CSCE689-608 Final Project
Coreset on NeRF

Dawei Xiang, Lipai Huang

Dec 1st, 2023

Layout

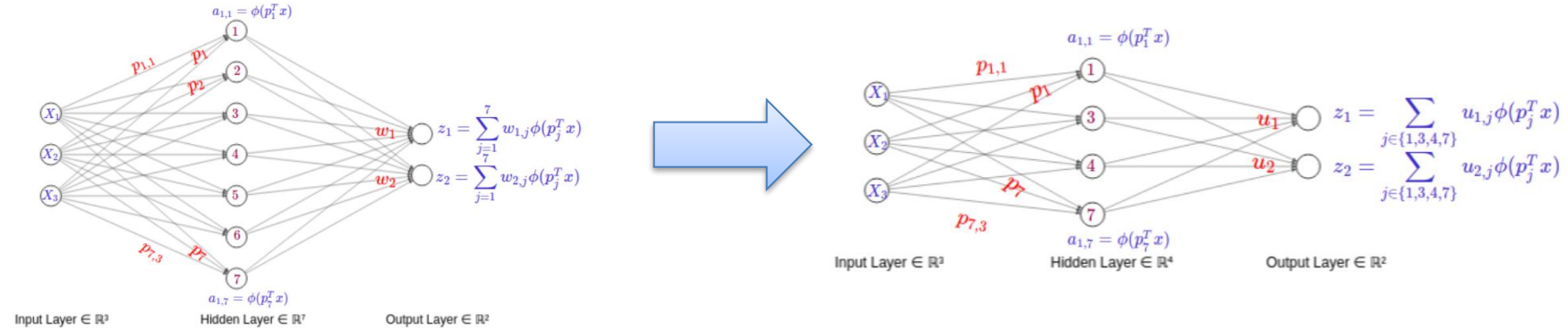
- Motivation
- Coreset on Neural Network
- Introduction of Neural Radiance Fields (NeRF)
- Methodologies
- Results
- Conclusion

1. Motivation

- Trend of Overparameterization -> Trend of smaller model
  - Less parameters but keeps performance
  - Rather sacrifice performance for lower computation cost
- 3D Reconstruction
  - Interesting topic
  - Relatively not over-explored
  - Coreset/pruning be adapted in 3D Reconstruction?
  - Performance? Cost?

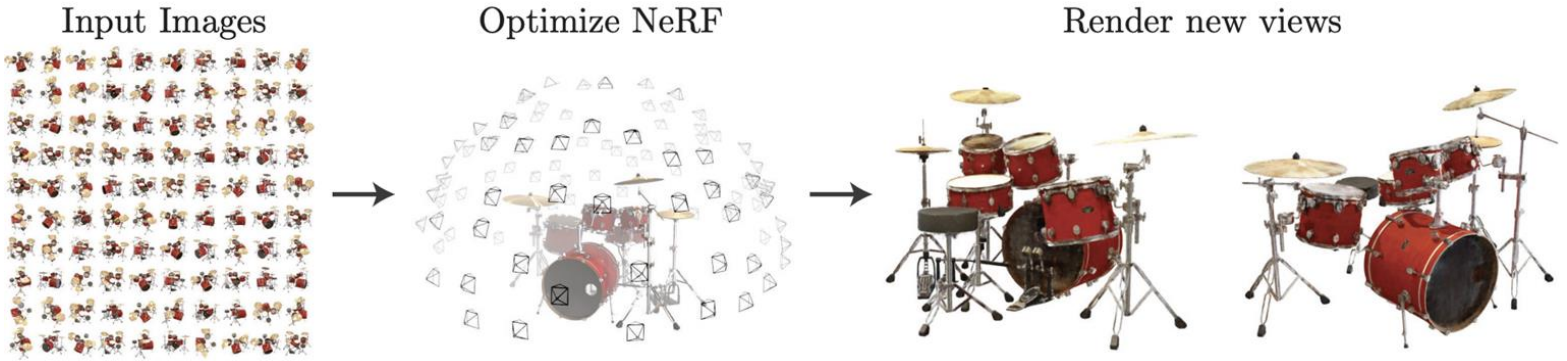
1. Coreset on Neural Network

- Well-pruned NN can guarantee the trade-off between Compression Rate and Approximation Error.
  - Input: neurons at current layer
  - Output: Selected optimal neurons and new connections weights for next layer
  - Apply layer-wisely



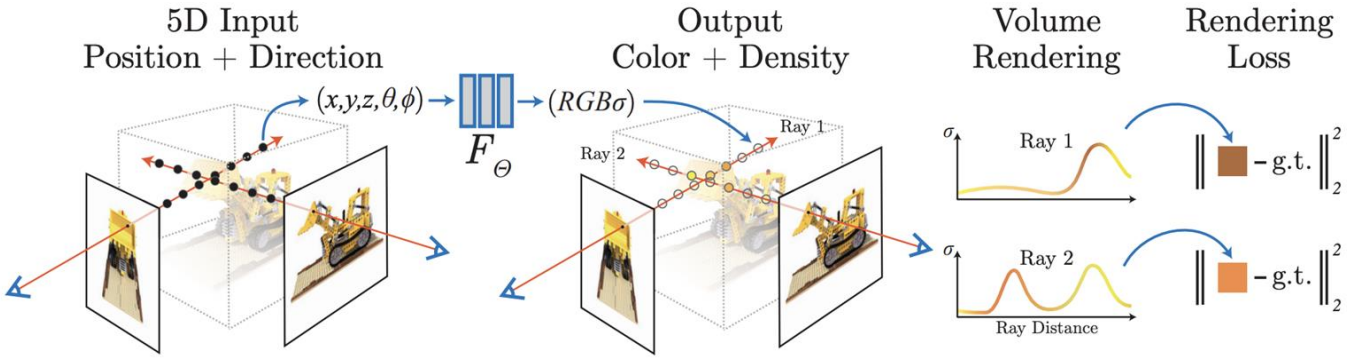
2. Introduction of NeRF

- Neural Radiance Fields (NeRF): creating highly realistic 3D models from a set of 2D images
- Input: images of an object.
- Output: a rendered image viewing from a new angle that doesn't appear in training set.



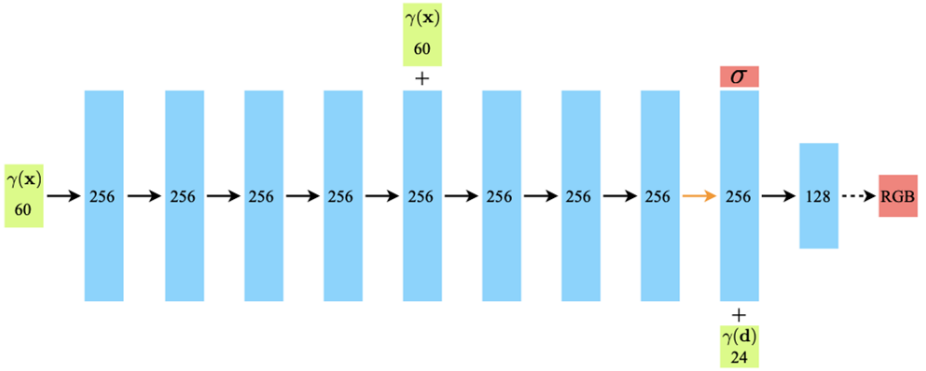
2. Introduction of NeRF

- Generate a NeRF from a specific viewpoint
  - Generate a sampled set of 3D points—by marching camera rays through the scene.
  - Produce an output set of densities and colors—by inputting sampled points with their corresponding 2D viewing directions into the neural network.
  - Accumulate densities and colors into a 2D image—by using classical volume rendering techniques.



2.1 NeRF base structure

- Backbone: MLP
- Input: 3D Location + 2D Direction
- Output: Emitted color RGB + Volume Density



γ(x): Positional encoding of input location
σ: Volume density
→: Layer with ReLU
⇢: Layer with Sigmoid
→: Layer without activation
+: Concatenation

2.2 NeRF main challenges

- Scene-specific
  - Dynamic scene problem
- Extensive Data Needs
  - Needs large number of input images taken from various viewpoints
- High Computational Requirements
  - Long Training Times
  - Extremely large parameters

related works

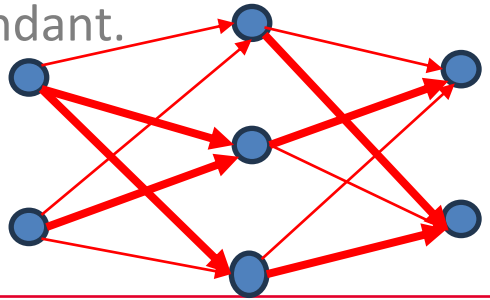
[1] has studied the feasibility of model compression in NeRF. It basically pruned some edges between neurons to reveal its sparsity. But we are looking at something deeper about the neurons instead of only edges. [2] used SVD decomposition to decompose the original NeRF representation to lower-rank representations. [3] proposed a pruning method working on 3D voxel-based representations as Instant-NGP and achieved satisfying result. Its main idea to prune the hash table with different limits.

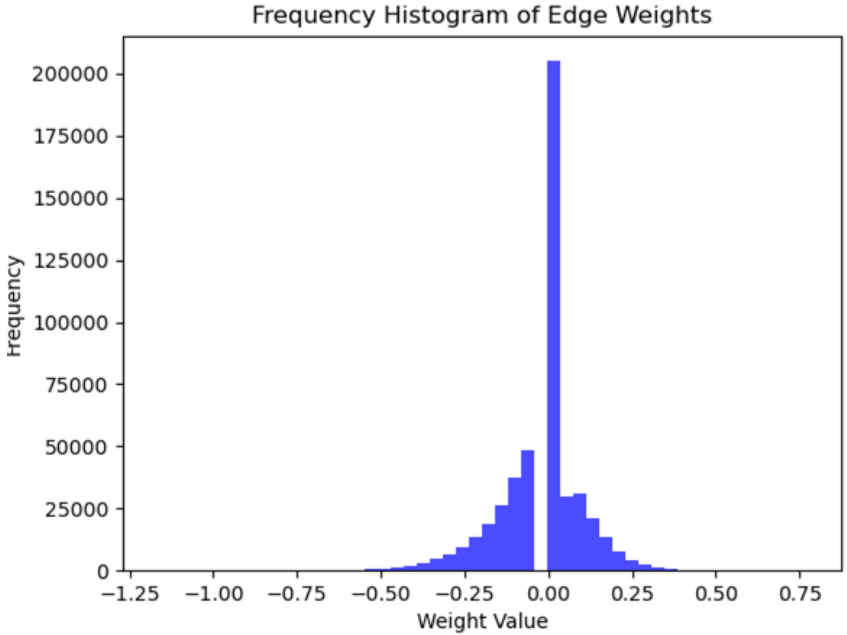
[1] Neural 3D Scene Compression via Model Compression. Arxiv.org
[2] Compressible-composable NeRF via Rank-residual Decomposition. NeuIPS 2022.
[3] HollowNeRF: Pruning Hashgrid-Based NeRFs with Trainable Collision Mitigation, ICCV 2023

3. Sparcity in NeRF MLP

We have extracted the edge weights in the above MLP model.

- a large amount of edge weights are indeed very small (less than 0.05) => MLP in NeRF is indeed sparse, and there is some space for compression.
- sparsity: some edges are redundant.





3.1 Pruning edges

TEXAS A&M UNIVERSITY

we first tried to prune some low-weight edges: we discard the edges that have weight less than 0.05 which are almost 40% of the total edges.

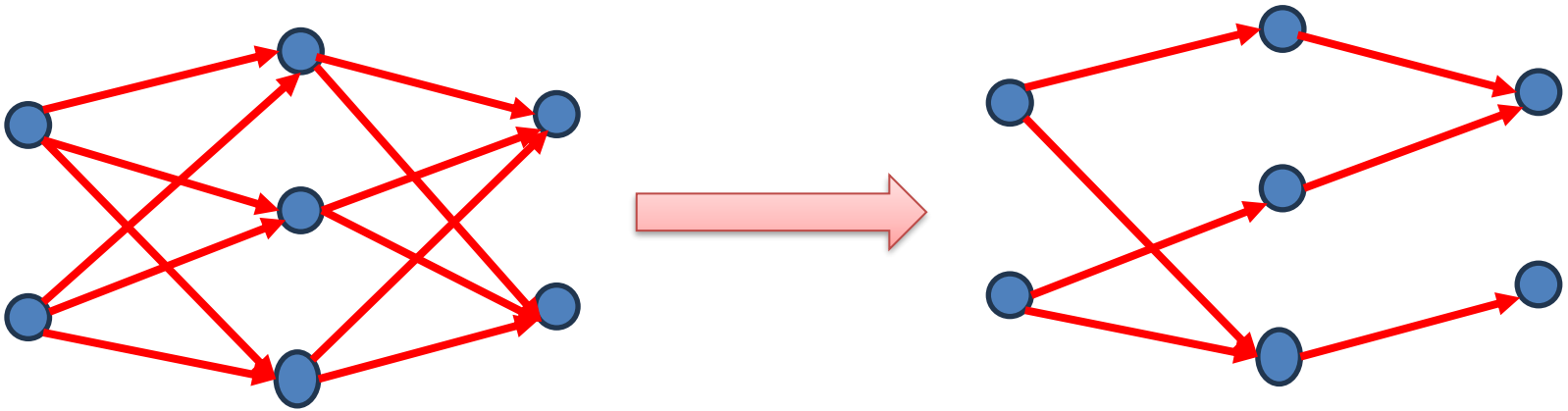
Edge pruning threshold	0	0.05	0.1
Remaining percentage	100%	40%	20%
PSNR on test set	21.5	21.3	20.8

this means around 40% of the edges can be considered as real important edges.

the problem of edge pruning only

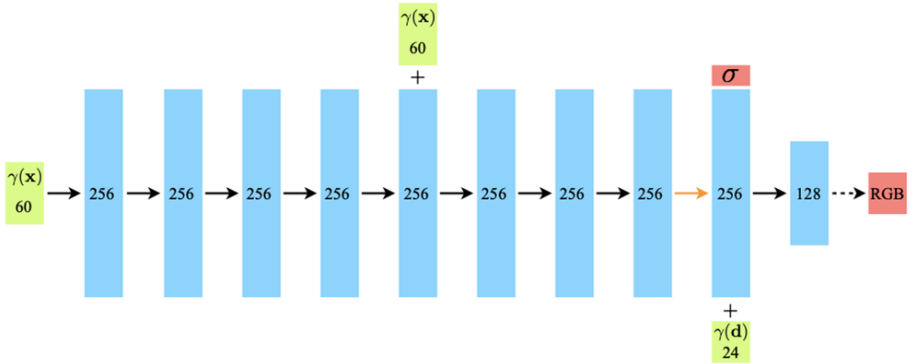
TEXAS A&M UNIVERSITY

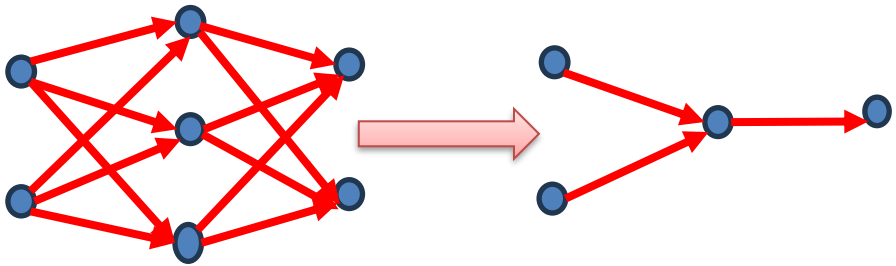
pruning edges could not bring substantial speed increase because still a lot of edges are not null. The weight matrix between two layers is still the same.



3.2 pruning neurons

The original MLP is connected by 7 layers of 256*256 full connection layer. Our initial idea is to find a way to compress to 64*64 fully connection layers for compression.





3.2.1 uniform sampling

TEXAS A&M UNIVERSITY

We first tried uniform sampling, which refers to select neurons in each layer with equal weight/possibility.

	baseline	Uniform sampling
Connection layer size	256*256	64*64
PSNR	21.5	16.5

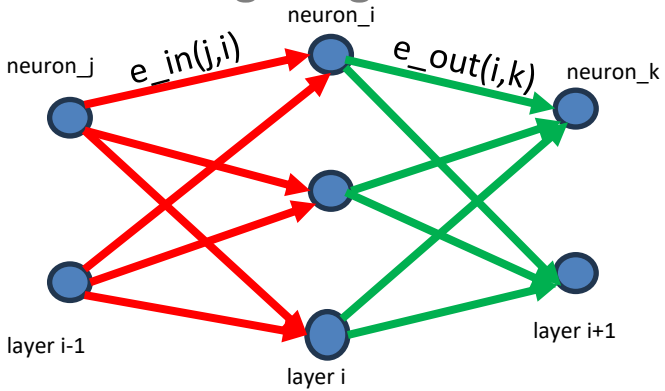
This shows:

1. randomly picking neurons doesn't work.
2. we need a way to find the important neurons.

3.2.2 importance sampling

Pruning by importance refers to prune the neurons by their importance weights. For each layer we select the top k neurons with the highest importance.

For a neuron i, we define two types of importance weight: incoming weight and outcoming weight:



$$w_{in}(i) = \frac{\sum_{j \in l_{i-1}} |e_{ji}|}{|l_{i-1}|}$$

$$w_{out}(i) = \frac{\sum_{k \in l_{i+1}} |e_{ik}|}{|l_{i+1}|}$$

for a neuron i in layer l_i, the previous layer is l_{i−1} and the next layer is l_{i+1}

	method	psnr
no pruning	baseline	21.5
importance pruning	select neurons with w_{in}	19.5
	select neurons with w_{out}	20.0
	select neurons with $w_{in} \times w_{out}$	20.0

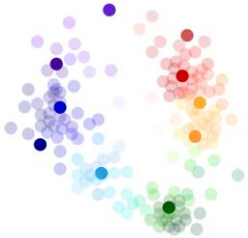
result of pruning 256*256 to 64*64

3.2.3 coreset

Coreset refers to construct a small set of points to approximate a large set.

basic idea: assign each point a weight and sample by probability proportional to its weight.

Massive Dataset



Coreset

Algorithm 1: CORESET(P, w, m, ϕ, β)

Input: A weighted set (P, w),
an integer (sample size) $m \geq 1$,
an (activation) function $\phi : \mathbb{R} \to [0, \infty)$,
an upper bound $\beta > 0$.

Output: A weighted set (C, u); see Theorem 7 and Corollary 8.

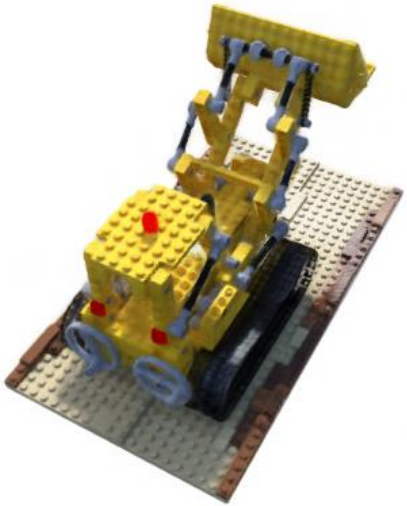
for *every* $p \in P$ **do**
  $\mathrm{pr}(p) := \dfrac{w(p)\phi(\beta \|p\|)}{\sum_{q \in P} w(q)\phi(\beta \|q\|)}$
  $u(p) := 0$
$C \leftarrow \emptyset$
for m *iterations* **do**
  Sample a point q from P such that $p \in P$ is chosen with probability $\mathrm{pr}(p)$.
  $C := C \cup \{q\}$
  $u(q) := u(q) + \dfrac{w(q)}{m \cdot \mathrm{pr}(q)}$ the weight don't sum to 1
return (C, u)

Connection layer size	256*256	64*64	64*64
Pruning	baseline	importance pruning	coreset
PSNR	21.5	20.0	20.1

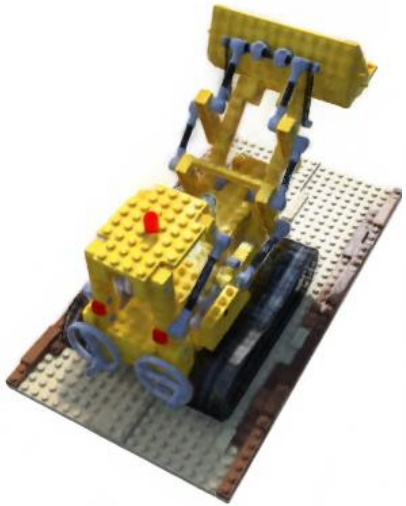
3.3 results

Connection layer size	256*256	128*128	64*64
Model parameter size	595K	288K	177K
Model size	2.38Mb	1.14Mb	0.7Mb
PSNR	21.5	**21.3**	20.1
Training time	78.75 min/100K iteration	51.2 min/100K iteration	46.25 min/100K iteration

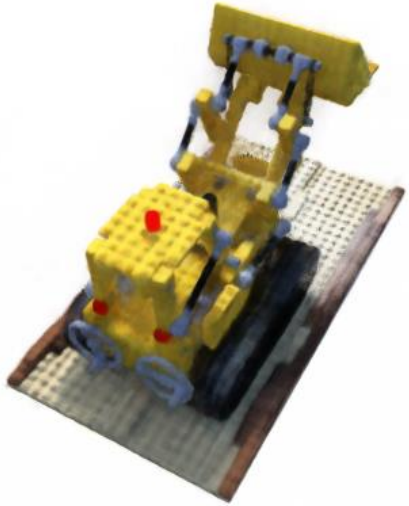
3.4 visualization results



baseline:256*256 PSNR:21.5



coreset: 128*128 PSNR:21.3



coreset: 64*64 PSNR:20.1

4. Conclusion& Future work

TEXAS A&M UNIVERSITY

Conclusion:

we show that importance pruning and coreset sampling could reduce the training time of NeRF by 35% and model size by 50% while keeping the performance.

Thank you for listening!