Fast-forward *FARGO*: Accelerating Protoplanetary Disk Simulations with Limited Data

Anonymous Authors[1]

Abstract

Hydrodynamic simulations of planets embedded in accretion disks around young stars are an essential tool in the study of planet formation and evolution. However, these simulations are expensive, and any quantitative comparisons between planet-formation theory and the observations of planets will require the execution of at least thousands of simulations. We present a U-net emulator built to accelerate the simulation package *FARGO*. Our emulator, `Freesbee`, was pre-trained with disk surface densities from 940 short, low-resolution *FARGO* simulations and fine-tuned using only 10 to 30 longer and higher-resolution simulations. The emulator takes as input the disk density and embedded planet position from the first few timesteps of a brief simulation and outputs the final state 5,000 dynamical times (years at 1 a.u.) later. The emulated disk densities have median fractional errors (relative to high-resolution *FARGO* runs) ranging from 0.9 to 6 percent with a median value of 3 percent; they are over 10^6 times faster to compute.

1. Introduction

The field of planet formation theory relies heavily on simulations that model protoplanetary disks and the evolution of planets within them. Scientists have developed various tools to study the time evolution of the differential equations that describe protoplanetary systems. One widely used tool is the Fast Advection in Rotating Gaseous Objects (*FARGO*) code (Benítez-Llambay & Masset 2016, Masset 2000), a parallel hydrodynamics and magnetohydrodynamics code that simulates planet-disk interactions. Despite the utility of *FARGO*, the computational expenses associated with running such simulations remain a barrier to addressing some of the most complex questions in planetary system evolution. For example, planet migration, which refers to the change of a planet's semi-major axis over time, is a process that can take place over hundreds of thousands of years. Problems requiring such long-term simulations can often rapidly increase computational costs.

1.1. Related Work

Machine learning techniques are increasingly being used to mitigate some of the computational costs associated with running expensive simulations. For example, Timpe et al. (2020) describes a data-driven emulator that predicts the outcome of pairwise collisions between planetary-sized bodies in the late stages of planet formation. Jamieson et al. (2023) and He et al. (2019) present emulators to accelerate cosmological dark-matter simulations to predict the large-scale structure of the Universe. Emulators are being used in the field of Earth sciences to learn derivatives governing the chaotic evolution of oceans and atmospheres (Nonnenmacher & Greenberg, 2021). Across many physical science domains, investigators are turning to deep learning to build fast emulators for computer simulations and digital twins, aiming to enable parameter exploration and uncertainty quantification (Kasim et al., 2021).

However, the datasets necessary to effectively train and validate a neural network are generally large. Running sufficient protoplanetary disk evolution simulations to train a traditional neural network is extremely expensive. U-nets, a type of Convolutional Neural Network (CNN), may offer a solution. This architecture, first developed for Biomedical Image Segmentation, has been shown to outperform standard CNNs when trained with smaller data sets (Ronneberger et al., 2015). This advantage is likely due to its numerous shared weights and its inherent equivariance to translation symmetries. The typical network consists of an encoder, which contracts the information to a smaller-dimensional space, and a decoder, which symmetrically reconstructs the encoded data.

U-net models are already being employed as emulators for a variety of tasks. For example, in geology, Jiang et al. (2021) developed an autoregressive residual U-net to predict the time-dependent subsurface flow for geological systems.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Bertrand et al. (2023) trained a model that emulates cardiac electrophysiology simulations and accurately predicts cardiac activation time maps. In astrophysics, Rouhiainen et al. (2024) learned a denoising function for cosmological simulations with a similar architecture.

1.2. Our contribution

In this paper, we introduce `Freesbee`, a U-net built and trained in PyTorch (Paszke et al., 2019) to time-evolve the surface density of protoplanetary disks and accelerate the *FARGO* simulations. We pre-trained the network with 1D surface densities of a wide array of short-duration, low-resolution *FARGO* simulations. We then fine-tuned the network on higher-resolution simulations of longer duration.

Section 2 contains a brief background on protoplanetary disk evolution. Section 3 describes the *FARGO* simulations used for pre-training and fine-tuning the network. The network architecture, training, and validation are outlined in Section 4. The resulting emulations are shown in Section 5. Lastly, Section 6 concludes with a discussion and summary of our work and suggests directions for future research.

2. Background

Understanding the evolution of protoplanetary disks and the formation of planets embedded in them is one of the most active areas of research in planetary science. Protoplanetary disks are disks of gas and dust surrounding young stars. They are the birthplaces of planets, which form from collisions and accretion of disk material.

The evolution of protoplanetary disks and the planets within them is a complex process that occurs over millions of years. Planets and their host disk interact gravitationally, producing torques that can result in planetary migration and a reshaping of the material within the disk. Planet-disk interactions are non-trivial and can give rise to non-axisymmetric features in the distribution of material in the disk. Studying the coupled evolution of planetary systems is an ongoing area of research.

For a more extensive review of planet formation and disk evolution see Armitage (2020).

3. *FARGO* Simulations

3.1. Overview

The *FARGO* code simulates the evolution of protoplanetary disks by solving hydrodynamics and magnetohydrodynamics equations (i.e. continuity, energy conservation, Navier-Stokes). A simple N-body simulator is employed to study the dynamics of the embedded planets.

The disk mass is described by a radial surface density profile:

$$\Sigma(r) = \Sigma_0 \left(\frac{r}{1\,\text{a.u.}}\right)^{-n} \tag{1}$$

where Σ_0 is the initial surface density at $r = 1\,\text{a.u.}$ and n the slope of the surface density profile

The physical geometry of the disk is determined by the aspect ratio:

$$h(r) = \frac{H}{r} = h_0 \left(\frac{r}{1\,\text{a.u.}}\right)^{f} \tag{2}$$

where H is the vertical height, h_0 is the aspect ratio at $r = 1\,\text{a.u.}$, and f is the flaring index.

The kinematic viscosity ν, which relates to the rate of gas diffusion in the disk, depends on the parameter α, the speed of sound c_s, and the vertical height H of the disk as follows:

$$\nu = \alpha\, c_s\, H \tag{3}$$

The planets are initialized by defining their initial radial position and mass, and whether they are allowed to interact with the disk. Other initial parameters include the duration of the simulations, angular and radial resolutions, and radial limits of the system.

3.2. Pre-training Simulations

To pre-train our network, we ran quick, low-resolution *FARGO* simulations in NYU's Greene Supercomputer with one Jupiter-mass planet and a variety of disk initial conditions to cover part of the parameter space studied in the literature (e.g. Kanagawa et al. 2015, Zormpas, Apostolos et al. 2022). The chosen values used for all the variable parameters are listed in Table 1.

α	Σ_0 (g/cm)	n	h_0 (H/R_0)
$\{3.0, 1.0\} \times 10^{-2}$	5,658	2	0.1
$\{3.0, 1.0\} \times 10^{-3}$	1,886	1.5	0.07
$\{3.0, 1.0\} \times 10^{-4}$	1,698	1.2	0.04
$\{3.0, 1.0\} \times 10^{-5}$	1,257	0.8	-
-	943	0.5	-
-	629	0.2	-
-	-	0.0	-

Table 1. Table of *FARGO* variable parameters used in our pre-training dataset. We created simulations with combinations of the listed parameters.

Our pre-training simulations have a grid resolution of $384(\phi) \times 128(r)$ and span from $0.4\,\text{a.u.}$ to $2.5\,\text{a.u.}$. We ran the simulations for 25 orbits at $1\,\text{a.u.}$ (25 years), with a time-step of 1/2 a year.

Lastly, we averaged the 2D surface density azimuthally for each time-step in the pre-training simulations, resulting

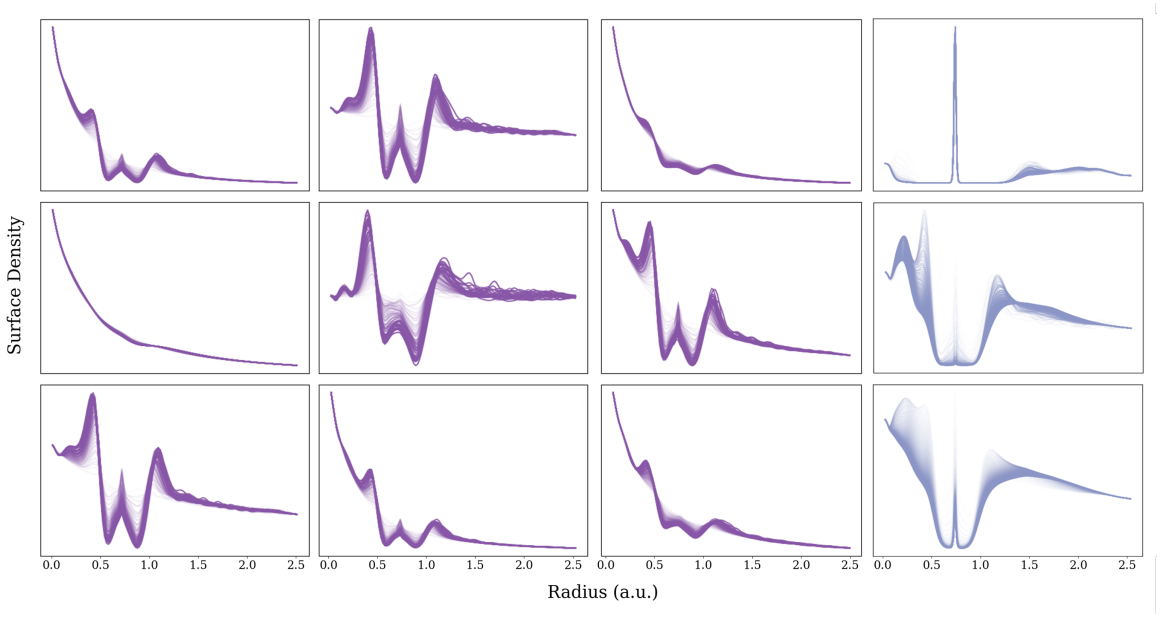

Figure 1. Disk surface density as a function of radius in a.u. from randomly selected *FARGO* runs. The time evolution is depicted by the darkening of the lines, where the lightest color represents the initial condition. The pre-training simulations (first three columns, left to right) span 25 years, while the fine-tuning systems (last column) span 6,000 years.

in 1D surface densities as a function of radius. Figure 1 contains examples of the surface density evolution simulated with *FARGO* that were used to pre-train our network.

3.3. Fine-tuning Simulations

For fine-tuning, we used *FARGO* simulations with higher resolution of longer duration. We recreated the simulations from Fung et al. (2014). The grid domain was the same as for our pre-training, but the numerical resolution was increased to 864×256. The simulations were run for 6,000 orbits at $1\,\mathrm{au}$, with a time-step of 100 years.

The values for α, h_0, and q (denoting planet mass in solar masses) are listed in Table 2, and n was fixed to be $1/2$. We prevented the planet from feeling the disk's influence, leading to a fixed orbit at $1\,\mathrm{a.u.}$ and removing any Σ_0 dependence.

As with the pre-training simulations, we calculated the 1D surface densities as a function of radius for each fine-tuning simulation time-step by averaging the 2D surface density over azimuth. The last column of Figure 1 shows examples of the surface density evolution simulated with *FARGO* that were used to fine-tune our network.

α	h_0	q
0.001	0.03	0.0001
0.01	0.04	0.001
0.1	0.06	0.01
-	0.07	-
-	0.08	-
-	0.1	-

Table 2. Table of *FARGO* variable parameters used in our fine-tuning simulations. We created simulations with all possible combinations of the listed parameters. We chose Σ_0 to be 0.005, but due to the planet's orbit being fixed, the evolution does not depend on the value of Σ_0.

4. Neural Network

4.1. Architecture

`Freesbee` is a U-net built to take as inputs the 1D surface density at three different time-steps along with the planet position and to output the 1D surface density and planet position at a later time-step. The planet position was appended as a float at the end of the surface density. For the pre-training we chose $t_{in} = \{1, 5, 10\}$ as the input time-steps and $t_{out} = 35$ as the output time-step. For the fine-tuning, $t_{in} = \{1, 5, 10\}$ and $t_{out} = 60$.

The U-net architecture consists of an encoder or contracting path followed by a decoder or expanding path (see Fig. 2). The encoder captures context by gradually reducing the spatial dimensions of the inputs through convolutional and pooling layers, extracting hierarchical features from the input data. The decoder employs upsampling and convolutional layers to increase the resolution of the features extracted by the encoder and constructs an output of the same size

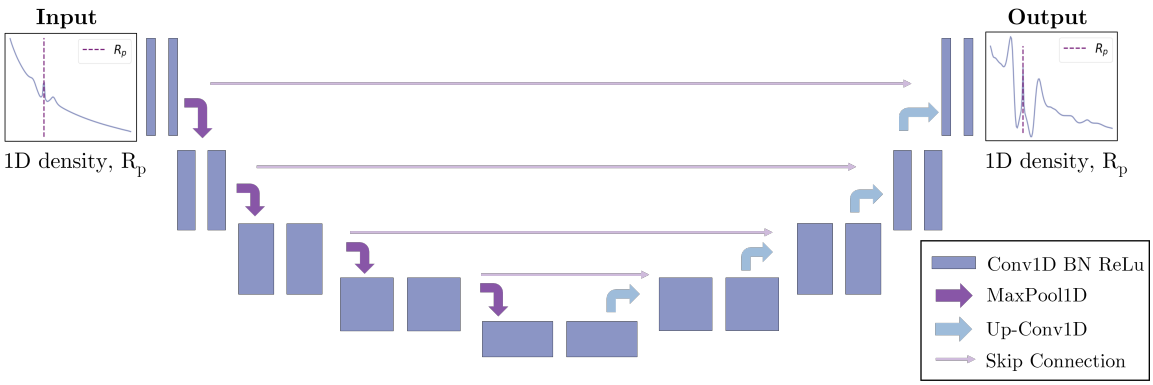

Figure 2. U-net architecture diagram. Each convolutional block contains two 1D convolutional layers with batch normalization and the ReLu activation function. The contracting path with 1D MaxPool layers between convolutional blocks is the encoder. The decoder is the expanding path, with 1D transpose convolution operations applied between convolutional blocks. Features from the encoder are concatenated to the corresponding decoder block via skip connections.

as the input, preserving the contextual information learned in the contracting stage. Features from the encoder are concatenated to the corresponding decoder blocks through skip connections.

We employed the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 0.001 to minimize the L1 loss. Additionally, a learning rate scheduler was implemented, reducing the learning rate by a factor of 0.9 every 30 epochs. The architecture's hyperparameters were optimized using the Weights and Biases development platform (Biewald, 2020), and are listed in Appendix A.

4.2. Training and Validation

The network underwent pre-training for 100 epochs and the model parameters corresponding to the lowest validation loss were used to initialize the fine-tuning training. We then trained the network for an additional 100 epochs with a range of $N = \{10, 15, 20, 25, 30\}$ of the high-resolution simulations.

The fine-tuning training was carried out four separate times for each N, cross-validating with 10 different validation simulations. For every N, the lowest validation loss in each of the four runs was recorded. Figure 3 illustrates the mean validation loss for each N, and the standard deviation from each of the four cross-validation runs.

Finally, to assess the pre-training utility, we trained the model using just the fine-tuning data for each N. Similarly to the fine-tuning, we trained the model four times for each N, cross-validating with 10 different simulations each time. The best validation losses were recorded, and their mean and standard deviation were plotted on Figure 3 alongside the pre-trained and fine-tuned model for comparison.

5. Results

We compared the network's performance with and without pre-training and for $N = \{10, 15, 20, 25, 30\}$. Figure 3 illustrates the dependence of the minimum validation L1 loss on the number of simulations used for training for both the pre-trained and non-pre-trained models. We found that the pre-trained model yields lower validation losses, with the lowest corresponding to $N = 30$.

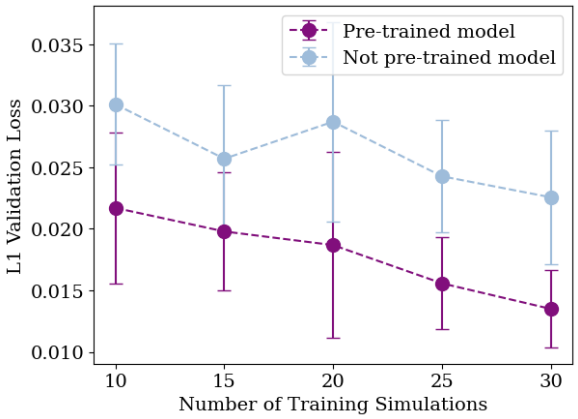

Figure 3. Minimum validation L1 loss as a function of the number of fine-tuning training *FARGO* simulations N. The error bars represent the standard deviation over each of the four cross-validation runs for every N.

We visualized the 60th time-step surface densities for validation systems as predicted by the pre-trained and fine-tuned model with $N = 30$. Figure 4 compares these simulated surface densities and their emulated counterparts for a few validation examples. The median fractional errors of the validation emulated disk densities compared to the high-resolution *FARGO* runs ranged from 0.9 to 6 percent

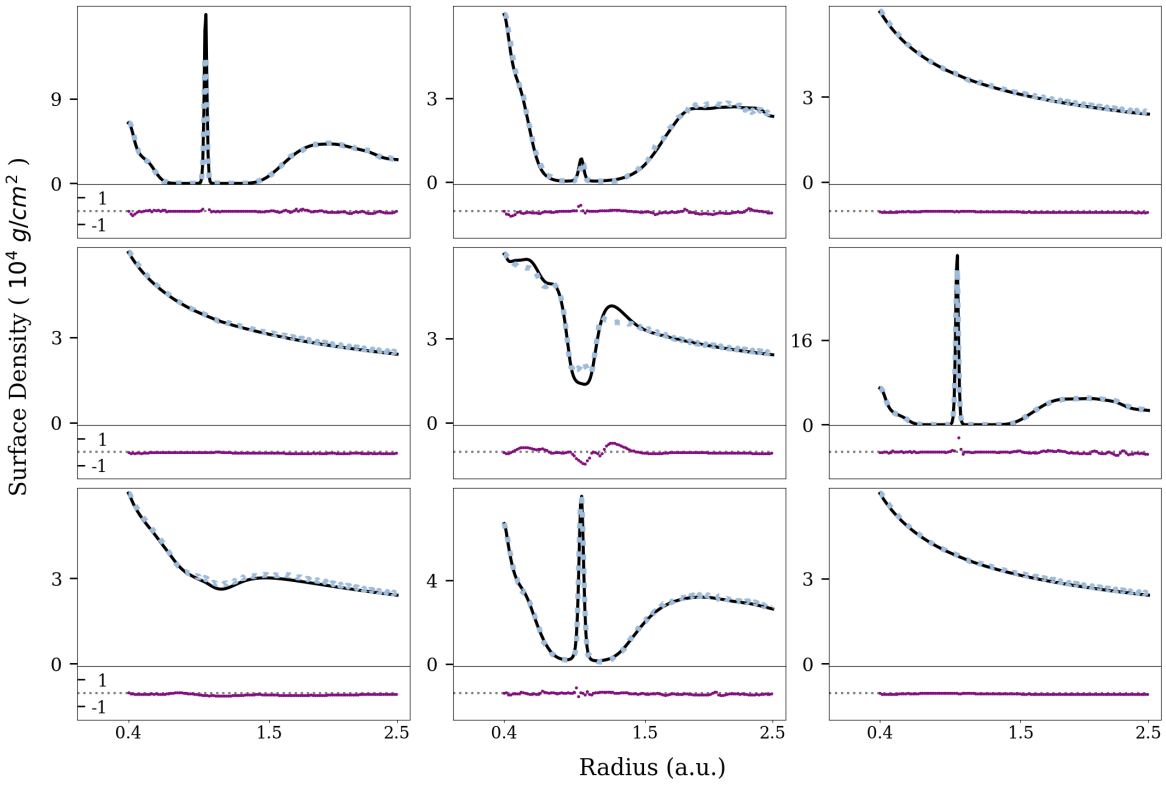

Figure 4. Surface density and residuals of the 60th time-step for fine-tuning validation systems. The top panel in each row contains surface density plots as a function of radius, where the blue dotted lines depict the emulations and the black solid lines depict the simulations. The midpoint of the y-axis is labeled on each top panel. The bottom panels in each row show the residuals as a solid purple line, with a y-axis range of $\pm 1 \times 10^4$ g/cm^{2}and a dotted gray line across 0 g/cm^2. The radial range for all panels is $0.4-2.5$ a.u.

and had a median value of 3 percent. Our network is also equipped to predict planetary positions. However, we have not included a visualization of this capability because the fine-tuning simulations had fixed planets at 1 a.u.. Consequently, the network's ability to accurately determine the planet's position is technically trivial in this scenario.

Our results demonstrate that `Freesbee` is significantly faster than *FARGO* in time-evolving protoplanetary disk surface densities. `Freesbee` can emulate 5,000-year time evolution in the order of milliseconds. In contrast, *FARGO* requires approximately three hours on state-of-the-art GPU cluster hardware. `Freesbee` is at least 10^6 times faster than *FARGO*, underscoring its efficiency and potential for large-scale, high-resolution studies of protoplanetary disk dynamics.

6. Discussion and Future Work

In just milliseconds, our model can successfully emulate the 5,000 years of planetary system evolution that typically requires around 3 GPU hours to simulate using *FARGO*. Such a dramatic improvement in computational speed can facilitate more extensive parameter space explorations, enable the study of long-term disk evolution, and ultimately allow for quantitative comparisons between the theory and the observations of planet populations.

Even though the fine-tuning systems have a fixed orbit and therefore a constant planet position of 1 a.u., the planets in our pre-training systems do interact with the disk and feel torques that may push them inwards or outwards. We know planets have the ability to change their semi-major axis, making `Freesbee`'s ability to track planet positions crucial for simulating planetary migration. Thus, fine-tuning `Freesbee` with simulations depicting planet migration is a sensible next step and part of our short-term research plan.

A key functional test of `Freesbee` that we aim to implement is whether the emulated results agree with established analytical results on planet formation. For example, there is a relationship between the depth of the gap that forms in a disk and the disk/planet properties, illustrated in Figure 1 of Kanagawa et al. (2015). Therefore, an effective test would be to verify that the emulations exhibit the expected dependence of gap depth on system parameters. Such tests can verify that the emulator is accurate in a physical

sense. Moreover, although we did not observe a noticeable relationship between residuals and simulation parameters, this initial observation warrants a more extensive formal analysis.

Future work should also focus on optimizing both the input and output time-steps. Our initial choice of t_{in} was based on a visual inspection of the emulated systems when compared to simulations. We observed an overall decline in emulation quality when the model was trained with earlier time-steps. However, the aim is to strike a balance between good model performance and low computational cost. While this study serves as a proof of concept, a systematic optimization of input and output parameters is an important next step to enhance the model's predictive capabilities.

We originally did not use all of the available high-resolution simulations for fine-tuning because one of our goals is to explore the minimum number of simulations required for our model to effectively emulate planetary system evolution. Because simulations are computationally expensive, determining the minimal data requirement is crucial to reduce computational costs. While our results indicate that more training data generally improves performance, we will continue working on identifying methods to reduce resource usage without compromising accuracy.

The current implementation works on 1D, azimuthally averaged disk profiles. The complete 2D surface density contains more information about the gap and disk morphology due to non-axisymmetric features that can arise in the surface density profile of disks containing migrating planets. Thus, `Freesbee` is currently being modified to accommodate two-dimensional inputs and outputs.

Lastly, it is important to note that `Freesbee` was validated solely through cross-validation, which may have limitations in fully assessing its performance. Future work should focus on extensive testing with a broader range of simulations to provide more robust validation and further enhance the reliability and applicability of the emulator

References

Armitage, P. J. *Astrophysics of Planet Formation*. Cambridge University Press, 2 edition, 2020.

Benítez-Llambay, P. and Masset, F. S. Fargo3d: A new gpu-oriented mhd code. *The Astrophysical Journal Supplement Series*, 223(1):11, 2016. URL http://stacks.iop.org/0067-0049/223/i=1/a=11.

Bertrand, A., Camps, J., Grau, V., and Rodriguez, B. Deep learning-based emulation of human cardiac activation sequences. In Bernard, O., Clarysse, P., Duchateau, N., Ohayon, J., and Viallon, M. (eds.), *Functional Imaging and Modeling of the Heart*, pp. 213–222, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-35302-4.

Biewald, L. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Fung, J., Shi, J.-M., and Chiang, E. How empty are disk gaps opened by giant planets? *The Astrophysical Journal*, 782(2):88, January 2014. ISSN 1538-4357. doi: 10.1088/0004-637x/782/2/88. URL http://dx.doi.org/10.1088/0004-637X/782/2/88.

He, S., Li, Y., Feng, Y., Ho, S., Ravanbakhsh, S., Chen, W., and Póczos, B. Learning to predict the cosmological structure formation. *Proceedings of the National Academy of Sciences*, 116(28):13825–13832, June 2019. ISSN 1091-6490. doi: 10.1073/pnas.1821458116. URL http://dx.doi.org/10.1073/pnas.1821458116.

Jamieson, D., Li, Y., de Oliveira, R. A., Villaescusa-Navarro, F., Ho, S., and Spergel, D. N. Field-level neural network emulator for cosmological n-body simulations. *The Astrophysical Journal*, 952(2):145, July 2023. ISSN 1538-4357. doi: 10.3847/1538-4357/acdb6c. URL http://dx.doi.org/10.3847/1538-4357/acdb6c.

Jiang, Z., Tahmasebi, P., and Mao, Z. Deep residual u-net convolution neural networks with autoregressive strategy for fluid flow predictions in large-scale geosystems. *Advances in Water Resources*, 150:103878, 2021. ISSN 0309-1708. doi: https://doi.org/10.1016/j.advwatres.2021.103878. URL https://www.sciencedirect.com/science/article/pii/S0309170821000336.

Kanagawa, K. D., Muto, T., Tanaka, H., Tanigawa, T., Takeuchi, T., Tsukagoshi, T., and Momose, M. Mass Estimates of a Giant Planet in a Protoplanetary Disk from the Gap Structures. *The Astrophysical Journal Letters*, 806 (1):L15, June 2015. doi: 10.1088/2041-8205/806/1/L15.

Kasim, M. F., Watson-Parris, D., Deaconu, L., Oliver, S., Hatfield, P., Froula, D. H., Gregori, G., Jarvis, M., Khatiwala, S., Korenaga, J., Topp-Mugglestone, J., Viezzer, E., and Vinko, S. M. Building high accuracy emulators for scientific simulations with deep neural architecture search. *Machine Learning: Science and Technology*, 3(1):015013, dec 2021. doi: 10.1088/2632-2153/ac3ffa. URL https://dx.doi.org/10.1088/2632-2153/ac3ffa.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980.

Masset, F. FARGO: A fast eulerian transport algorithm for differentially rotating disks. *Astronomy and Astrophysics*

Supplement Series, 141:165–173, January 2000. doi: 10.1051/aas:2000116.

Nonnenmacher, M. and Greenberg, D. Deep emulators for differentiation, forecasting and parametrization in earth science simulators. *Journal of Advances in Modeling Earth Systems*, 13, 06 2021. doi: 10.1029/2021MS002554.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

Rouhiainen, A., Gira, M., Münchmeyer, M., Lee, K., and Shiu, G. Super-resolution emulation of large cosmological fields with a 3d conditional diffusion model, 2024.

Timpe, M. L., Han Veiga, M., Knabenhans, M., Stadel, J., and Marelli, S. Machine learning applied to simulations of collisions between rotating, differentiated planets. *Computational Astrophysics and Cosmology*, 7(1), December 2020. ISSN 2197-7909. doi: 10.1186/s40668-020-00034-6. URL http://dx.doi.org/10.1186/s40668-020-00034-6.

Zormpas, Apostolos, Birnstiel, Tilman, Rosotti, Giovanni P., and Andrews, Sean M. A large population study of protoplanetary disks - explaining the millimeter size-luminosity relation with or without substructure. *Astronomy and Astrophysics*, 661:A66, 2022. doi: 10.1051/0004-6361/202142046. URL https://doi.org/10.1051/0004-6361/202142046.

A. Architecture Parameters

Layer	Channels	Kernel	Stride	Padding
conv_enc11	3-64	3	1	1
conv_enc12	64-64	3	1	1
maxpool1	64-64	2	2	0
conv_enc21	64-128	3	1	1
conv_enc22	128-128	3	1	1
maxpool2	128-128	2	2	0
conv_enc31	128-256	3	1	1
conv_enc32	256-256	3	1	1
maxpool3	256-256	2	2	0
conv_enc41	256-512	3	1	1
conv_enc42	512-512	3	1	1
maxpool4	512-512	2	2	0
conv_enc51	512-1024	3	1	1
conv_enc52	1024-1024	3	1	1
upconv1	1024-512	2	2	0
conv_dec11	1024-512	3	1	1
conv_dec12	512-512	3	1	1
upconv2	512-256	2	2	0
conv_dec21	512-256	3	1	1
conv_dec22	256-256	3	1	1
upconv3	256-128	2	2	0
conv_dec31	256-128	3	1	1
conv_dec32	128-128	3	1	1
upconv4	128-64	2	2	0
conv_dec41	128-64	3	1	1
conv_dec42	64-64	3	1	1
conv_dec51	64-1	1	1	0

The top (bottom) of the table describes the encoder (decoder) configuration.