Feasibility Study of Deep Learning-Based Collisionless Gravitational N-Body Numerical Simulations (Postprint)

Authors: Zhao Zicheng, hidden dragon, Dong Xiaobo, Meng Runyu, Zhong Shiyan, Chen Junyi, Xiang Zikun

Date: 2021-06-18T00:00:00+00:00

Abstract

This paper proposes replacing the fast Fourier transform method with deep neural networks for solving the potential energy in the PM-Tree method for collisionless gravitational N-body numerical simulations, to enhance the efficiency of the PM-Tree method and verify the feasibility of accelerating collisionless gravitational N-body numerical simulations via deep learning approaches. Collisionless gravitational N-body numerical simulations hold significant importance for studying the formation and evolution of galaxies, dark matter halos, and large-scale cosmic structures. However, traditional methods for collisionless gravitational N-body numerical simulations are computationally prohibitive for large-scale problems, with the primary computational bottleneck of the commonly used PM-Tree method being the solution for potential energy (solving the Poisson equation). This paper proposes employing deep neural networks to replace traditional methods for accelerating the solution of the Poisson equation. Following extensive architectural adjustments, training, and testing of deep neural network model structures, an overall Encoder-Decoder architecture supplemented with local residual network structures was ultimately adopted. The computational time complexity of solving the Poisson equation with deep neural networks was verified to be ; when tested on the same data, it demonstrates superior speed compared to solutions using the fast Fourier transform method and the finite difference method; under the same sampling rate, it achieves higher accuracy than the fast Fourier transform method; and it exhibits scalability. Therefore, in collisionless gravitational N-body numerical simulations, the employment of deep neural networks can enhance the speed of solving potential energy in the PM-Tree method, thereby effectively improving the overall simulation speed.

Full Text

Feasibility Study of Collisionless Gravitational N-Body Numerical Simulation Based on Deep Learning

Zhao Zicheng[12], Long Qian[1], Dong Xiaobo[1], Meng Runyu[12], Zhong Shiyan[1], Chen Junyi[1], Xiang Zikun[3]

[1]Yunnan Observatories, Chinese Academy of Sciences, Kunming 650216, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]School of Physics, University of Science and Technology of China, Hefei 230026, China

Abstract

This paper proposes replacing the Fast Fourier Transform method with a deep neural network to solve the potential energy in the PM-Tree method for collisionless gravitational N-body numerical simulations, thereby improving the efficiency of the PM-Tree method and verifying the feasibility of using deep learning to accelerate collisionless gravitational N-body simulations. Collisionless gravitational N-body numerical simulations are crucial for studying the formation and evolution of galaxies, dark matter halos, and large-scale cosmic structures. Traditional methods for such simulations are computationally expensive for large-scale problems, with the PM-Tree method' s primary bottleneck being the potential energy calculation (solving the Poisson equation). We propose using deep neural networks to accelerate Poisson equation solving, and after extensive adjustments, training, and testing of various deep neural network architectures, we ultimately selected an Encoder-Decoder structure supplemented with residual network local structures. We verify that the deep neural network achieves $O(N)$ computational time complexity for solving the Poisson equation. Tested on identical data, it is faster than both Fast Fourier Transform and finite difference methods; at equivalent sampling rates, it achieves better accuracy than the Fast Fourier Transform method; and it demonstrates scalability. Therefore, in collisionless gravitational N-body numerical simulations, deep neural networks can accelerate the potential energy solving step in the PM-Tree method, effectively improving overall simulation speed.

Keywords: Poisson equation; gravitational N-body numerical simulation; deep neural network

---

Gravitational N-body simulation is one of the primary methods for studying the formation and evolution of astronomical systems such as galaxies, dark matter halos, and large-scale cosmic structures. Since gravity is a long-range force, for gravitational systems at galactic or larger scales with numerous particles, the motion of member stars (or dark matter particles) can be treated as occurring under a collective gravitational potential without considering interactions

between individual stars (or dark matter particles)—the so-called "collisions." This concept defines collisionless gravitational systems and represents a mean-field approximation. For time-evolution N-body numerical simulations of such systems, we can neglect pairwise gravitational interactions between "N-body" particles and instead only need to compute the common gravitational field (i.e., solve the Poisson equation) and its effect on individual particles—this is collisionless gravitational N-body numerical simulation [**?**]. In recent applications (Gadget-3, Gadget-4), the PM-Tree method is one of the main approaches for collisionless gravitational N-body simulations, and the most time-consuming step in this method is solving for the potential energy (solving the Poisson equation) [**?**]. Consequently, the speed of solving the Poisson equation is a major factor affecting the computational cost of collisionless gravitational N-body simulations.

In specific gravitational N-body simulations, the number of "N-body" particles N does not necessarily correspond to the actual number of stars or dark matter particles; rather, N is determined by simulation accuracy requirements and computational capacity. Traditional collisionless gravitational N-body numerical simulation methods are currently suitable for simulating cosmic systems with particle numbers ranging from 10^6 to 10^{13} [**?**]. Simulating larger-scale cosmic systems requires faster methods. Deep neural networks have proven effective for solving various equations and offer high speed, so we propose using deep neural networks to replace the Fast Fourier Transform method for calculating the time-consuming potential energy solving step (solving the Poisson equation) in the traditional PM-Tree method. For a system with 10^{13} particles, the deep neural network approach could theoretically be approximately 40 times faster for a single potential energy solve, and the cumulative acceleration becomes substantial when iterated throughout the simulation. We demonstrate the effectiveness of deep neural networks for accelerating Poisson equation solving from multiple perspectives, laying the groundwork for future research on accelerating collisionless gravitational N-body numerical simulations. Moreover, due to the scalability of deep neural networks, our proposed model can also be applied to other physical problems requiring Poisson equation solutions.

Gravitational N-Body Numerical Simulation Methods

First, the Particle-to-Particle (PP) method directly calculates Newtonian gravitational forces between particles with computational time complexity of $O(N^2)$ [**?**]. Second, when the system contains many particles, Tree codes developed since the 1980s can accelerate simulations [**?**, **?**]. Tree codes use recursive subtree calculations to eliminate redundant computations, reducing the computational time complexity of gravitational N-body simulations to $O(N \log N)$, with applications including GASOLINE [**?**] and iVINE [**?**]. Third, the Particle Mesh (PM) method: particle position information is first converted to uniform grid density information using the Cloud-In-Cell (CIC) method, then the Poisson equation based on these grids is solved using the Fast Fourier Transform (FFT) method.

This method has computational time complexity of $O(N \log N)$, where N represents the number of grid cells [?]. Fourth, to improve the accuracy of simulated particles within PM method grids, Tree codes are used within grids to calculate inter-particle interactions, known as the PM-Tree method [?]. Since Tree codes run in parallel with each small step of the PM method, and each Tree code computation takes less than or equal to half the time of a PM method substep [?], the PM-Tree method' s computational time complexity is also $O(N \log N)$. The PM-Tree method is currently one of the main methods for collisionless gravitational N-body simulations, with applications such as Gadget-2 [?]. Finally, recent N-body programs (like Gadget-3, Gadget-4) [?] have also implemented the Fast Multipole Method (FMM) [?], which can reduce computational time complexity to $O(N)$ and outperform Tree methods at the same accuracy, though FMM algorithms are more complex, particularly challenging to implement with dynamic time steps, special boundary conditions, and efficient parallelization.

Traditional gravitational N-body numerical simulation methods are too slow for ultra-large-scale particle numbers, necessitating faster approaches. Deep neural networks have achieved remarkable results in many fields [?, ?] and are frequently used for solving equations [?, ?]. Lagaris et al. demonstrated that neural networks can solve ordinary and partial differential equations [?]. Gravitational N-body numerical simulation problems involve massive computational requirements for high-dimensional data, while convolutional neural networks excel at extracting multi-scale features from high-dimensional inputs [?, ?], effectively addressing the curse of dimensionality in numerical PDE solutions [?]. Therefore, we propose using deep convolutional neural networks to replace the Fast Fourier Transform method for faster potential energy solving (solving the Poisson equation) in the PM-Tree method for collisionless gravitational N-body numerical simulations.

1 Related Work

In recent years, several methods using deep neural networks to solve partial differential equations have emerged. The Jeans equation, defined by velocity moments of the collisionless Boltzmann equation, has the same form as the (inviscid) Navier-Stokes equations in fluid dynamics, so research conclusions from solving Navier-Stokes equations in fluid dynamics also apply to gravitational N-body numerical simulations [?]. The steady flow approximation is a Navier-Stokes fluid problem that is difficult to compute and very time-consuming as an iterative process; Guo implemented fast and accurate fitting using convolutional neural networks [?]. Zhu et al. used convolutional neural networks for random porous media flow problems, achieving excellent results with very little data, outperforming Monte Carlo methods [?]. Saakaar et al. demonstrated the effectiveness of convolutional neural networks for flow field prediction problems, with speeds faster than Reynolds Averaged Navier-Stokes methods [?]. Khoo et al. proved that neural networks can concisely solve partial differential equation problems with uncertainties [?]. Adler et al. showed that using deep neural net-

works for iterative solving of ill-posed inverse problems yields significant speed improvements [**?**].

However, these deep neural network methods for solving partial differential equations parameterize the solution operator indirectly and lack robustness regarding mesh size. For gravitational N-body numerical simulations, which involve continuous iteration, simpler and more efficient solution methods are needed. Recently, some methods for directly solving partial differential equations for specific problems have emerged. Wei Tang et al. achieved excellent results solving two-dimensional Poisson equations in image form in the electromagnetic domain [**?**]. Weinan et al. proposed the Deep Ritz method, using deep neural networks to find solutions in the Ritz method for solving partial differential equations, proving that deep neural networks outperform finite difference methods [**?**]. Kaushik et al. demonstrated the effectiveness and convergence of using deep neural network parameterization to solve elliptic partial differential equation problems in infinite-dimensional spaces [**?**]. Smith et al. created EikoNet, proving that deep neural network methods can stably solve Eikonal equations with significant computational speed advantages and broad applicability [**?**].

2 Our Work and Contributions

Combining current deep learning characteristics—high speed, effective solution of high-dimensional problems, and ability to solve partial differential equations—we propose using deep neural networks to replace the Fast Fourier Transform method for solving potential energy (solving the Poisson equation) in the PM-Tree method for collisionless gravitational N-body numerical simulations. This approach could theoretically complete large-scale collisionless gravitational N-body numerical simulations faster and offers good scalability and ease of implementation. Our deep neural network model and corresponding training and testing code are open-sourced on GitHub at: https://github.com/hi76/Poisson-slover-based-on-Encoder-Decoder-Neural-Network.

The earliest direct solution methods for N-body numerical simulations were extremely time-consuming. The mean-field approximation transformed the intractable N-body problem into an approximate statistical problem that was easier to solve but still slow. Later optimization methods emerged, such as the PM method, PM-Tree method, and FMM method. Among these, the PM-Tree method offers advantages in speed and accuracy compared to most other N-body numerical simulation methods, making it one of the current mainstream approaches. The primary time-consuming step in the PM-Tree method is solving for potential energy (solving the Poisson equation). We propose using a deep neural network model to replace the Fast Fourier Transform method for accelerating potential energy solving in the PM-Tree method, thereby verifying the feasibility of deep learning methods for accelerating collisionless gravitational N-body numerical simulations.

3.1 N-Body Numerical Simulation

In physics and astronomy, dynamical simulation of a system composed of particles under certain physical forces is called N-body simulation. Gravitational N-body simulation primarily seeks numerical solutions to the equations of gravitational interaction among N particles. It is a common tool in astrophysics, applied to systems ranging from few-body systems to cosmic large-scale structures, such as the Earth-Moon-Sun system to the Milky Way. The equation for gravitational interaction among N particles is [**?**]:

$$\ddot{\mathbf{r}}_i = -G \sum_{j=1, j\neq i}^{N} \frac{m_j(\mathbf{r}_i - \mathbf{r}_j)}{|\mathbf{r}_i - \mathbf{r}_j|^3}$$

An N-particle gravitational interaction system defines a $6N + 1$-dimensional phase space, where at each moment each particle is determined by N position and velocity vectors. The solution to this equation is a trajectory in phase space. Direct solution is extremely time-consuming, requiring simplified numerical methods.

3.2 Mean-Field Method

The mean-field method is an early approach to N-body numerical simulation. When using statistical methods to describe gravitational N-body problems, the system is described by the N-particle distribution function $f^{(N)}(\mathbf{x}_1, \mathbf{v}_1; \ldots; \mathbf{x}_N, \mathbf{v}_N; t)$, which contains complete information about the particles. When particle correlations are neglected, i.e., $f^{(N)}(\mathbf{x}_1, \mathbf{v}_1; \ldots; \mathbf{x}_N, \mathbf{v}_N; t) = \prod_{i=1}^{N} f(\mathbf{x}_i, \mathbf{v}_i, t)$, this is the mean-field approximation—the collisionless N-body system. In the mean-field approximation, the force on a particle depends on the distribution function rather than pairwise gravitational interactions. We can use the single-particle distribution function $f(\mathbf{x}, \mathbf{v}, t)$ to construct the system' s mean-field description, where $f(\mathbf{x}, \mathbf{v}, t)d\mathbf{x}d\mathbf{v}$ represents the probability of finding a particle in a 6-dimensional phase space volume $d\mathbf{x}d\mathbf{v}$ at position $\mathbf{x}$ with velocity $\mathbf{v}$. The evolution of this distribution function is described by the collisionless Boltzmann equation [**?**]:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f - \nabla_{\mathbf{x}} \Phi \cdot \nabla_{\mathbf{v}} f = 0$$

The total potential energy is the sum of external and internal potentials:

$$\Phi(\mathbf{x}, t) = \Phi_{ext}(\mathbf{x}, t) + \Phi_{int}(\mathbf{x}, t)$$

where $G = 6.670 \times 10^{-11}$ is the gravitational constant and Φ_{ext} is the external potential. This internal potential is the common self-gravitational potential

formed by all N-body particles' gravitational energy, obtained by solving the Poisson equation based on all particles' density distribution $\rho(\mathbf{x}, t)$:

$$\nabla^2 \Phi_{int}(\mathbf{x}, t) = 4\pi G \rho(\mathbf{x}, t)$$

where $\rho(\mathbf{x}, t) = \int f(\mathbf{x}, \mathbf{v}, t) d\mathbf{v}$ is the density distribution corresponding to the distribution function. The mean-field method transforms the intractable N-body problem into an approximate statistical problem that is easier to solve, though its direct computation remains time-consuming, leading to subsequent optimization methods like the PM method [**?**].

3.3 PM-Tree Method

The PM-Tree method is another N-body numerical simulation approach that adds Tree codes within PM method grids for finer simulation. Due to its relatively high speed and accuracy, it is commonly used in recent N-body numerical simulation applications [**?**]. The basic steps in each time step (depending on the simulation time interval and accuracy requirements) can be summarized as [**?**]:

1. First, identify the grids where Tree codes are applied and their particles; all other particles are simulated only by the PM method.
2. Run half of the PM method steps.
3. Calculate the external total potential energy for each Tree code application grid—the sum of potentials from other grids.
4. Apply Tree codes to these grids simultaneously and add the results to the running PM method steps.
5. Calculate the potential energy in the PM method and the corresponding acceleration for each particle.
6. Integrate each Tree into the final steps of the PM method.
7. Update particle velocity and position information, then loop back to step (1).

Since Tree codes run in parallel with PM method steps in the PM-Tree method, and each Tree code computation takes no more than half the time of a PM method substep [**?**], the PM-Tree method' s time consumption is primarily determined by the PM method steps. In the PM method, solving for potential energy is the main computational bottleneck, requiring solution of the Poisson equation. Therefore, the key to accelerating the PM-Tree method lies in accelerating Poisson equation solving. The Fast Fourier Transform method is commonly used to solve Poisson equations in PM-Tree method applications [**?**]. This paper proposes using deep neural networks to replace the Fast Fourier Transform method for solving all Poisson equations in the PM-Tree method, thereby accelerating the PM-Tree method and N-body numerical simulations.

3.4 Numerical Solution of Poisson Equation

Traditional methods for solving Poisson equations include finite element, finite difference, finite volume, and Fast Fourier Transform methods. Numerical solution using computers first requires converting the Poisson equation into algebraic equations—difference equations.

The two-dimensional Poisson equation' s difference equation representation is:

$$\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j} = \rho_{i,j}$$

The left side is essentially a convolution of ϕ with a second-order difference operator D. In computer operations, formula (5) can be simplified as generating a simulated potential distribution matrix ϕ and a second-order difference operator D. The finite difference method solves the Poisson equation by expanding formula (5) into a solvable linear system and solving it with various iterative methods, with computational time complexity of $O(N^2)$ [**?**].

The Fast Fourier Transform (FFT) method uses Fourier transforms to convert convolution in the real domain to multiplication in the complex domain, transforming formula (6) into an easily solvable equation in the complex domain. After solving, an inverse Fourier transform yields the numerical solution of the Poisson equation [**?**]. The FFT method has computational time complexity of $O(N \log N)$ and, due to its speed, is the primary method for solving Poisson equations in recent gravitational N-body simulation applications (Gadget-3, Gadget-4, etc.) [**?**].

The three-dimensional Poisson equation' s second-order difference operator differs from the two-dimensional case, with larger computational requirements. Our goal is to train a deep neural network to achieve $O(N)$ computational time complexity for solving the deconvolution process in formula (6), thereby verifying the feasibility of replacing the Fast Fourier Transform method with deep neural networks for solving Poisson equations and accelerating collisionless gravitational N-body simulations. Since our focus is verifying whether deep neural networks can solve the deconvolution process in formula (6), we can validate this using the less computationally intensive two-dimensional Poisson equation.

Current N-body numerical simulations primarily handle particle numbers between 10^6 and 10^{13}. With increasing computational power, the simulatable particle number scale grows at approximately double the rate annually, following Moore' s Law. In collisionless gravitational N-body numerical simulations, the main factor affecting simulation speed is the Poisson equation solving speed. The commonly used FFT method has computational time complexity of $O(N \log N)$. For a particle number scale of 10^{13}, each Poisson equation solve could theoretically be about 40 times faster with our deep neural network model, which has computational time complexity of $O(N)$ [**?**]. In the PM-Tree method, the Poisson equation solving step is executed many times in loops, so the cumulative acceleration from an $O(N)$ method for collisionless gravitational N-body

numerical simulations will be substantial. Moreover, as simulated particle number scales continue to increase in the future, the time savings from an $O(N)$ method will become even more pronounced.

3.5 Deep Neural Network Model

Deep neural networks possess strong high-dimensional feature extraction capabilities. Internally, they operate in a very high-dimensional feature space, making complex deep neural network models robust to input data that doesn' t change too drastically. The transition from two-dimensional to three-dimensional data is not a huge change relative to the very high-dimensional internal feature space of deep neural networks. Even when data complexity changes significantly, as long as the feature principles remain unchanged, deep neural networks can correspondingly increase model complexity while maintaining effectiveness [**?**, **?**]. Therefore, our main goal is to verify that deep neural networks can implement the deconvolution process in formula (6).

We experimented with dozens of network structures for fast Poisson equation solving and ultimately identified the best-performing architecture. Since end-to-end neural network structures for semantic translation commonly use Encoder-Decoder architectures [**?**], which share similarities with the $\phi \to \rho \to \phi$ transformation in formula (4), we adopted this overall architecture. Network building blocks borrow from ResNet [**?**] structures: one type adds a convolutional layer' s input to its output (residual convolution), while another adds a transposed convolutional layer' s input to its output (residual deconvolution). To ensure the input simulated distribution map ρ and output predicted potential map ϕ have the same size, we used a fully symmetric network structure: the first half applies one convolution and one residual convolution three times consecutively, then performs channel-preserving convolution and deconvolution to extract high-dimensional features. The latter half applies one residual deconvolution and one convolution three times consecutively to express high-dimensional features as the required potential distribution, as shown in [Figure 1: see original paper]. All convolutional or transposed convolutional layers have kernel size 3, stride 1, and padding 1. Activation functions use ReLU functions wrapped in a linear function.

Our training objective is to minimize the mean squared error between predicted values and true potential values:

$$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\phi_{pred,i} - \phi_{true,i})^2$$

We use maximum absolute difference to evaluate error:

$$L_{max} = \max|\phi_{pred} - \phi_{true}|$$

The computational time complexity of convolutional neural network models is [**?**]:

$$O(\sum_{l=1}^{d} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2)$$

where l is the convolutional layer index, d is network depth, s_l is kernel size, m_l is output feature map size, and n_l is the number of kernels in layer l. Since m_l equals the output feature map size of the last layer m_d, and the neural network model' s intermediate parameters are fixed, the computational time complexity of this paper' s neural network model is $O(N)$. Our experiments further demonstrate that the neural network model is a linear complexity function.

4 Training the Model

Our objective is to use deep neural networks to replace the Fast Fourier Transform method for solving potential energy (solving the Poisson equation) in the PM-Tree method, verifying the feasibility of deep learning for accelerating collisionless gravitational N-body numerical simulations. The key task is verifying whether deep neural networks can implement the deconvolution process in formula (6), requiring training a deep neural network model to serve as a Poisson equation solver. We use randomly generated matrices to simulate the potential energy ϕ that needs to be predicted after gridding, which is the numerical solution of the Poisson equation. We obtain the simulated input distribution ρ by applying second-order differencing to the generated images. We build a deep neural network model that uses multiple convolutional layers to extract features and multiple transposed convolutional layers to express features and output predicted potential energy. We generate millions of data images and train the deep neural network model using mean squared error. Finally, we test the trained model for accuracy, evaluate it on simple isochrone potential problems, compare its accuracy with the Fast Fourier Transform method, compare its computation time with both Fast Fourier Transform and finite difference methods, and test it with various gridding methods.

4.1 Dataset

The true values consist of randomly generated 64×64 grids. We assume the solution region is sampled as a 64×64 matrix, with the surrounding 一圈值设置为 0 as Dirichlet boundary conditions. Dirichlet boundary conditions are used because in gravitational N-body numerical simulations, the large region around the matter clump of interest is typically vacuum and can be approximated as zero. To prevent network training from being overly influenced by the random function itself (which produces almost all irregular jump values with few smooth or identical cases), we also artificially added special patterns for data augmentation: Gaussian smoothing of central regions, setting central region values to

0, and setting corner region values to 0.99. These images simulate solutions to the Poisson equation—the potential energy distribution ϕ—and serve as the prediction target for the deep neural network model, as shown in [Figure 2: see original paper]-[Figure 5: see original paper].

The distribution ρ to be solved can be obtained from the simulated potential ϕ through formula (6), as shown in [Figure 6: see original paper]-[Figure 9: see original paper]. These images serve as input to the deep neural network model. Since the data is randomly generated, there is an unlimited supply of training data. We generate one million pairs of data for each training session.

4.2 Training Parameters

We implement the deep neural network model in PyTorch. Each training session uses one million pairs of data with batch size set to 64. The optimizer is Adam [**?**], with learning rate adjusted from 10^{-3} to 10^{-5} as needed. Each dataset is trained for 10 epochs. The GPU used is Tesla P100.

5 Results

5.1 Training Results

Training results are shown in [Figure 10: see original paper]-[Figure 13: see original paper]. Inputting the simulated distribution function ρ into the neural network model yields predicted images that are compared with simulated potential ϕ true-value images, and their differences are calculated. The relative error between prediction and simulation ϕ can be seen to be at most 10^{-4}, with accuracy meeting Poisson equation solving objectives.

5.2 Test Results

Using 10,000 newly generated data pairs for testing, the distribution of maximum errors is shown in [Figure 14: see original paper], and the accuracy rates at different maximum error precisions are statistically presented in .

The test results for 10,000 sets of data show that the deep neural network model's predicted images are visually almost indistinguishable from simulated potential ϕ true-value images, with basically matching image features.

5.3 Testing with Different Gridding Data

To verify model universality, we test the neural network model trained on 64×64 data on images with progressively increasing gridding sizes (128×128, 256×256, 512×512), as shown in [Figure 15: see original paper]-[Figure 16: see original paper].

The maximum error trend flattens and does not increase linearly with gridding size. The mean squared error plot more clearly shows that as gridding size increases, the averaged squared error actually decreases, stabilizing when the

gridding size is sufficiently large. Two examples at 128×128 and 512×512 are illustrated in [Figure 17: see original paper]-[Figure 24: see original paper].

The results show that a network trained on 64×64 data remains universal at larger scales. Comparative analysis across various scales reveals that the increase in maximum error is due to larger-sized data containing more local special patterns—these are the points with larger errors observed during test set evaluation. Overall, however, models trained on small-size data can be applied to large-size data, meaning model training time costs can be very low.

5.4 Isochrone Potential Problem Test

We apply the trained model to a real physical problem—the isochrone potential problem. We directly generate isochrone potential problem inputs ρ according to formulas (11-12) and feed them into the trained deep neural network model for testing, obtaining predictions that are compared with simulated ϕ values, as shown in [Figure 25: see original paper]-[Figure 28: see original paper].

The deep neural network model performs well even on actual physical problems. Using the same data for Fast Fourier Transform method solving for accuracy comparison, as shown in [Figure 29: see original paper]-[Figure 30: see original paper], we observe that at the same sampling rate, the deep neural network model achieves higher accuracy than the Fast Fourier Transform method, meeting accuracy requirements.

5.5 Computation Time Comparison with Fast Fourier Transform and Finite Difference Methods

We generate data with progressively increasing gridding sizes and compare computation times among the trained deep neural network model, Fast Fourier Transform method, and finite difference method. Test results are shown in [Figure 31: see original paper]-[Figure 32: see original paper].

The experimental results demonstrate that the deep neural network method for solving Poisson equations becomes faster than both Fast Fourier Transform and finite difference methods as data gridding size increases. This is due to two factors: first, deep neural networks can utilize GPUs for fast computation; more importantly, the experimental results show that deep neural network computation time grows linearly, while Fast Fourier Transform grows at $O(N \log N)$ and finite difference methods grow even faster at $O(N^2)$. This conclusion corroborates the theoretical analysis of convolutional neural network computational time complexity presented earlier.

5.6 Overview of Failed Neural Network Model Structures

We describe some network structures that produced poor experimental results, using the same data generation method as above. The first is a pure convolutional network structure with channel numbers: 64-128-256-512-256-128-64.

The mean squared error could only converge to around 0.001. Inserting normalization layers after each convolutional layer in this structure worsened performance, with mean squared error converging only to around 0.01. Adding a building block that convolves input once, adds the output to the input as the next layer' s input, then convolves back to the original dimension performed worse than pure convolution, with mean squared error converging only to around 0.01. Adding two fully connected layers at the end of the pure convolutional network not only dramatically increased parameters but only achieved mean squared error around 0.001. Adding fully connected layers to both front and back achieved mean squared error of 0.0005, but with large parameter counts and slow training, still underperforming simple pure convolution. Changing stride to 2 in two middle layers of the pure convolutional network to extract more concise features, then using transposed convolution for upsampling in subsequent layers, reduced mean squared error to 0.0005—better than pure convolution but still not meeting accuracy requirements. A network structure with alternating single convolution and single transposed convolution iterations only achieved mean squared error of 0.001. Based on these experiments, we designed our final deep neural network model structure by synthesizing the advantages and disadvantages of various architectures.

5.7 Interesting Experimental Phenomenon

All our data uses Dirichlet boundary conditions. We experimented with what happens without boundary conditions. Training the neural network model on 64×64 data without set boundaries produced training processes and accuracy similar to the bounded case, with relative error precision reaching 10^{-4} at the same gridding size. However, when applying this trained model to larger gridding size data, an interesting phenomenon occurred: it could only predict within a 64×64 range from the edges, with the central portion nearly zero, as shown in [Figure 33: see original paper]-[Figure 40: see original paper].

Analyzing this phenomenon in conjunction with the model' s universality across different gridding data suggests it may occur because the trained neural network model performs local iterative operations while solving the Poisson equation from boundaries inward.

5.8 Experimental Summary

Comprehensive experimental results demonstrate that: deep neural network models can rapidly solve potential energy (Poisson equation) in the PM-Tree method; their accuracy is no lower than the Fast Fourier Transform method; their speed surpasses both Fast Fourier Transform and finite difference methods, with the advantage becoming more pronounced as particle number scale increases in collisionless gravitational N-body numerical simulations; they are applicable to larger gridding size data without retraining, showing scalability. Therefore, deep learning methods are feasible for accelerating collisionless gravitational N-body numerical simulations.

6 Conclusion

Building on research of neural networks solving various equations and several collisionless gravitational N-body numerical simulation methods, we propose an Encoder-Decoder deep neural network model supplemented with residual local structures to solve potential energy (Poisson equation) in the PM-Tree method, replacing the commonly used FFT solver. We verify that deep neural network models can quickly solve Poisson equations; theoretically, deep neural networks have $O(N)$ computational time complexity; we verify that deep neural network computation time is lower than Fast Fourier Transform and finite difference methods; we verify that at the same discretization sampling rate, deep neural network accuracy is no lower than Fast Fourier Transform methods; we verify that deep neural networks have scalability, with models trained on small gridding datasets applicable to large gridding datasets. Moreover, deep neural network models can better utilize GPU computational resources for parallel processing, with speed advantages becoming more evident at larger particle number scales. Based on these verification results, we propose that using deep neural networks to replace FFT methods for solving potential energy (Poisson equation) in the PM-Tree method can accelerate the primary time-consuming step in PM-Tree, thereby accelerating collisionless gravitational N-body numerical simulations and verifying the feasibility of deep learning in this domain. This lays the foundation for future practical applications in large-scale collisionless gravitational N-body numerical simulations.

7 Future Work and Outlook

Future work focuses on three aspects. First, extending Poisson equation solving from two-dimensional to three-dimensional (simply by changing 2D convolutions to 3D convolutions in the current neural network model). Second, adding iterative solving processes to enable time-dependent simulations. Third, embedding the model into actual PM-Tree methods for collisionless gravitational N-body numerical simulations.

We thank Liu Dongming and Zhai Shuncheng for their suggestions during this work, and Gu Jianyu for corrections to portions of the manuscript.

References

[1] W. Dehnen and J. I. Read. N-body simulations of gravitational dynamics. *The European Physical Journal Plus*, 126(5), May 2011.

[2] Springel, Volker; Pakmor, Rüdiger; Zier, Oliver; Reinecke, Martin. Simulating cosmic structure formation with the GADGET-4 code. *arXiv preprint arXiv:2010.03567*, 2020.

[3] Sverre J Aarseth. *Gravitational N-body simulations: tools and algorithms*. Cambridge University Press, 2003.

[4] Josh Barnes and Piet Hut. A hierarchical o (n log n) force-calculation algorithm. *nature*, 324(6096):446–449, 1986.

[5] Andrew W Appel. An efficient program for many-body simulation. *SIAM Journal on Scientific and Statistical Computing*, 6(1):85–103, 1985.

[6] James W Wadsley, Joachim Stadel, and Thomas Quinn. Gasoline: a flexible, parallel implementation of treesph. *New astronomy*, 9(2):137–158, 2004.

[7] M Gritschneder, T Naab, A Burkert, S Walch, F Heitsch, and M Wetzstein. ivine–ionization in the parallel tree/sph code vine: first results on the observed age-spread around o-stars. *Monthly Notices of the Royal Astronomical Society*, 393(1):21–31, 2009.

[8] JW Eastwood and RW Hockney. *Computer simulation using particles*. New York: Mc GrawHill, 1981.

[9] Volker Springel, Simon DM White, Adrian Jenkins, Carlos S Frenk, Naoki Yoshida, Liang Gao, Julio Navarro, Robert Thacker, Darren Croton, John Helly, et al. Simulations of the formation, evolution and clustering of galaxies and quasars. *nature*, 435(7042):629–636, 2005.

[10] Paul Bode and Jeremiah P Ostriker. Tree particle-mesh: An adaptive, efficient, and parallel code for collisionless cosmological simulation. *The Astrophysical Journal Supplement Series*, 145(1):1, 2003.

[11] Volker Springel. The cosmological simulation code gadget-2. *Monthly notices of the royal astronomical society*, 364(4):1105–1134, 2005.

[12] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of computational physics*, 73(2):325–348, 1987.

[13] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[14] Yoshua Bengio. Learning deep architectures for AI. Now Publishers Inc, 2009.

[15] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[16] Donald F Specht et al. A general regression neural network. *IEEE transactions on neural networks*, 2(6):568–576, 1991.

[17] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[19] Zhen Zuo, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, Bing Wang, and Yushi Chen. Convolutional recurrent neural networks: Learning spatial dependencies for image representation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 18–26, 2015.

[20] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[21] Jiequn Han, Arnulf Jentzen, and E Weinan. Overcoming the curse of dimensionality: Solving high-dimensional partial differential equations using deep learning. *arXiv preprint arXiv:1707.02568*, pages 1–13, 2017.

[22] J Binney and S Tremaine. *Galactic dynamics*, ed. Binney, J. & Tremaine, S, 1987.

[23] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 481–490, 2016.

[24] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

[25] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019.

[26] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artificial neural networks. *arXiv preprint arXiv:1707.03351*, 2017.

[27] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017.

[28] Wei Tang, Tao Shan, Xunwang Dang, Maokun Li, Fan Yang, Shenheng Xu, and Ji Wu. Study on a poisson' s equation solver based on deep learning technique. In *2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, pages 1–3. IEEE, 2017.

[29] E Weinan and Bing Yu. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[30] Kaushik Bhattacharya, Bamdad Hosseini, Nikola B Kovachki, and Andrew M Stuart. Model reduction and neural networks for parametric pdes. *arXiv preprint arXiv:2005.03180*, 2020.

[31] Jonathan D Smith, Kamyar Azizzadenesheli, and Zachary E Ross. Eikonet: Solving the eikonal equation with deep neural networks. *arXiv preprint arXiv:2004.00361*, 2020.

[32] Michele Trenti and Piet Hut. N-body simulations (gravitational). *Scholarpedia*, 3(5):3930, 2008.

[33] Bagla and Jasjeet S. TreePM: A code for cosmological N-body simulations. *Journal of Astrophysics and Astronomy*, Volume 23, Issue 3:185-196, 2002.

[34] Zhang Wensheng. *Finite Difference Methods for Partial Differential Equations in Science Computation*[M]. Higher Education Press, 2006:115-137,172-185.

[35] Roland C Le Bail. Use of fast fourier transforms for solving partial differential equations in physics. *Journal of Computational Physics*, Volume 9, Issue 3:440-465, 1972.

[36] Shamil Chollampatt and Hwee Tou Ng. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *arXiv preprint arXiv:1801.08831*, 2018.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[38] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5353–5360, 2015.

[39] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Note: Figure translations are in progress. See original paper for figures.