Deep Learning Modeling of Subgrid Physics in Cosmological N-body Simulations

Georgios Markos Chatziloizos
LAMSADE, University Paris Dauphine - PSL
g.chatziloizos@gmail.com

Tristan Cazenave
LAMSADE, University Paris Dauphine - PSL
Tristan.Cazenave@dauphine.psl.eu

François Lanusse
AIM, CEA, CNRS, Université Paris-Saclay, Université Paris Diderot,
Sorbonne Paris Cité, F-91191 Gif-sur-Yvette, France
francois.lanusse@cea.fr

Abstract

Calculating N-body simulations has been an extremely time and resource consuming process for researchers. There have been many schemes trying to approximate the correct positions of celestial objects of simulations. In this paper, we propose using Neural Networks in the physical and in the Fourier domain in order to correct a Particle Mesh scheme, primarily in the smaller scales i.e., the smaller details of the N-body simulations. In addition, we used a recently proposed in the literature technique to train our models i.e. through an Ordinary Differential Equations (ODE) solver. We present our promising results of the different types of Neural Networks that we experimented with.

1 Introduction

An N-body simulation simulates the motion of celestial objects, that interact with each other via physical forces. Since the universe is mostly consisted of dark matter and dark energy, we focused primarily on dark matter N-body simulations from CAMELS [13]. For the simulations which are periodic, we used a Particle Mesh (PM) scheme, which provides fast but not that accurate results in the smaller scales. The N-body problem is one of the most famous ones in astrophysics. In this problem, N particles interact with each other with gravitational forces. When the number of particles is fairly small, the prediction of their positions even by a CPU is straightforward. Is the time also trivial when we need to calculate a system of hundred of thousand or even millions celestial objects? Obviously not, trying to brute force a system like that could even take up to $5 * 10^4$ hours in CPU time, which is about 6 years [4].

Therefore, researchers have turned to Deep and Machine Learning techniques in order to solve the N-body problem through predicting the positions of the particles, instead of calculating every position for every time step for every particle. There is a number of machine learning methods that can simulate computationally expensive N-body simulations. Specifically, He et al. in [5] developed a deep neural network to learn the nonlinear mapping from first order perturbation theory linear displacements to the displacement field of Fast Particle Mesh simulations. By training a generative model to super-resolve the particle displacement field, Li et al. in [10] showed how to improve the resolution of a low-cost approximate N-body. These methods depend on large Deep Convolutional Networks to learn an efficient mapping that produces the necessary outputs even if they use particle displacements as inputs and outputs of their modeling.

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

In order to shorten the computation time and produce low-cost realizations of the large-scale structure, quasi N-body numerical methods have been devised. These methods include the Fast Particle Mesh (FastPM) N-body solver proposed by Feng et al. in [14] and the Potential Gradient Descent (PGD) proposed by Dai et al. in [3]. The PGD method is a gradient based method to correct the PM approximation in FastPM and improve the modeling of the matter distribution within halos. Very recently, Lanzieri et al. in [9] presented an alternative to the PGD correction scheme for a quasi N-body PM solver, based on Neural Network implemented as a Fourier-space filter.

The goal of this work is to improve the precision of N-body simulations snapshots from a Particle Mesh scheme. To achieve this, we employed and compared various Deep Neural Networks to learn effectively corrections to the differential equations solved by the PM solver to recover the correct behaviour on small scales.

2 Process

This work was implemented in JAX [1]. The code[1] ran in an NVIDIA RTX A6000 GPU. We used 10 data points for training and one for testing. Each data point of our dataset needed about 15GB of storage. Having completed setting up our data and the prepossessing part, we used an ODE solver from JAX as a black box and calculated the gradients with the adjoint sensitivity method [12]. In this way we get an approximation of the positions and the velocities of the particles. The Particle Mesh scheme creates a density field in 3D and the ODE solver solves it in the Fourier Domain. The way the approximation is calculated is the following:

The Particle Mesh scheme gets the positions of the particles in the 3 dimensions ($\#ofParticles^3$,3) and produces the density field delta (N, N, N). Afterwards, with the help of Fast Fourier Transformation, it calculates the gravitational potential with dimensions $(N, N, \frac{N}{2} + 1)$ and finally with the inverse Fast Fourier Transformation the gravitational forces on the particles ($\#ofparticles^3$,3). From the forces and velocities, the new positions and velocities are computed. Unfortunately, this scheme is not capable of approximating the smaller scales (small details) accurately, but only the larger scales. Thus, the use of a Neural Network is considered necessary in order to model the smaller scales that are not correct. Our Neural Networks correct either on the Gravitational Potential or on the Gravitational Forces. To be more precise, the ODE for correcting the Gravitational Potential and the Gravitational Forces are:

$$\begin{cases} \frac{d\mathbf{x}}{da} & = \frac{1}{a^3 E(a)}\mathbf{v} \\ \frac{d\mathbf{v}}{da} & = \frac{1}{a^2 E(a)} F_\theta(input) \end{cases} \tag{1}$$

For the gravitational potential, we have implemented the parametric function F_θ, hybrid between a physical model and neural network:

$$F_\theta(input) = \frac{3\Omega_m}{2}\nabla\left[\phi_{PM}(\mathbf{x}) * \mathcal{F}^{-1}(1 + f_\theta(input))\right] \tag{2}$$

For the gravitational forces, we have implemented the parametric function F_θ, a hybrid between the physical model and a neural network:

$$F_\theta(input) = Forces + f_\theta(input) \tag{3}$$

where x are the positions and v the velocities of the particles, $\mathcal{F}^{-1}$ is the inverse Fourier Transform, ϕ_{PM} is the approximated gravitational potential (which we correct), $Forces$ are the approximated gravitational Forces (which we correct) and derive from ϕ_{PM} which is the approximated gravitational potential, α is the cosmological scale factor and f_θ is one of the networks that we present at section 3 with their $input$.

[1] `https://github.com/GeoMarX/PM_NNs`

For the loss function we are trying to fix the positions of the particles and use as a regularizer the power spectrum.

$$\mathcal{L} = \sum_i \|\mathbf{x}_i^{True} - \mathbf{x}_i\|_2^2 + \lambda \| \frac{\mathbf{P}_i}{\mathbf{P}_i^{True}} - 1 \|_2^2 \tag{4}$$

where i represents a snapshot, x is the positions of the particles, λ is the regularization rate (hyper parameter), while P represents the Power spectrum for each snapshot.

Finally, we chose the Adam optimizer from Optax[7] to train the following Neural Networks and use a learning rate of 0.005.

3 Neural Network Architectures and Results

In this work, we used several Neural Network Architectures, some simple ones and some Deeper and more complicated ones which either correct the Gravitational Potential or directly the Gravitational Forces.

3.1 Residual on the Gravitational Potential

3.1.1 SpectralConv3d / Simple Fourier

Our first attempt was a Fourier layer (SpectralConv3d) from [11]. This layer implements a convolution in Fourier space as a trainable weight map that gets multiplied with the Fourier Transform of the input signal. This procedure does not happen in the Euclidean space (N, N, N), but in the Fourier Space $(N, N, \frac{N}{2} + 1)$ which we transitioned to through the Fast Fourier Transformation. The output of the model is also an array of dimensions $(N, N, \frac{N}{2} + 1)$.

3.1.2 Fourier B-spline

This Neural Network was proposed by [9] and it is defined as the previous one in the Fourier Domain, but instead of parameterising the Fourier convolution by an $N \times N \times N/2$ weight map, it parameterises the filter in terms of an isotropic 1d transfer function.

3.2 Residual on the Gravitational Forces

This task is a bit more difficult than the previous one and thus we need more complex neural networks which need even 10 times more training time than the previous Neural Networks. The following networks are different from the previous ones as they calculate directly the correction on the gravitational forces of each particle in the 3D space. So, as an input we use the positions of the particles shapes (N, N, N) and also we use the time as a plane to the input. There was major improvement when the time dependence was added to the "equation". In both of the following networks, the wrap padding was used for Convolutional layers. This "tool" was absolutely necessary to ensure the periodicity of the simulation volume.

3.2.1 Fourier Space and CNNs with skip connections

This is the FN03d Neural Network proposed by [11] with some small improvements. It combines SpectralConv3d and Convolution layers. We added some skip connections and the CNNs have kernel 3 and not just 1, which is just a multiplication.

There are 4 layers of SpectralConv3d and 4 layers of CNNs also. Each SpectralConv3d is added to one CNN and the previous input (skip connection). The activation function GELU [6] was used for both Fourier and "Real" space. Crucial for the efficiency of the network was the 'wrap' padding that was added at the CNNs in the physical space. Finally, the number of output filters is 10 for the FN03d.

3.2.2 MobileNet

This is the only Neural Network that does not involve Fourier space and the correction happens only in physical space. For this particular Neural Network, we added as an input the velocities of the particles which improved the model's efficiency, in contrast to the previous one. This network was the slowest to compute in comparison to the previous ones.

Since their conception MobileNets have been extremely useful in computer vision[8], but also have other applications such as in the game of Go[2]. The network is consisted of 6 blocks that are similar to MobileNet Blocks. Each block is consisted of one Convolution3D layer, followed by a DepthwiseConvolution3D layer and finally a Convolution3D layer. In every case we used a $3 \times 3 \times 3$ kernel and GELU[6] activation function. Also, padding was 6 at each side of the cube for each Convolution layer that we had with mode 'wrap'. The number of channels used in the Convolution layers is 20 for the MobileNet. At the end, there is one Convolution3D layer $1 \times 1 \times 1$ just to bring our data in the dimension $(N, N, N, 3)$, so the neural ODE function can use them to create the forces for all particles.

3.2.3 All-in-one Power Spectra

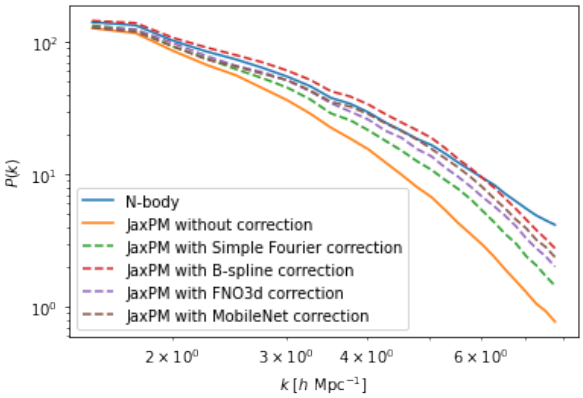

Figure 1: Power Spectrum Comparison

The power spectrum was used as an evaluation method for the Neural Networks. In Figure 1, the orange line is what the Particle Mesh scheme provides us with and the blue line is the true power spectrum. We use the Neural Networks in order to diminish as much as possible this gap. We can conclude that the models that have the best Power Spectrum, are the B-spline and the MobileNet. Even though the B-spline has better smaller scales, which means it is more accurate in the small details than the other models, it "over corrects" (it is above the blue line) in the larger scales. The FNO3d follows closely the latter two Networks and has similar results.

4 Conclusions

In this work, we tested many different and unique Neural Networks in order to correct a PM scheme through an ODE solver and get as close as possible to the N-body simulations. Most of the models involved the symmetrical Fourier space, while the MobileNet was exclusively in the physical space. Also, we experimented with correcting the gravitational potential and the gravitational forces. For the power spectrum correction, the B-spline and the MobileNet seemed to have had the best results.

For future work, it would be really interesting to experiment with a combination of the B-spline and a MobileNet correcting at the same time the gravitational potential and the gravitational forces. In this way, we will try to achieve to fix the over correction of the larger scales of the B-spline model, but also improve the smaller scales in which the MobileNet is under performing in comparison to the B-spline.

Impact statement

These models allow us to solve the N-body problem better than just the Particle Mesh scheme. The more complex models that we improved and implemented, are capable of taking as an input the density field and the velocities of the particles. Also, in future work, they will allow us with the use of a latent variable to map other types of particles such as gas particles.

References

[1] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[2] Tristan Cazenave. Mobile networks for computer go. *IEEE Transactions on Games*, 14(1):76–84, 2022.

[3] Biwei Dai, Yu Feng, and Uroš Seljak. A gradient based method for modeling baryons and matter in halos of fast simulations. *Journal of Cosmology and Astroparticle Physics*, 2018(11):009–009, nov 2018.

[4] Biwei Dai and Uroš Seljak. Learning effective physical laws for generating cosmological hydrodynamics with lagrangian deep learning. *Proceedings of the National Academy of Sciences*, 118(16), apr 2021.

[5] Siyu He, Yin Li, Yu Feng, Shirley Ho, Siamak Ravanbakhsh, Wei Chen, and Barnabás Póczos. Learning to predict the cosmological structure formation. *Proceedings of the National Academy of Sciences*, 116(28):13825–13832, 2019.

[6] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2016.

[7] Matteo Hessel, David Budden, Fabio Viola, Mihaela Rosca, Eren Sezener, and Tom Hennigan. Optax: composable gradient transformation and optimisation, in jax!, 2020.

[8] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[9] Denise Lanzieri, François Lanusse, and Jean-Luc Starck. Hybrid physical-neural odes for fast n-body simulations, 2022.

[10] Yin Li, Yueying Ni, Rupert A. C. Croft, Tiziana Di Matteo, Simeon Bird, and Yu Feng. Ai-assisted superresolution cosmological simulations. *Proceedings of the National Academy of Sciences*, 118(19):e2022038118, 2021.

[11] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

[12] Lev Semonovic Pontrjagin. *The mathematical theory of Optimal Processes*. MacMillan, 1964.

[13] Francisco Villaescusa-Navarro, Daniel Anglé s-Alcázar, Shy Genel, David N. Spergel, Rachel S. Somerville, Romeel Dave, Annalisa Pillepich, Lars Hernquist, Dylan Nelson, Paul Torrey, Desika Narayanan, Yin Li, Oliver Philcox, Valentina La Torre, Ana Maria Delgado, Shirley Ho, Sultan Hassan, Blakesley Burkhart, Digvijay Wadekar, Nicholas Battaglia, Gabriella Contardo, and Greg L. Bryan. The CAMELS project: Cosmology and astrophysics with machine-learning simulations. *The Astrophysical Journal*, 915(1):71, jul 2021.

[14] Uroš Seljak Patrick McDonald Yu Feng, Man-Yat Chu. Fastpm: a new scheme for fast simulations of dark matter and haloes. *Monthly Notices of the Royal Astronomical Society*, 463(3):2273–2286, Dec 2016.