DistillNeRF: Data-Efficient Initialization of Neural Radiance Fields using Knowledge Distillation

Anonymous ACCV 2022 submission

Paper ID ***

Abstract. Neural Radiance Fields (NeRF) learn a high-quality continuous 3D implicit representation of a scene given multiple views. While the approach has gained popularity in Novel View Synthesis (NVS), its vanilla implementation is not suited for real-time applications. This paper presents DistillNeRF, a data-efficient method for initializing and breeding smaller models using Knowledge Distillation (KD). Smaller models naturally benefit from lower inference times but decrease in perceptual quality. DistillNeRF optimizes training time and builds a priority grid from a teacher network as a data-efficient proxy for sampling better training examples reducing the quality loss while using half the memory.

Fig. 1: DistillNeRF initializes smaller capacity NeRF models using KD from a teacher model. The figure is a depiction of the hot dog scene from the Blender Synthetic Dataset. The top row shows the occupancy voxel grid sampled from the pretrained NeRF. The middle row are renders from NeRF variations, and the bottom are their corresponding depth maps. The models used are NeRF, TinyNeRF, MicroNeRF, NanoNeRF, DistillNeRF, DistillTinyNeRF, DistillMicroNeRF, and DistillNanoNeRF respectively from left to right. The Distillation initialization reduces the perceptual quality loss observed in smaller models.

1 Introduction

Motivation. Novel View Synthesis (NVS) is a challenging problem for Computer Science as it requires an understanding of the 3D scene structure, image reasoning, and a method to render its internal representation into target viewpoints. Current advances in Deep Learning allowed neural approaches to gain popularity and enabled applications that were impossible to achieve without a significant loss in quality. Neural NVS is becoming crucial to graphics pipeline based on real-world imagery such as image and video viewpoint editing [1, 2], and photo-realistic facial puppetry in Virtual Reality (VR) chat applications [3, 4]. To this end, Neural Radiance Fields (NeRF) [5], learns a compact continuous 3D implicit representation of the scene and can generate high-quality novel views while dealing with thin, transparent, semi-transparent, and reflective surfaces.

Problem. NeRF optimizes a large capacity Multi Layer Perceptron (MLP) to regress a volume density and view-dependent RGB radiance from a 5D vector (x, y, z, θ, ϕ). In its vanilla implementation, NeRF uses two neural networks to perform a hierarchical sampling of the scene, a coarse for approximating the geometry and a fine to refine its prediction.

While this formulation of neural NVS allows unprecedented render quality, it suffers from long training and inference time, thus failing at providing real-time use-cases. These factors can be explained by the number of network queries needed to render a frame and by NeRF's two large capacity internal neural networks. Reducing both factors often results in lower quality renders and cannot be done without additional countermeasures. While the current literature has focused on reducing the number of queries [6, 7] and the use of render acceleration structures [8–13], this paper explores how to minimize NeRF's network capacity while limiting the quality degradation.

Solution. We introduce DistillNeRF, a data-efficient method to initialize and breed lower capacity NeRF models using Knowledge Distillation (KD). DistillNeRF uses only one network for both coarse and fine steps, naturally reducing the number of network queries and diving memory usage by half. Our method relies on extracting a priority occupancy voxel grid from a pretrained model, the teacher, to generate better data samples for training a smaller model, the student, and reduce the quality degradation due to the lower weight capacity.

Findings. We evaluate DistillNeRF on the Synthetic Blender NVS benchmark proposed in the original work [5] by distilling three flavors of NeRF students we named TinyNeRF, MicroNeRF, and NanoNeRF in decreasing order of depth and wideness. Our approaches reduces the quality degradation due to the lower capacity network on Mean Squared Error (MSE), Peak Signal to Noise Ratio (PSNR), and Structural Similarity (SSIM), using half the memory.

Contributions.

- A data-efficient method, DistillNeRF, to initialize and breed lower capacity NeRF models using KD. We observe reduced memory usage and lower quality degradation compared to previous initialization methods.
- A study comparing three low capacity flavours of NeRF architectures we named TinyNeRF, MicroNeRF, and NanoNeRF against the vanilla training procedure, a meta-learning initialization scheme called Reptile on a Blender Synthetic Dataset, and an ablation study to justify our design choices.
- An open-source C++ renderer using LibTorch, Cuda kernels, and OpenGL textures to limit memory transfers between the RAM and the VRAM.

Implication. Prior work focused on the acceleration of the rendering pipeline via the use of proxies such as acceleration structures, reformulation of the ray sampling strategies, and the radiance representation.

DistillNeRF is orthogonal to these contributions and focuses on reducing the size of the NeRF inner MLP. Our method can be applied in conjunction with sample reduction methods and acceleration structures. The reduction of the neural network size naturally accelerates the inference and rendering.

Such improvements would result in the use of neural NVS for real-time applications on consumer-grade hardware, the emergence of new artistic 3D workflows, and faster creative iterations.

Reproduction. For replicability and transparency, we published our implementation and experimentation at `https://github.com/after_publication`.

2 Related Work

Novel View Synthesis. Computer Graphics (CG) methods are defined as the set of formulations used to represent a scene geometry and rendering technics used to generate their image representation. As such, Novel View Synthesis (NVS) methods are categorized by these criteria.

Mesh [14–17] and Point-Cloud [18–21] methods relies on preprocessing steps such as Structure from Motion (SfM) or RGB-D sensors to reconstruct the scene geometry from a set of multiview images. Multi-Plane Images (MPI) approaches [22–25] represents the scene as a stack of image layers allowing high quality renders in low range viewpoint displacements. Voxel-based pipelines [26–31] make use of a voxel occupancy grid as a low resolution scene representation. The use of voxels for NVS is often limited by the lack of structural definition due to a lack of resolution.

These four methods are often referred to as classical frameworks and present limitations that implicit neural representation can alleviate. Implicit differentiable rendering methods [32, 5, 33–36] performs optimization in the image space by formulating the rendering pipeline as a differentiable rasterization process.

Such approaches require accurate geometry masks and cannot handle thin, transparent, nor semi-transparent objects. In contrast, the use of differentiable volumetric rendering in conjunction with Signed Distance Fields (SDF) [37, 38, 34] or Neural Radiance Fields (NeRF) [5] to represent the scene geometry does not require the use of any mask and can handle such complex objects. The use of Alpha-Blending enables those methods to recover thin, transparent, and semi-transparent materials.

Neural implicit representations and volumetric rendering are at the core of the current state of the art in geometric Computer Vision (CV) taks such as 3D shape reconstruction [39–41], scene relighting [42–45], object, human, and camera pose estimation [46–48], and more.

Training Stability. Recent contributions and application of NeRF have exposed the limitations of its vanilla implementation.

The NeRF internal neural network is sensible to initialization and may suffer from instabilities during training on certain scenes resulting in exploding or vanishing values. In their work [49], Tancik et al. proposed the use of a meta-learning approach called Reptile [50] to learn better initializations.

In Mip-NeRF [51], Barron et al. not only present a ray casting formulation to reduce aliasing when the camera position differs from the dataset but also propose a set of stabilization tricks. They replace the softplus activation with a shifted one, the sigmoid activation a widened one to avoid common failure modes and yield smoother optimizations.

Fast Rendering. Vanilla NeRF is not suited for realtime application (60 Frames per Second (FPS)). In follow-up works, acceleration structures and optimization tricks are used to reduce inference and accelerate rendering.

DeRF [8] splits the scene volume into sections for which a specific NeRF subnetwork is trained on. Similarly, KiloNeRF [10] pushes the concept further by training thousands of tiny Multi Layer Perceptron (MLP) on bounded regions of the scene.

Neural sparse voxel fileds [9] use a sparse voxel grid fitted to the scene geometry to cache and compute ray voxel intersections, thus avoiding sampling empty volumes. Hadman et al. [11] propose a similar approach by baking the NeRF model into an acceleration structure in combination with the use of differed rendering to accelerate the rendering performances. Yu et al. [12] build a plenoptic octree fitted onto the scene content and use spherical harmonics decomposition to encode the scene appearance.

DONeRF [6] trains a depth oracle in an end-to-end framework with the NeRF model to predict the surface location when raycasting. The number of samples per ray is then chosen accordingly to this prediction to reduce the amount of evaluation associated with empty space.

Extended Applications. Since publication, NeRF has gained attraction in the field of 3D scene reconstruction and NVS, and has been adapted to a multitude of use-cases.

NeRF++ [52] extends its applications to 360 captures of large-scale unbounded scenes, NeRF-W [53] to in the wild monument photographies, and NeRF in the Dark to High Dynamic Range (HDR) using noisy raw images. Others have appplied NeRF to text guided generation [39], free view realistic facial animation [54] and controllable human apearance and pose articulations [47].

Each of these contributions inherently suffers from NeRF vanilla formulation. Faster training, inference, and stability would extend their usage to real-world and real-time applications.

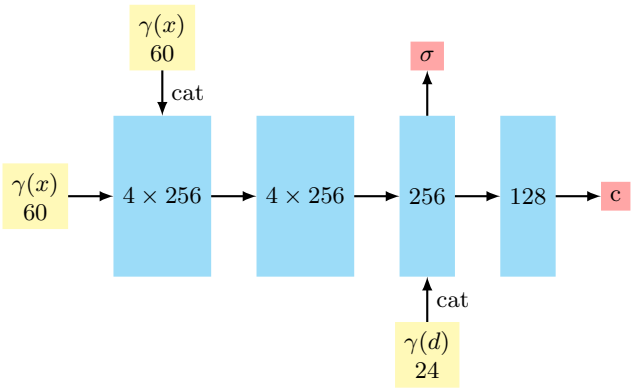

Fig. 2: Schematic of the NeRF internal MLP architecture. The first section of the network is in charge of learning valuable feature representations of the scene content given an positional encoded position vector $\gamma(x)$ with residual connections. These features are then used to regress a density value σ and a RGB radiance c given a positional encoded view direction $\gamma(d)$.

3 Background

Geometry Representation. Neural Radiance Fields (NeRF) encodes a scene content given multiple views into a neural network parameterized by θ, f_θ. The network takes a 5D input vector (x, y, z, θ, ϕ) composed of a 3D position vector and a view direction to encode view-dependent visual properties such as reflection and specularity. Both the position and view direction are encoded using a deterministic positional encoding γ to recover high-frequency details as demonstrated in a follow-up work by the authors [55]. NeRF's internal neural network is a simple 8 layer deep and 256 neurons wide Multi Layer Perceptron (MLP). A complete depicture of the architecture is shown in Figure 2.

Volumetric Rendering. NeRF uses a classical volume rendering pipeline with alpha blending to render a frame. Rays are cast from the camera origin towards the center of each pixel. To this end, 64 positions are uniformly sampled along

the ray direction. This step is referred to as the coarse step, which is part of the hierarchical sampling scheme proposed by Mildenhall et al. [5]. Every sample x_i is evaluated given the ray direction d by a coarse MLP f_c to output the corresponding density σ_i and the RGB radiance c_i.

$$(\sigma_i, c_i) = f_\theta(\gamma(x_i), \gamma(d)) \qquad 1 \leq i \leq 64 \tag{1}$$

The ray evaluations are then composed using a differentiable formulation of alpha blending to output the coarse pixel color $\hat{c}_c$.

$$\hat{c} = \sum_{i=1}^{64} T_i \alpha_i c_i \qquad T_i = \prod_{j=1}^{64} 1 - \alpha_j \qquad \alpha_i = 1 - exp(\sigma_i \delta_i) \tag{2}$$

T describes the transmittance and δ the distance between adjacent sample positions $\delta_i = ||x_{i+1} - x_i||$. The coarse evaluation is finally used to compute a probability density function along the ray to generate 64 new fine samples that are more likely to be close to the scene geometry. A second network, the fine network f_f is used to evaluate both coarse and fine samples to obtain a refined evaluation of the final pixel color $\hat{c}_f$.

In total $(64 + 128) \times H \times W$ network evaluations are performed during the generation of one frame, thus $122M$ evaluation when considering a 800×800 image. Only the fine network is used during inference.

Optimization. NeRF optimizes its coarse and fine networks in an end-to-end fashion by computing an L_2 loss in image space against ground truth.

$$\mathcal{L} = \frac{1}{P} \sum_{p=1}^{P} ||c_p - \hat{c}_{cp}||_2 + ||c_p - \hat{c}_{fp}||_2 \tag{3}$$

4 Proposed Method

Our proposed method, DistillNeRF, aims at breeding smaller capacity Neural Radiance Fields (NeRF) Multi Layer Perceptron (MLP)s by initializing them from pretrained high capacity models. We propose to use Knowledge Distillation (KD) to transfer the knowledge captured by the weights of a large capacity network called the teacher to a low capacity network called the student. During this KD phase, the student is trained on samples produced by the teacher network to match its output distribution. The quality of the student is thus bound by the teacher's examples quality.

This section describes how the teacher model is trained, how to generate a student candidate, and how to sample good teacher examples.

4.1 Teacher Model

The teacher model follows the same architecture as the vanilla NeRF described in Figure 2 with modified activation functions as proposed by Barron et al. in Mip-NeRF [51]. The softplus activation is replaced by a shifted version, Equation 4, and the sigmoid activation by a widened version, Equation 5.

$$\text{softplus}(x) = log(1 + exp(x - 1)) \tag{4}$$

$$\text{sigmoid}(x) = \frac{1 + 2\epsilon}{1 + exp(-x))} - \epsilon \qquad \epsilon = 1e - 3 \tag{5}$$

Contrary to the original work [5], we train the same network f_θ for both coarse and fine steps. Using only one network during training reduces memory usage by half and network evaluation from $64 + 128 = 192$ to 128 per pixel. Training is thus less resource-intensive and faster.

Table 1: Models architectures. The width describes the number of neurons in each layer of the NeRF internal neural network, the depth the number of hidden layers, and residual the presence or absence of a residual connection in the model. The models are displayed in descending order of size (MB). Smaller models naturally demonstrate higher render frame rates (FPS). However, low capacity results in perceptual degradation (PSNR). Results are averaged over the 8 scenes of the Blender Synthetic Dataset.

Model	**Width**	**Depth**	**Residual**	**PSNR ↑**	**FPS ↓**	**MB ↓**
NeRF	256	8	✓	**30.01**	0.28	2.38
TinyNeRF	128	4	✓	26.37	0.94	0.37
MicroNeRF	64	4	×	24.09	1.71	0.10
NanoNeRF	32	2	×	23.45	**2.16**	**0.03**

4.2 Student Model

The architecture of NeRF internal neural network is a sample MLP described in Figure 2. Reducing its inference time requires either a reduction of capacity, the number of parameters θ, or the use of faster operation routes in the network graph.

DistillNeRF generates student networks by reducing three hyperparameters: the MLP width, depth, and an eventual residual connection. Following this recipe, we propose three new flavors of NeRF: TinyNeRF, MicroNeRF, and NanoNeRF in decreasing order of size. Their respective hyperparameters are displayed in Table 1. The parameters are chosen to reflect a wide range of student

variations. Smaller models naturally benefit from faster inference and rendering time. NeRF renders 400×400 images at 0.28 Frames per Second (FPS) compared to 2.16 FPS for NanoNeRF without the use of any additional acceleration technique.

4.3 Knowledge Distillation

The student candidates are initialized using KD. Their weights are distilled from the pretrained NeRF teacher model. The teacher examples are randomly sampled from a Bounding Box (BBox) centered on the scene geometry. The BBox is described by a set of six parameters $((x_a, x_b), (y_a, y_b), (z_a, z_b))$, two for each axis initialized to a value of $(-20, 20)$. Random positions and direction vectors are then sampled inside this structure.

The student models are trained to match the teacher's predictions. We optimize the student's parameters to minimize a mixture of L_2 losses for density σ and radiance c. The alpha blending values α are computed instead of the density output σ using $\delta = 3.125e-2$, a value close to the ones encountered during training. This computational change, introduced by Reiser et al. in KiloNeRF [10], reduces the emphasis on small differences between big densities.

$$\mathcal{L}_{KD} = \frac{1}{2}(||\hat{c}_t - \hat{c}_s||_2 + ||\hat{\alpha}_t - \hat{\alpha}_s||_2) \tag{6}$$

Fig. 3: Visualization of the occupancy voxel grid slices normalized between 0 and 1 for a vanilla NeRF model trained on the Lego scene from the Blender Synthetic Dataset.

4.4 Priority Occupancy Voxel Grid

NeRF learns to handle empty space in the early stage of its training process. Thus, we optimize the data-efficiency of our distillation process by introducing a priority occupancy voxel grid to emphasize non-empty space. The occupancy scores act as a rough description of the scene content distribution. An example visualization is displayed in Figure 3.

To fill this grid, the BBox is first refined to make it fit the geometry bounds more tightly. The structure is turned into a $256 \times 256 \times 256$ voxel grid with 256 bins on each axis. Each voxel is then filled with a 32 samples Monte Carlo

approximation of the mean density σ evaluated through the teacher network. The grid is normalized to values between 0 and 1. At this stage of the process, the 3D grid cells contain scores describing the occupancy of the scene. The BBox is finally tightened to the closest cell with a score greater or equal to a selected threshold $\tau = 1e - 2$.

The exact same process is used to generate a $16 \times 16 \times 16$ low-resolution priority occupancy voxel grid with 16 bins. The probability score of each cell is clamped to a minimum value of 0.2 to avoid forgetting empty space during the distillation process. During KD, the position vectors are generated by drawing random cells taking into account their priority, their score, and randomly offset by the voxel size. Finally, the ray directions are generated by sampling a normalized unit sphere.

5 Implementation

DistillNeRF is implemented using the PyTorch Python library. While this approach allows fast prototyping and training, a naive implementation is not suited for production and real-time applications. This sections describes a set of strategies we employ to reduce inference time and accelerate volume rendering.

Inference. For inference, we reduce the Multi Layer Perceptron (MLP) weights to half-precision, FP16. This precision reduction has the benefit of halving memory usage at training and inference time. The model is then exported as a TorchScript module to be used in a C++ environment thanks to the LibTorch API. The batch size is also maxed out during inference, $8,192$ on an Nvidia GeForce RTX 3090.

Rendering. We also implement a set of specialized Cuda kernels to handle the volumetric rendering process, including the ray casting, the alpha blending computation, and accumulation. The final pixel colors are then written into Cuda memory to the VRAM and bound to an OpenGL 2D texture. The texture is finally projected on a quad fitting the screen. Every computation is done on the GPU to limit RAM to VRAM memory transfers and maximize utilization.

6 Evaluation Setup

Data. We evaluate and train our method on the Blender Synthetic Dataset proposed by Neural Radiance Fields (NeRF)'s authors [5]. The dataset gathers a set of 8 scenes: chair, drums, ficus, hotdog, lego, materials, mic, and ship, each containing 100 views of the scene for training, 100 for validation, and 200 for testing. The scenes are rendered using the Blender Cycles ray tracing engine, and the views are sampled from a centered hemisphere surface with the view direction targeting the center. The rendered images are of size 800×800 and contain various challenging materials, including thin, transparent, and semi-transparent objects. The generated images are reduced to 400×400 using nearest pixel

values to make the training of every model variation and initialization methods affordable.

Metrics. The NeRF literature reports evaluations with a set of objective metrics such as the Mean Squared Error (MSE), the Peak Signal to Noise Ratio (PSNR), Structural Similarity (SSIM) [56] and the Learned Perceptual Image Patch Similarity (LPIPS) [57]. The MSE evaluates the mean squared distance of the generated and the target image at a pixel level. The PSNR is derived from the MSE metric and measures the ratio between the maximum possible power of the image and the power of corrupting noise affecting the fidelity of its approximation. The SSIM measures the similarity between a pair of images at different window sizes. The LPIPS makes use of a pretrained foundation model, AlexNet, and measures the perceptual feature differences between the two images.

In their contribution, Barron et al. [51], demonstrate a strong correlation between the LPIPS and SSIM. The LPIPS is not measured in this paper to limit redundancy.

We also report the models' sizes in MB and their rendering performance in Frames per Second (FPS) including the ray generation, casting, the model inference, the ray radiance and alpha blending aggregation.

Baseline. DistillNeRF is an initialization method used to breed smaller NeRF models for faster inference. In this setup, we evaluate our method against two baselines, the vanilla NeRF [5], and the meta-learning Reptile initialization [49]. NeRF, TinyNeRF, MicroNeRF, and NanoNeRF are benchmarked on the three methods.

Training. The models are trained using 64 coarse and 64 fine samples with half-precision enabled. We optimize NeRF using the same regime as the original implementation for 20 epochs using the Adam optimizer with a $5e-4$ learning rate which is decayed to $5e-6$ over time using a log scheduler. Before training, the model's weights are initialized for 1 epoch with 16 samples using the SGD optimizer with a $5e-2$ learning rate for Reptile, and $100,000$ steps using the Adam optimizer with a $5e-4$ learning rate for Knowledge Distillation (KD). The training hyperparameters are chosen to match the initialization time required for both methods.

Measurements. Training and inference measurements are realized on a modern station equipped with an AMD Ryzen 9 5900X 12-Core Processor, 32 Go of DDR4 RAM, and an NVidia GeForce RTX 3090 with 24 Go of VRAM.

7 Results

We quantitatively in Table 2 and qualitatively in Figure 5, 6, and 4 show that our initialization method DistillNeRF outperforms the meta-learning Reptile initialization in Mean Squared Error (MSE), Peak Signal to Noise Ratio (PSNR), and Structural Similarity (SSIM). We also provide an ablation study in Table 3 to justify our current design strategy.

Table 2: Benchmark on Blender Synthetic Dataset re-scaled to 400 × 400. Results are averaged on the 8 scenes of the dataset. Our approach, highlighted in light gray, improves the performances on MSE, PSNR, and SSIM compared to ReptileNeRF for similar training regimes and presents lower quality degradation on smaller models.

Model	Train (h:m:s) ↓	MSE ↓	PSNR ↑	SSIM ↑
NeRF	**02:44:58**	1.24e-03	30.01	0.94
ReptileNeRF	03:59:56	1.17e-03	30.90	**0.95**
DistillNeRF	03:02:49	**1.15e-03**	**31.02**	**0.95**
TinyNeRF	**01:08:40**	2.15e-03	26.37	0.91
ReptileTinyNeRF	01:35:57	1.98e-03	28.29	0.91
DistillTinyNeRF	01:25:46	**1.96e-03**	**28.39**	**0.92**
MicroNeRF	**00:39:51**	2.98e-03	24.09	0.87
ReptileMicroNeRF	00:52:24	2.80e-03	26.51	0.88
DistillMicroNeRF	00:56:55	**2.63e-03**	**26.82**	**0.89**
NanoNeRF	**00:32:27**	3.97e-03	23.45	0.85
ReptileNanoNeRF	00:40:56	3.87e-03	25.01	0.85
DistillNanoNeRF	00:49:37	**3.53e-03**	**25.36**	**0.86**

Benchmark. For similar training regimes and approximately the same training duration, our initialization method DistillNeRF outperforms the Reptile meta initialization procedure on average on MSE, PSNR and SSIM on the Blender Synthetic Dataset.

Table 3: Ablation study on the Blender Synthetic Dataset hot dog scene rescaled to 400 × 400. The table studies the implication of the occupancy voxel grid used of importance sampling and its refinement. The ablation is realised on the most challenging NeRF variation, NanoNeRF.

Model	MSE ↓	PSNR ↑	SSIM ↑
NanoNeRF + KD	1.41e-3	28.66	0.92
NanoNeRF + KD + Voxel Grid	1.21e-3	29.25	0.92
NanoNeRF + KD + Voxel Grid + Refined	**1.05e-3**	**29.89**	**0.93**

Ablation. The ablation study evaluates the impact of our design choices. We report the MSE, PSNR and SSIM on the Blender Synthetic hot dog scene for NanoNeRF, our most challenging model variation. The results shows that the

use of a well fitted occupancy voxel grid for importance sampling improves the standard Knowledge Distillation (KD) strategy.

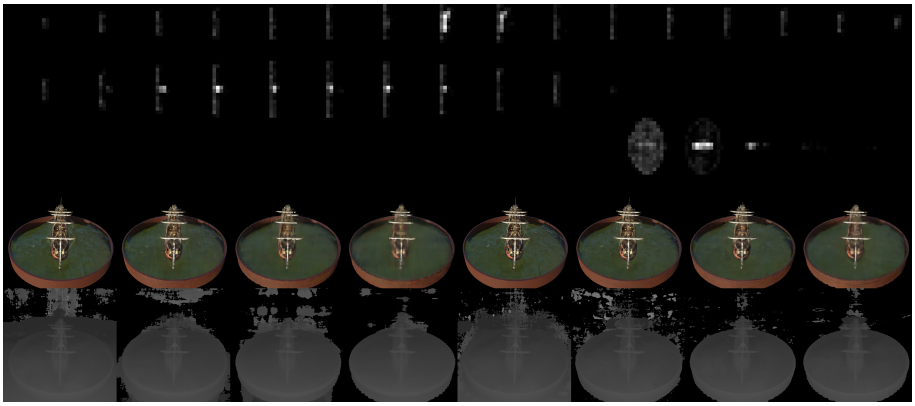

Fig. 4: Renders of the Blender Synthetic Dataset ship scene. The top row shows the occupancy voxel grid, the middle are renders from NeRF variations, and the bottom are their corresponding depth maps. The models used are NeRF, TinyNeRF, MicroNeRF, NanoNeRF, DistillNeRF, DistillTinyNeRF, DistillMicroNeRF, and DistillNanoNeRF respectively from left to right.

8 Conclusion

Our work DistillNeRF addresses the need for faster rendering performances in Neural Radiance Fields (NeRF) methods. While the current state of the art has focused on reducing the number of sample evaluations performed to train such models, we explored the reduction of the inner Multi Layer Perceptron (MLP) size. Smaller models naturally exhibit faster inference but often result in a quality drop. We demonstrate that the use of Knowledge Distillation (KD) in conjunction with an occupancy voxel grid as a proxy for the importance-sampling of teacher examples reduces the perceptual loss induced by a small capacity network. Our method can be combined with other approaches such as acceleration structure caching, sampling oracles, and others to allow real-time rendering at a smaller footprint.

Future Work The use of KD for initialization requires training a teacher network which constitutes a consequent overhead. In future work, we want to explore the use of other proxies such as the information contained in the training images to initialize NeRF approaches at a smaller cost.

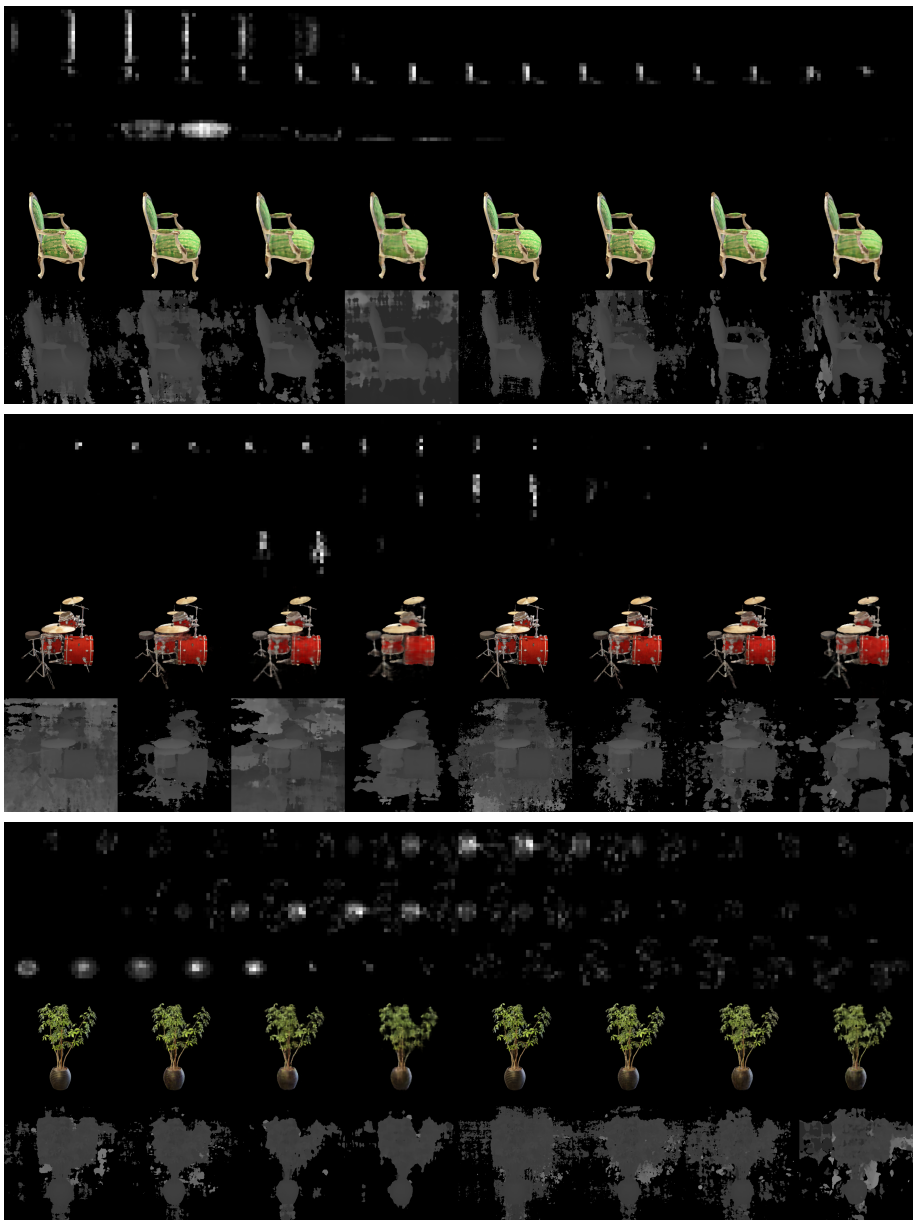

Fig. 5: Renders of the Blender Synthetic Dataset chair, drums and ficus scenes. The top row shows the occupancy voxel grid, the middle are renders from NeRF variations, and the bottom are their corresponding depth maps. The models used are NeRF, TinyNeRF, MicroNeRF, NanoNeRF, DistillNeRF, DistillTinyNeRF, DistillMicroNeRF, and DistillNanoNeRF respectively from left to right.

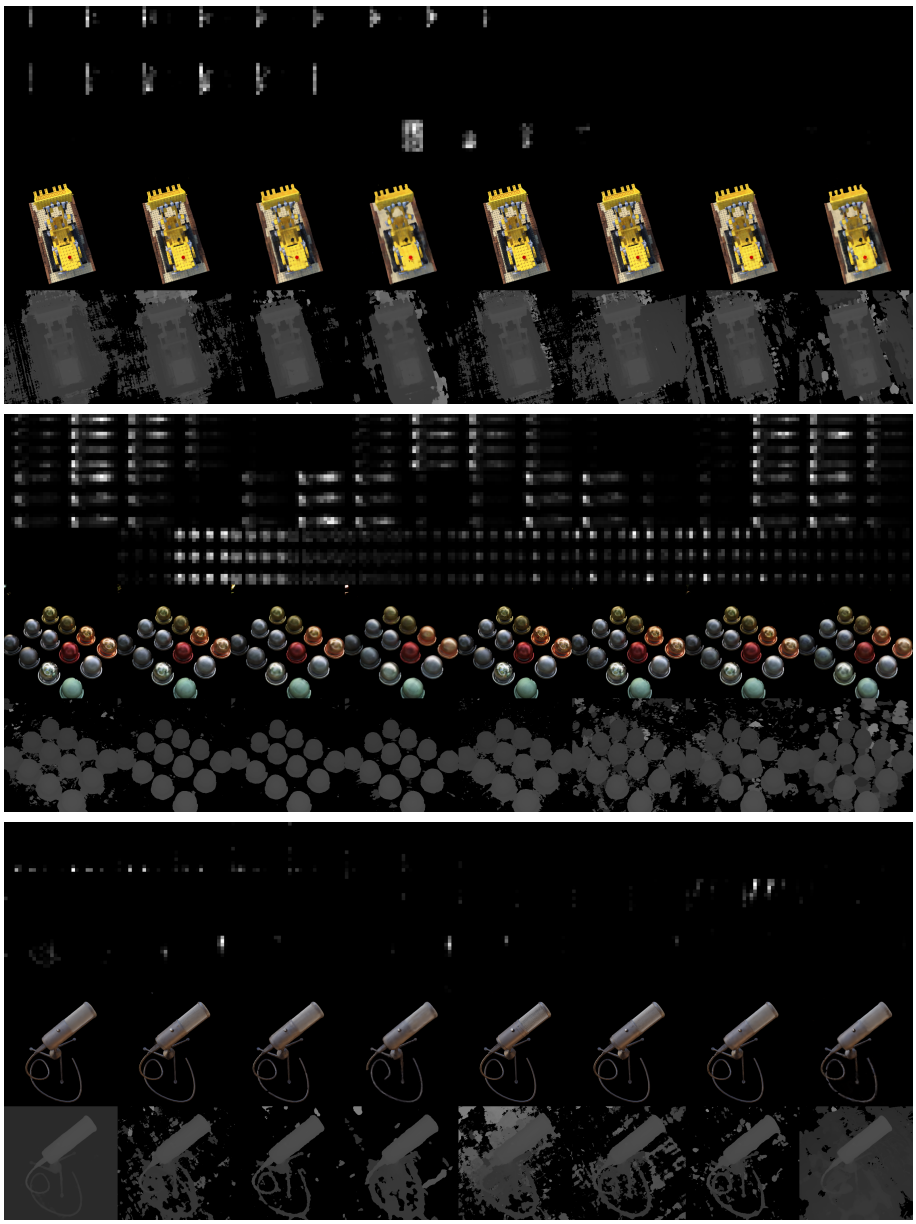

Fig. 6: Renders of the Blender Synthetic Dataset lego, materials and mic scenes. The top row shows the occupancy voxel grid, the middle are renders from NeRF variations, and the bottom are their corresponding depth maps. The models used are NeRF, TinyNeRF, MicroNeRF, NanoNeRF, DistillNeRF, DistillTinyNeRF, DistillMicroNeRF, and DistillNanoNeRF respectively from left to right.

References

1. Li, T., Slavcheva, M., Zollhoefer, M., Green, S., Lassner, C., Kim, C., Schmidt, T., Lovegrove, S., Goesele, M., Lv, Z.: Neural 3d video synthesis. arXiv e-prints (2021) arXiv–2103
2. Zhang, J., Liu, X., Ye, X., Zhao, F., Zhang, Y., Wu, M., Zhang, Y., Xu, L., Yu, J.: Editable free-viewpoint video using a layered neural representation. ACM Transactions on Graphics (TOG) **40** (2021) 1–18
3. Gafni, G., Thies, J., Zollhofer, M., Nießner, M.: Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 8649–8658
4. Xu, H., Alldieck, T., Sminchisescu, C.: H-nerf: Neural radiance fields for rendering and temporal reconstruction of humans in motion. Advances in Neural Information Processing Systems **34** (2021)
5. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., eds.: Computer Vision – ECCV 2020, Cham, Springer International Publishing (2020) 405–421
6. Neff, T., Stadlbauer, P., Parger, M., Kurz, A., Mueller, J.H., Chaitanya, C.R.A., Kaplanyan, A., Steinberger, M.: Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. arXiv preprint arXiv:2103.03231 (2021)
7. Arandjelović, R., Zisserman, A.: Nerf in detail: Learning to sample for view synthesis. arXiv preprint arXiv:2106.05264 (2021)
8. Rebain, D., Jiang, W., Yazdani, S., Li, K., Yi, K.M., Tagliasacchi, A.: Derf: Decomposed radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 14153–14161
9. Liu, L., Gu, J., Lin, K.Z., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. arXiv preprint arXiv:2007.11571 (2020)
10. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. arXiv preprint arXiv:2103.13744 (2021)
11. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. arXiv preprint arXiv:2103.14645 (2021)
12. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. arXiv preprint arXiv:2103.14024 (2021)
13. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. arXiv preprint arXiv:2103.10380 (2021)
14. Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M.: Unstructured lumigraph rendering. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '01, New York, NY, USA, Association for Computing Machinery (2001) 425–432
15. Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '96, New York, NY, USA, Association for Computing Machinery (1996) 11–20
16. Waechter, M., Moehrle, N., Goesele, M.: Let there be color! large-scale texturing of 3d reconstructions. In: ECCV. (2014)

17. Wood, D.N., Azuma, D.I., Aldinger, K., Curless, B., Duchamp, T., Salesin, D.H., Stuetzle, W.: Surface light fields for 3d photography. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '00, USA, ACM Press/Addison-Wesley Publishing Co. (2000) 287–296
18. Aliev, K.A., Sevastopolsky, A., Kolos, M., Ulyanov, D., Lempitsky, V.: Neural point-based graphics. In Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., eds.: Computer Vision – ECCV 2020, Cham, Springer International Publishing (2020) 696–712
19. Riegler, G., Koltun, V.: Free view synthesis. In: European Conference on Computer Vision, Springer (2020) 623–640
20. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
21. Song, Z., Chen, W., Campbell, D., Li, H.: Deep novel view synthesis from colored 3d point clouds. In Vedaldi, A., Bischof, H., Brox, T., Frahm, J., eds.: Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIV. Volume 12369 of Lecture Notes in Computer Science., Springer (2020) 1–17
22. Flynn, J., Broxton, M., Debevec, P., DuVall, M., Fyffe, G., Overbeck, R., Snavely, N., Tucker, R.: Deepview: View synthesis with learned gradient descent. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 2367–2376
23. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Trans. Graph. **38** (2019)
24. Srinivasan, P.P., Mildenhall, B., Tancik, M., Barron, J.T., Tucker, R., Snavely, N.: Lighthouse: Predicting lighting volumes for spatially-coherent illumination. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 8080–8089
25. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. arXiv preprint arXiv:1805.09817 (2018)
26. Kutulakos, K., Seitz, S.: A theory of shape by space carving. In: Proceedings of the Seventh IEEE International Conference on Computer Vision. Volume 1. (1999) 307–314 vol.1
27. Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., Sheikh, Y.: Neural volumes: Learning dynamic renderable volumes from images. ACM Trans. Graph. **38** (2019)
28. Penner, E., Zhang, L.: Soft 3d reconstruction for view synthesis. ACM Trans. Graph. **36** (2017)
29. Seitz, S., Dyer, C.: Photorealistic scene reconstruction by voxel coloring. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. (1997) 1067–1073
30. Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhofer, M.: Deepvoxels: Learning persistent 3d feature embeddings. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 2437–2446
31. Szeliski, R., Golland, P.: Stereo matching with transparency and matting. In: Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271). (1998) 517–524

32. Liu, S., Chen, W., Li, T., Li, H.: Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. arXiv preprint arXiv:1901.05567 (2019)
33. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2020)
34. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 165–174
35. Park, K., Mousavian, A., Xiang, Y., Fox, D.: Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. (2020) 10710–10719
36. Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Basri, R., Lipman, Y.: Multiview neural surface reconstruction by disentangling geometry and appearance. arXiv preprint arXiv:2003.09852 (2020)
37. Chibane, J., Mir, A., Pons-Moll, G.: Neural unsigned distance fields for implicit function learning. arXiv preprint arXiv:2010.13938 (2020)
38. Liu, S., Zhang, Y., Peng, S., Shi, B., Pollefeys, M., Cui, Z.: Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 2019–2028
39. Jain, A., Mildenhall, B., Barron, J.T., Abbeel, P., Poole, B.: Zero-shot text-guided object generation with dream fields. arXiv preprint arXiv:2112.01455 (2021)
40. Jang, W., Agapito, L.: Codenerf: Disentangled neural radiance fields for object categories. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2021) 12949–12958
41. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2019) 2304–2314
42. Boss, M., Braun, R., Jampani, V., Barron, J.T., Liu, C., Lensch, H.: Nerd: Neural reflectance decomposition from image collections. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2021) 12684–12694
43. Rudnev, V., Elgharib, M., Smith, W., Liu, L., Golyanik, V., Theobalt, C.: Neural radiance fields for outdoor scene relighting. arXiv preprint arXiv:2112.05140 (2021)
44. Srinivasan, P.P., Deng, B., Zhang, X., Tancik, M., Mildenhall, B., Barron, J.T.: Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 7495–7504
45. Zhang, X., Srinivasan, P.P., Deng, B., Debevec, P., Freeman, W.T., Barron, J.T.: Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. arXiv preprint arXiv:2106.01970 (2021)
46. Guo, J., Yang, Z., Lin, X., Zhang, Q.: Template nerf: Towards modeling dense shape correspondences from category-specific object images. arXiv preprint arXiv:2111.04237 (2021)
47. Su, S.Y., Yu, F., Zollhöfer, M., Rhodin, H.: A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. Advances in Neural Information Processing Systems **34** (2021)

48. Yen-Chen, L., Florence, P., Barron, J.T., Rodriguez, A., Isola, P., Lin, T.Y.: inerf: Inverting neural radiance fields for pose estimation. arXiv preprint arXiv:2012.05877 (2020)
49. Tancik, M., Mildenhall, B., Wang, T., Schmidt, D., Srinivasan, P.P., Barron, J.T., Ng, R.: Learned initializations for optimizing coordinate-based neural representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 2846–2855
50. Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999 (2018)
51. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. arXiv preprint arXiv:2103.13415 (2021)
52. Zhang, K., Riegler, G., Snavely, N., Koltun, V.: Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492 (2020)
53. Martin-Brualla, R., Radwan, N., Sajjadi, M.S., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 7210–7219
54. Athar, S., Shu, Z., Samaras, D.: Flame-in-nerf: Neural control of radiance fields for free view face animation. arXiv preprint arXiv:2108.04913 (2021)
55. Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. NeurIPS (2020)
56. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13** (2004) 600–612
57. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 586–595