Computational Visual Media
https://doi.org/10.1007/s41095-022-0327-z Vol. 9, No. 2, June 2023, 401–404

Short Communication

JNeRF: An efficient heterogeneous NeRF model zoo based on Jittor

Guo-Wei Yang[1] (✉), **Zheng-Ning Liu**[2], **Dong-Yang Li**[1], and **Hao-Yang Peng**[1]

Neural radiance fields (NeRFs) for novel-view synthesis have attracted the attention of researchers in computer vision and graphics. Unlike traditional methods using explicit expressions, NeRFs represent a scene as an implicit neural radiance field. When rendering, NeRF queries the color density at every position in the scene through a neural network.

NeRF brings a wide range of possibilities for real-world 3D reconstruction and rendering, but problems remain to be solved. Previous works have improved NeRF's sampling technique, position encoding method, network structure, etc., but these improvements are difficult to be combined as the different modules are not well decoupled. Recent works have significantly sped up the core GPU computation of NeRF, leaving the deep learning framework as a major computational cost. Thus, it has been suggested to replace the frameworks by pure CUDA programs, but this limits maintainability and extendability.

Therefore, we propose JNeRF, a unified, efficient, framework-friendly NeRF model zoo based on Jittor.

1 JNeRF architecture

In this paper, we propose JNeRF, an efficient heterogeneous NeRF model zoo based on Jittor [1]. We have analyzed existing NeRF methods, and summarized the main architecture of NeRF in 7 modules, as shown in Fig. 1. All modules are decoupled from each other so that they can be easily modified or replaced. The methods used by different modules in JNeRF can be changed by modifying configuration files. These modules perform:

Camera pose estimation. For input data lacking camera pose annotations, pose is estimated using some existing camera pose estimation method.

Image feature extraction. This module is optional; it is usually used in few-shot NeRF. It can provide extra prior information to subsequent modules.

Sampling. NeRF generally integrates sample points for rendering by ray marching. Various alternative sampling methods are available to increase rendering speed and improve rendering quality.

Position encoding. The positions of sampled points are further encoded by this module. Compared to using basic positions, position encoding can express a more complex and elaborate geometric structure.

Network structure. Usually, NeRF employs a simple MLP as the network structure, but more sophisticated network structures may bring better results.

Scene representation. Going beyond representing the model in terms of RGB and density, it is possible to use an illumination-decoupled representation, an explicit representation, etc.

3D generation. This module outputs various 3D scenes by using NeRF as part of the generative approach in a GAN.

2 Why JNeRF is efficient

2.1 An example

Instant-NGP [2], proposed in early 2022, is representative of an approach we have implemented using JNeRF. Its hash position encoding method greatly shortens the training time: it can train NeRF to output a preliminary result within 5 s. However, its main code of the original Instant-NGP is written in CUDA, making it difficult for other researchers to build upon it. Instead, our JNeRF implementation is more extensible due to our proposed architecture. However, there is no silver

1 Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: G.-W. Yang, ygw19@mails.tsinghua.edu.cn (✉); D.-Y. Li, lidongyang2001@gmail.com; H.-Y. Peng, phy22@mails.tsinghua.edu.cn.
2 Fitten Tech Co., Ltd., Beijing 100084, China. E-mail: lzhengning@gmail.com.

Manuscript received: 2022-09-19; accepted: 2022-11-21

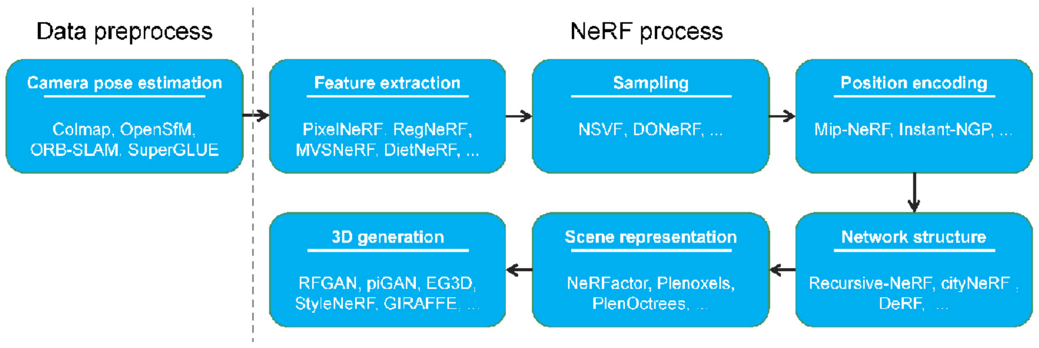
Data preprocess
NeRF process
Camera pose estimation
Colmap, OpenSfM, ORB-SLAM, SuperGLUE
Feature extraction
PixelNeRF, RegNeRF, MVSNeRF, DietNeRF, ...
Sampling
NSVF, DONeRF, ...
Position encoding
Mip-NeRF, Instant-NGP, ...
Network structure
Recursive-NeRF, cityNeRF, DeRF, ...
Scene representation
NeRFactor, Plenoxels, PlenOctrees, ...
3D generation
RFGAN, piGAN, EG3D, StyleNeRF, GIRAFFE, ...

Fig. 1 JNeRF architecture.

bullet. It took us great efforts in optimization to achieve almost the same performance as the original Instant-NGP implementation. Overall, the high efficiency of the JNeRF model zoo can be attributed to: just-in-time compilation, higher computational graph processing efficiency, automatic operator fusion, quantization, etc.

2.2 Just-in-time compilation

Jittor is a just-in-time deep learning framework: unlike other frameworks that require all operators to be precompiled, Jittor can be compiled at runtime. This allows Jittor to make dynamic shapes, such as the number of input rays for NeRF, static, so that the compiler can perform efficient optimizations such as loop unrolling. At the same time, Jittor can adjust the optimization strategy according to the input to achieve a better optimization effect, such as adjusting the dimension of the loop split according to the size of shape.

2.3 Computational graph processing efficiency

Jittor is a framework that focuses on efficiency. Its framework overhead, including computational graph creation, computational graph forwarding, optimization, etc., is less than that of other frameworks such as PyTorch [3]. This overhead is mainly performed on the CPU, so can be run in parallel with the GPU without affecting the training speed (as they take less time than the GPU processing), improving tasks that were slow in previous training approaches. As NeRF approaches have developed, faster training methods have been proposed, such as Instant-NGP and Plenoxels. As the training speed increases, the additional overhead of the framework becomes more significant, and even limits further improvement of training speed. As a consequence, more and more researchers are inclined to implement ideas directly using CUDA, which greatly hinders scientific research.

2.4 Automatic operator fusion

Combining matrix multiplication (matmul) and activation operations has proved efficient in deep learning. JNeRF adds fusion strategies for NeRF tasks to Jittor. Matmul usually uses an operator from the blas library, such as cublasGemm from cublas, so operator fusion fuses the matmul operator and activation operator in the computation graph. However, fusion in this way results in useless memory operations, because activation can be directly applied to the accumulator matrix without redundant procedures. Jittor combines the matmul and activation operation into a single CUDA operator, as in the implementation of Instant-NGP. Jittor uses NVIDIA's warp level matrix operations (wmma) to introduce tensor cores into the matmul operation and applies the activation function to the results of matmul in each thread block to achieve fully-fused performance. Jittor uses the int4 data type to store the weight matrices to further accelerate the whole process. Moreover, Jittor arranges the matrix memory schemes to fit the wmma's memory layout, thereby avoiding the transposition operation in Instant-NGP. Using the fully-fused matmul operator can lead to an 18.7% improvement in speed while maintaining the same result quality.

2.5 Quantization

Some NeRF models like Instant-NGP can also meet the accuracy requirements when using float16 for parameters during training. Jittor supports float16, and mixed float32 and float16 computation. This mixed precision support allows these models to be trained faster with little loss of final quality. In training the Instant-NGP model, we store the MLP network parameters as float16 and, use float16 for rendering calculations, giving a 30.4% improvement in training speed over using float32.

3 JNeRF hardware adaptation

While Nvidia GPUs are the most popular backends for both deep learning frameworks and NeRF models, other accelerator hardware can also be used to train and deploy NeRF. The AMD ROCm platform is one of the most successful GPU computing solutions apart from Nvidia CUDA, and JNeRF provides a heterogeneous framework adapted to both Nvidia CUDA and AMD ROCm backends for application to a range of production environments. Support for other hardware is also planned.

ROCm is open-source, and unlike CUDA, ROCm is designed to support heterogeneous hardware including both Nvidia and AMD GPUs. The eco-system provided by ROCm is similar to that for CUDA (MIOpen vs. cuDNN, rocBLAS vs. cuBLAS, rocPRIM vs. CUTLASS, and RCCL vs. NCCL) so that JNeRF may readily be accelerated by AMD GPUs.

It requires extra effort to develop and maintain multiple versions of hardware-specific source code for the same purpose. Also, it is difficult to customize JNeRF for researchers unfamiliar with ROCm. Therefore, we propose a Jittor JIT extension, the *hardware adaptation translator*. Rather than generating both CUDA and ROCm code simultaneously, the JIT extension generates ROCm source code from existing CUDA code so that the ROCm version can benefit from existing CUDA optimization techniques. This translator not only translates function calls and computing kernels from CUDA to ROCm, but also carries out further optimization according to the AMD hardware details, including memory capacity, memory bandwidth, parallel algorithms, etc.

4 Experiments

We have evaluated our JNeRF implementation of Instant-NGP on the Synthetic-NeRF dataset [4], comparing it to the original implementation. We used 200 images for training, and 200 images for testing, measuring the PSNR for each scene after 5 min of training on an RTX3090. Table 1 shows that the quality from the JNeRF-implementation surpasses the original; see also Fig. 2, giving results for the Mic and Lego scenes. Our implementation provides sharper detail and more accurate reflections than the original.

JNeRF Instant-NGP can be trained quickly. Figure 3 shows rendering results after training for 0.5 s, 1 s, and 5 s on an RTX3090, the latter being a clear image.

5 Conclusions and future work

JNeRF provides an efficient cross-platform model zoo. However, relatively few models are supported by JNeRF so far; we hope to support further

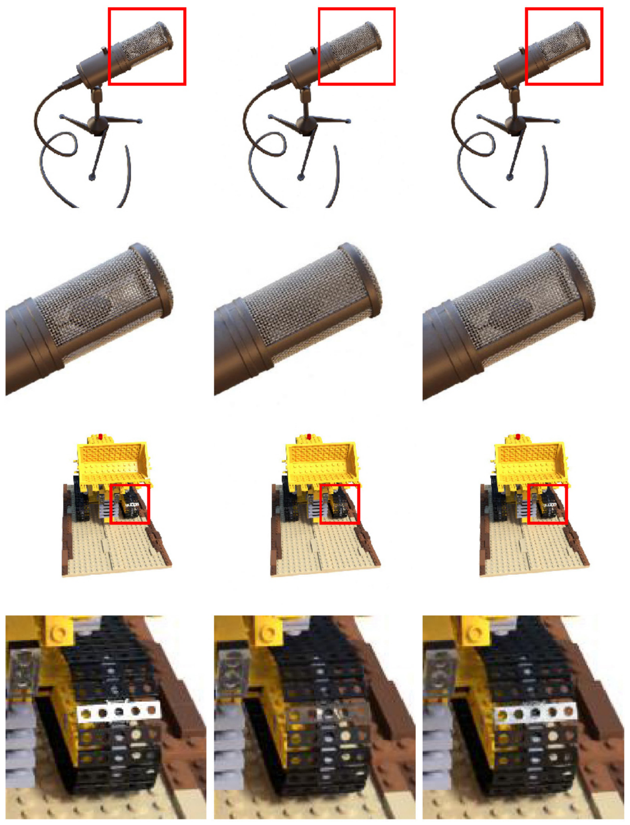
Fig. 2 Renderings of Mic and Lego scenes. Left to right: ground-truth, original implementation, and JNeRF implementation.

Table 1 PSNR (dB) achieved by JNeRF implementation and the original implementation of Instant-NGP

Model	Implementation	Lego	Chair	Drums	Ficus	Ship	Mic	Hotdog	Materials	avg.
Instant-NGP	Original (5 min)	37.08	35.63	**26.86**	33.82	**31.91**	36.91	**37.91**	**30.48**	33.83
Instant-NGP	JNeRF (5 min)	**37.21**	**35.79**	26.61	**34.30**	31.75	**37.64**	37.79	30.37	**33.93**

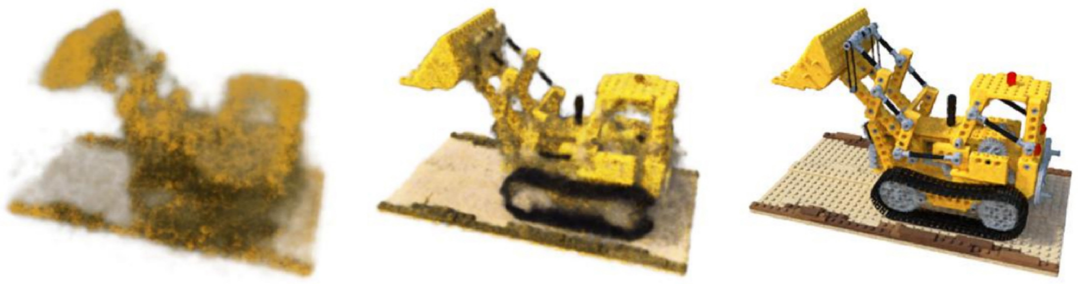

Fig. 3 Rendering by Instant-NGP implemented using JNeRF after training on an RTX3090 for 0.5 s, 1 s, and 5 s, respectively.

representative NeRF works in future, including ones for 3D generation, image features through advanced image backbones, different model representations, etc. We also hope to allow use of JNeRF on various devices, such as mobile platforms.

Acknowledgements

This paper was supported by National Key R&D Program of China (Project No. 2021ZD0112902)

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Hu, S. M.; Liang, D.; Yang, G. Y.; Yang, G. W.; Zhou, W. Y. Jittor: A novel deep learning framework with meta-operators and unified graph execution. *Science China Information Sciences* Vol. 63, No. 12, Article No. 222103, 2020.

[2] Müller, T.; Evans, A.; Schied, C.; Keller, A. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint* arXiv:2201.05989, 2022.

[3] Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An imperative style, high-performance deep learning library. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Article No. 721, 8026–8037, 2019.

[4] Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; Ng, R. NeRF. *Communications of the ACM* Vol. 65, No. 1, 99–106, 2022.

Guo-Wei Yang is a Ph.D. student in the Department of Computer Science and Technology, Tsinghua University, where he also received his B.S. degree in 2019. His research interests include computer graphics, neural rendering, and computer vision.

Zheng-Ning Liu is a research scientist at Fitten Tech Co., Ltd. He received his bachelor and Ph.D. degrees in computer science from Tsinghua University in 2017 and 2022, respectively. His research interests include 3D reconstruction, geometric modeling and processing.

Dong-Yang Li is an undergraduate student at Tsinghua University. His research interests include computer graphics and computer vision.

Hao-Yang Peng is a Ph.D. student in the Department of Computer Science and Technology, Tsinghua University. His research interests include computer graphics and computer vision.