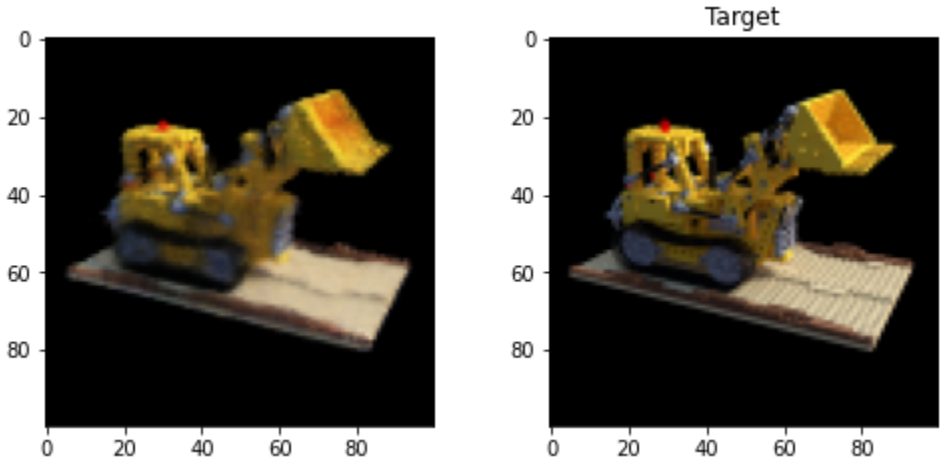
The final results we achieved can be seen in the figure below. The number in the model name signifies how many training iterations the model was trained for. The number after pruning denotes what percent of the neurons were pruned, and the x + y tells how many iterations the model was pre-trained for and how many fine-tuning iterations the model underwent.

The most notable and unexpected result is that pruning takes a longer time instead of getting some speedup. This took some time to investigate and we still haven't gotten to the bottom of it - it is our top priority for future work. The data we used for training and validation are a small subset of the original(only 106 images of only 1 type) meaning our results are incomparable to the original paper. What we know already is that runtime measurements have low statistical reliability on the Google Colab setup that we were using.

Another note is that the above networks have 2-layers with 128 neurons per layer vs the original 8-layers and 256 neurons. The main reason for this way the memory overflow we experienced during training likely caused by a poor implementation of a batching system meaning again our results are incomparable with the original paper.

Model name	L2 training		L2 validation	
	avg	median	avg	median
NeRF 5000	0.00582	0.00525	0.00602	0.00530
NeRF 10000	0.00406	0.00390	0.00433	0.00384
NeRF 20000	0.00337	0.00323	0.00357	0.00334
NeRF + Pruning95 5000 + 5000	0.00413	0.00379	0.00422	0.00362
NeRF + Pruning95 10000 + 10000	0.00313	0.00295	0.00321	0.00276
NeRF + Pruning95 20000 + 20000	0.00238	0.00230	0.00238	0.00214

Model name	PSNR training		PSNR validation		Time
	avg	median	avg	median	
NeRF 5000	22.54	23.02	22.20	22.76	2.06
NeRF 10000	24.07	24.26	23.62	24.15	2.06
NeRF 20000	24.88	25.08	24.47	24.82	2.05
NeRF + Pruning95 5000 + 5000	23.98	24.40	23.74	24.43	2.50
NeRF + Pruning95 10000 + 10000	25.18	25.47	24.93	25.62	2.53
NeRF + Pruning95 20000 + 20000	26.38	26.54	26.23	26.70	2.49



This is an above average example of the of image quality from the model we used. On the left is our image on the right is the original. The metrics for this image are: L2 - 0.00193, PSNR - 27.13.

6 Future work

There are many paths future research can take. First, the results achieved in this paper are questionable, and figuring out the cause of the unreliable testing, unexpected results, and fixing the memory and data issues will be key to any future progress. Beyond that further optimization like Quantization, multiscale representation [1] , learned initializations [8] , relighting [7] , batching [3]

References

[1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *ICCV*, 2021.

[2] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*. OpenReview.net, 2019. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2019.html#FrankleC19.

[3] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021.

[4] Marc Levoy. Efficient ray tracing of volume data. *ACM Trans. Graph.*, 9(3):245–261, jul 1990. ISSN 0730-0301. doi: 10.1145/78964.78965. URL https://doi.org/10.1145/78964.78965.

[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. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[6] 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. *arXiv e-prints*, art. arXiv:1806.08734, June 2018.

[7] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021.

[8] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *CVPR*, 2021.