Department: Head
Editor: Name, xxxx@email

DLA-VPS: Deep Learning Assisted Visual Parameter Space Analysis of Cosmological Simulations

Cheng Sun
National Taiwan Normal University

Ko-Chih Wang
National Taiwan Normal University

***Abstract*—Cosmologists often build a mathematics simulation model to study the observed universe. However, running a high-fidelity simulation is time-consuming and thus can inconvenience the analysis. This is especially so when the analysis involves trying out a large number of simulation input parameter configurations. Therefore, selecting an input parameter configuration that can meet the needs of an analysis task has become an important part of the analysis process. In this work, we propose an interactive visual system, which efficiently helps users understand the parameter space related to their cosmological data. Our system utilizes a GAN-based surrogate model to reconstruct the simulation outputs without running the expensive simulation. We also extract information learned by the deep neural network-based surrogate models to facilitate the parameter space exploration. We demonstrate the effectiveness of our system via multiple case studies. These case study results demonstrate valuable simulation input parameter configuration and subregion analyses.**

THE INTRODUCTION

Exploring and analyzing the parameters of the initial condition of our universe is one of the most important tasks in cosmology, one that can help cosmologists to study the formation and physical constants of our habitable universe or build a more precise universe model [18]. Cosmologists often develop computer simulations to simulate the universe in relation to particular initial parameter configurations that they are interested in. By frequently running the simulation with initial parameter configurations and studying the simulation outputs in detail, scientists can obtain a clear picture of how initial parameter configurations affect the universe's formation and even answer critical scientific questions. However,

when the sizes of the parameter study, and of the resultant simulation outputs become too large, such an analysis can become a computationally prohibitive task.

We worked with cosmologists to develop an interactive visual analysis system which can assist cosmologists in visualizing and analyzing their cosmological simulations. Nyx [1] is a widely used cosmological simulation developed by Lawrence Berkeley National Laboratory. Nyx takes a multi-dimensional parameter configuration as input to simulate the evolution of the universe and produce multi-variant simulation outputs. Our domain experts are interested in studying the impact of parameter configuration on simulation output and in finding out parameter configurations that can produce desired simulation outputs, such as the best parameter configuration for producing outputs that match the current state of the observable universe. To accomplish this task, scientists have to identify crucial initial parameter configurations by repeatedly running the simulation and analyzing the simulation outputs. A common practice is to find the parameter configurations by running a low-resolution simulation first before running the computationally expensive high-resolution simulation. However, even such low-resolution simulations still require several minutes to hours of execution time on a supercomputer, and so represent an obstacle in terms of the scientists' analysis tasks. A popular and efficient way to mitigate this problem is to build an emulator which can quickly mimic the simulation on the commodity computer. Experts can interactively explore the parameter space on a commodity computer to minimize the requirement of running the actual simulation on the supercomputer. For example, Hazarika et al. [9] developed an emulator to emulate a yeast cell polarization simulation.

This work proposes a deep-learning-assisted visual analysis system, which utilizes a generative adversarial network (GAN) to emulate Nyx simulation for conducting efficient data exploration and analysis. After the emulator is trained by the data of sampled parameter configurations, the emulator can efficiently predict 3D simulation output of arbitrary parameter configurations. In addition, we also estimate the parameter sensitivity and provide the metric to efficiently compare multi-variant simulation output by opening the black box of the trained emulator. In the end, we develop an interactive visual analysis tool based on the GAN-based emulator and the information extracted from the emulator. Through the tool, experts can get a clear picture of the parameter space, compare data of different parameter configurations in detail, and efficiently locate the parameter configurations that have the potential to produce the desired simulation outputs.

Related Works

Visual Exploration of Parameter Space

Parameter space exploration is to study the relationship between the simulation parameters and the simulation outputs. Wang et al. [19] conducted a systematic survey which covers the analysis of parameter space. Bruckner and Möller [2] proposed a visual system to help visual effect designers search for the best fit parameter configuration for their desired explosion animations. Demir et al. [4] developed a bidirectional link of multi-charts and volume visualization for users to compare 3D scalar ensemble data visually. Wang et al. [20] proposed a nested parallel coordinates plot to analyze the parameters of 2D climate ensemble datasets visually. Orban et al. [17] projected the input parameters and output data into 2D space and then developed an interactive visual system whereby users could explore the impact of input space on output space. Compared with the work in these studies, our work not only enables the visualization of the relationships between the input and output space, but also focuses on guiding the users to find valuable multi-dimensional parameter configurations.

Surrogate Model

The purpose of building a surrogate model (emulator) is to efficiently compute the outputs of a complex simulation and so facilitate scientists' analysis workflow. There are several classic algorithms used for designing surrogate models: e.g., radial basis function models, support vector regression models, and Gaussian process models [3], [13], [16]. Because of the rapid rise in the use of neural networks in the past decade, some surrogate models have been built on deep learning techniques. He et al. [11] proposed InSituNet to synthesize 2D visualization results of simulation outputs directly. This work also extracted extra

information about the sensitivity of parameters for 2D image subregions from the deep learning model to facilitate the data analysis. Hazarika et al. [9] proposed a system called NNVA. The deep learning model in NNVA can predict a protein described by 400 values by taking 35 parameter inputs. We not only develop a machine learning-based surrogate model to efficiently predict 3D simulation data but also calculate the sensitivity of simulation parameters for 3D raw data to assist in the parameter exploration.

Deep Learning for Visualization

In the past few years, the scientific visualization community has started to utilize deep learning to solve visualization problems. Han et al. [8] used a recurrent generative network to perform temporal super-resolution for time-varying scientific datasets. Guo et al. [6] proposed a deep learning framework that produces the spatial super-resolution of vector field datasets. He et al. [10] used a convolution-based discriminative network to produce collective comparisons between two ensemble members. Han et al. [7] proposed streamline and stream surface clustering, filtering, and selection through an autoencoder [12]. The above studies looked at various different types of scientific datasets under various different data exploration scenarios. Our work focuses on developing a deep-learning-based technique for 3D ensemble dataset synthesis and for facilitating efficient simulation parameter space exploration.

Background Review

Nyx cosmological simulation [1] uses N-body and gas dynamics code to simulate the universe evolution of a given multi-dimensional input parameters. The simulation results can be represented as quantities volumes. There are seven quantities in Nyx: density of dark matter particles (*density*), X-momentum (*xmom*), Y-momentum (*ymom*), Z-momentum (*zmom*), internal energy of the gas (*rho_e*), temperature (*temp*) and gravitational potential (*phi_grav*). Nyx is able to produce data with different resolutions, according to the scientists' requirements. In the data analysis pipeline, experts may use a number of different tools to explore the data produced at different stages. Classic scientific visualization techniques, such as volume rendering, are applied to present the big picture regarding the datasets and so make hypotheses. Ultimately, domain tools such as power spectrum computation [5], are required to verify the hypothesis. However, to have a clear picture of how input parameters affect the simulation results is not a trivial task.

Requirements Analysis and Overview

In the course of this project, we have interacted with three experts in the field. They mentioned several unfulfilled requirements and inconvenient situations within the current data analysis pipeline. From the discussions, we defined the goal of our work as being to facilitate parameter exploration on commodity computers to be performed before running a computationally expensive simulation on a supercomputer in order to make the final data analysis decision. We summed up their opinions and so developed the requirements for our system.

- R1 Generate simulation outputs quickly for analysis on the commodity computer. The domain experts often submit a computing job to the supercomputer, wait for the simulation to be run, and then move the data back to the commodity computer. This workflow, switching between the supercomputer and the computer for data analysis and the consequent delay is not convenient.
- R2 Provide a preliminary overview of the multi-dimensional parameter space. The domain experts indicated that they would like to have a convenient interface whereby to select the multi-dimensional parameter configurations they are interested in and an intuitive way to observe what parameter configurations have been visited in the multi-dimensional parameter space.
- R3 Be able to perform a detailed sensitivity analysis of parameter configurations for different quantities and subregions. The domain experts are often interested in where the domain properties, such as halo count derived from density quantity, change quickly. For example, a query might be, which parameter configuration could result in significant halo count changes in which spatial subregions. A

tool to help them find these parameter configurations, quantities, and subregions would facilitate their data analysis tasks.

R4 Allows scientists to use various visualization techniques to review predicted simulation outputs quickly. Although experts must use the analysis results from the domain tools to prove hypotheses in terms of their research community ultimately, these experts have nevertheless suggested that basic visualization techniques should be present in our system. This is because visual comparison is still efficient and effective in the intermediate stages of data analysis tasks.

Based on these requirements, we have designed a deep-learning-assisted visual analysis system to facilitate the pipeline related to cosmology-data analysis.

GAN-based Surrogate Model

We propose a GAN-based surrogate model which emulates the simulation quantity volumes. Our proposed GAN consists of two sub-networks: the generator, G, and the discriminator D. In the inference stage, we take only G as our surrogate model having the ability to predict the simulation quantity volume of given arbitrary parameter configurations. We also analyze our trained GAN-based surrogate model to extract valuable information and facilitate the data analysis.

Network Architecture

Generator As shown in Fig. 1(a), the generator, G, takes the simulation parameters as input and predicts the simulation quantity volume $\hat{V}$. The simulation parameters will first go through a collection of fully connected layers (FCs). Each FC performs a matrix multiplication of its weights and input. This calculation represents a transformation of the feature space that encodes the input parameters into a latent vector space. The latent vector is reshaped into a low-resolution 3D matrix and the following residual blocks (RB_G) upscale the matrix to desired resolution. The purpose of using residual blocks is to prevent the gradient vanishing problem. In the end, we apply a tanh function to normalize the value of the output quantity volume into $[-1,1]$.

Discriminator Our proposed discriminator, D, not only identifies how close to the actual simulation results a given volume is, but also identifies whether the predicted volume matches the simulation parameter configuration. As shown in the illustration Fig. 1(b), the discriminator takes the predicted/real volumes and the corresponding simulation parameter configuration as inputs, and then outputs a scalar value. D downsamples the input volume by using several convolution-based residual blocks (RB_D) to extract the spatial features as a feature vector. Meanwhile, D also encodes the simulation parameters into a latent vector by using multiple fully connected layers. Following the approach proposed by Miyato et al. [15], the feature vector and the latent vector are integrated into a scalar value. This scalar value then becomes the discriminator output, which indicates the likelihood of the input volume as a valid simulation output.

Loss Function

In this section, we introduce L_D and L_G, which represent the loss functions of the generator and the discriminator, respectively.

Adversarial loss The purpose of calculating the adversarial loss is in relation to the volume $\hat{V}$ that is predicted by the generator and the volume V that is derived from the real (simulation output) data distribution. To be more precise, D is to differentiate two probability distributions represented by two sets of samples. We use Wasserstein distance to measure the divergence between two given distributions. Therefore, the adversarial loss of discriminator L_{adv_D} is defined as:

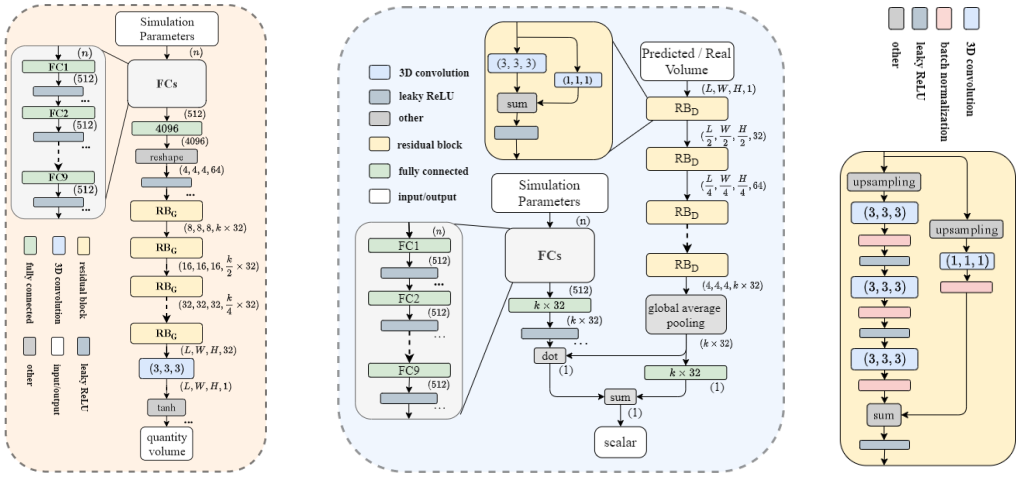
$$\begin{aligned} L_{adv_D}(V,\hat{V}) &= \mathbb{E}_{v\sim V}[D(v)] - \mathbb{E}_{\hat{v}\sim\hat{V}}[D(\hat{v})] \\ &= \mathbb{E}_{v\sim V}[D(v)] - \mathbb{E}_{p\sim P}[D(G(p))] \end{aligned} \quad (1)$$

where p is sampled from the parameter set P and $\hat{V}$ is the data generated by the generator, $G(p)$.

As a competitor of D, the mission of G is to decrease the divergence between generator distributions and real data distributions. Therefore, the adversarial loss of the generator L_{adv_G} takes the additive inverse of the second term in Equation 1:

$$L_{adv_G}(P) = \mathbb{E}_{p\sim P}[D(G(p))] \quad (2)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCG.2022.3169554, IEEE Computer Graphics and Applications



(a) (b) (c)

Figure 1: (a) The architecture of the generator. (b) The architecture of the discriminator. (c) The architecture of the residual block.

where p is sampled from parameter set P.

Volumetric loss Utilising only adversarial loss during training would cause the divergence between the generator distribution and real data distribution to get smaller, but, nevertheless, the value of the data derived from each distribution would not necessarily be the same. Therefore, we employ L2 distance as part of the loss in L_G to minimize the voxel-wise error.

$$L_V(P,V) = \mathbb{E}_{p\sim P, v\sim V}[\| G(p) - v \|_2] \qquad (3)$$

where $\| . \|_2$ denote L_2 norm.

Finally, the loss functions of the generator L_D and the discriminator L_G can be summarized as:

$$\begin{aligned} L_D &= L_{adv_D}(V,\hat{V}) \\ &= \mathbb{E}_{v\sim V}[D(v)] - \mathbb{E}_{\hat{v}\sim\hat{V}}[D(\hat{v})] \\ L_G &= \alpha L_{adv_G}(P) + \beta L_V(V) \\ &= \alpha(\mathbb{E}_{p\sim P}[D(G(p))]) \\ &\quad + \beta(\mathbb{E}_{p\sim P, v\sim V}[\| G(p) - v \|_2]) \end{aligned}$$

where α and β are hyperparameters whereby L_G can control the relative importance of perceptual similarity and feature similarity.

Deep Learning Analysis

In this section, we introduce how our system extracts the information learned by the deep learning model to facilitate the work of parameter space exploration.

Latent Feature Descriptor To understand the parameter space, it is essential to quantify the similarity of pairs (p_i, p_j), where p_i and p_j are arbitrary parameter configurations. In addition, an quick similarity computation among many parameter configurations is required to prevent from hindering the data analysis. To fulfill the goal, we utilize the knowledge learned by the generator in training. As discussed in Section "GAN-based Surrogate Model", G will learn to map the input parameter into a latent vector space and then reconstruct the simulation output. Thus, this latent vector could be seen as the corresponding parameter representation in a high-dimensional space; this, we call the *latent feature descriptor*. Once the training is completed, the parameter configurations which can produce similar outputs should be closer in latent space, and vice-versa. We take the output of the last fully connected layer (a 4096 dimensions vector) in G as the descriptor of the input parameter. The latent vector space represents a compact representation of the output space. Therefore, by calculating the Euclidean distance between two descriptors, we can measure the similarity between two given parameter configurations. Besides, since Nyx has seven quantity volume outputs, our experts are also interested in comparing the similarity across all these quantities. Thus, we concatenate each descriptor of seven quantities into a single *mix-descriptor* ($4096 \times 7 = 28672$ dimensions). We

then utilize this mix-descriptor to visualize high-level similarity across quantities in our interactive visual system.

Sensitivity Analysis Sensitivity analysis also plays an important role in parameter space exploration. It indicates the impact on the output when a parameter is changed slightly. Since our generator, G, is a differentiable function, we can calculate the partial derivative of the output with respect to the input using a back-propagation algorithm. Specifically, given an arbitrary parameter configuration $P_k = \{p_0, p_1, ...p_n\}$, where p_i $(i \in [0,n])$ represents an input parameter, G is performed to produce the predicted simulation output V_k, i.e. $G(P_k) = V_k$, where V_k is a $W \times H \times L$ volume. V_k can also be denoted as $V_k = \{s_0, s_1, ..., s_m\}$, where s_j is the value of the j-th spatial location of V_k, $j \in [0,m]$. Then, the gradient calculation of $G(P_k)$, i.e. $\frac{\partial G}{\partial P_k}$, can be referred to $\frac{\partial s_j}{\partial p_i}$, and this indicates the sensitivity of each voxel s_j with respect to the specific parameter, p_i. We utilize this analysis to help experts to identify the regions and quantity volumes with the highest sensitivities via our interactive visual system.

Interactive Visual System

Using the framework of the GAN-based surrogate model, we developed a visual analysis system (Fig. 2) that allows users to explore a number of different parameter configurations interactively.

Parameters View

The parameters view consists of two displays, a sunburst chart, and a parameter table (Fig. 2(a)).

Sunburst Chart A visualization that provides a big picture of the high dimensional parameter space and the corresponding sensitivity is crucial. We create a tree structure by a user-defined parameter order to represent the parameter space. For each simulation parameter, we split it into several value intervals, evenly. The intervals of each parameter are considered as tree nodes on which to build a hierarchical tree structure (Fig. 3(a)).

At the beginning of the process, we directly represent the hierarchical tree as a zoomable tree. However, a classic tree plot will waste a lot of empty space. Therefore, We present this hierarchical tree as a sunburst chart in the end (Fig. 2(a2)). The root of the hierarchical tree is shown at the center of the sunburst chart and the subsequent nodes extend outwards. The order of parameters to create the tree can be interactively adjusted by dragging and moving each parameter in the legend of the sunburst chart (Fig. 2(a1)).

As shown in Fig. 3(d), each node consists of two sectors: a *parameter sector* and a *recording sector*. The parameter sector (which is narrow) uses color to indicate the parameter value of a node. Any leaf node of the hierarchical tree represents a combination of the intervals of all the parameters and so is a subspace of the parameter space. Each subspace is further sub-divided into finer-grained subspaces. The granularity of the fine-grained subspaces is determined by the experts' domain knowledge. If a fine-grained subspace contains a parameter configuration that has been registered in the system, the fine-grained subspace is considered as having been 'visited'. The visited rate of fine-grained subspaces of a subspace is shown by the length of the recording sector (Fig. 3(b) and (c)). The small blocks that surround the sunburst chart are called *overview sectors* (Fig. 3(d)). The overview sectors are used to provide a quick overview of the average sensitivity of quantities in the corresponding parameter intervals. Each layer of the overview sector represents the sensitivity of a quantity. Blue and red color indicate positive and negative sensitivity values, respectively. A high saturation color indicates a larger absolute sensitivity value.

When users click on the outermost recording sectors, two tables are displayed under the sunburst view present the parameter configurations that have/have not been visited in these parameter intervals. Users can select non-visited parameter combinations and add them to our system by clicking the RUN button.

Parameter Table Users can switch to the parameter table display (Fig. 2(a4)) by clicking the up-left button. We use the parameter table to illustrate and to allow the manipulation of the parameter configurations that have been registered onto our system. Besides, users can click the buttons at the top of the table to perform various different tasks, such as to input the parameter configurations from a file, to clear all existing data

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCG.2022.3169554, IEEE Computer Graphics and Applications

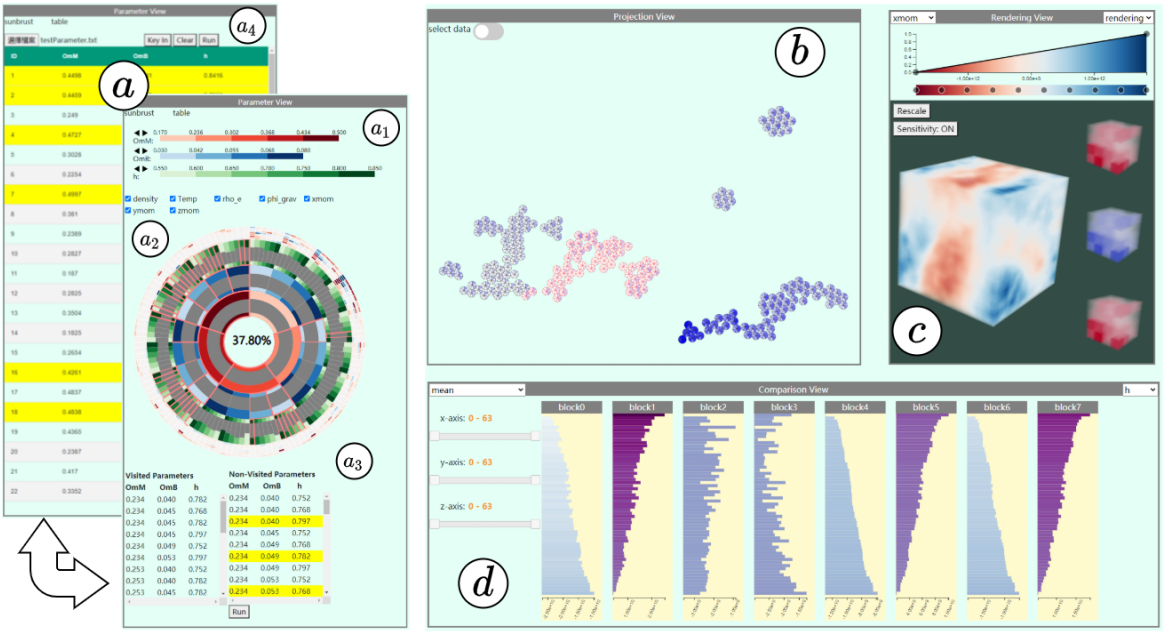


Figure 2: The interface of our visual analysis system. (a) Parameter view that showing the information of the simulation space: (a1) legend of the sunburst chart; (a2) interactive sunburst chart; (a3) visited and non-visited parameter tables; (a4) a table to show the registered parameter configurations. (b) Projection view that showing the dissimilarity of the registered parameter configurations' descriptors. (c) Rendering view that showing the rendering result of the predicted simulation output. (d) Comparison view which shows the comparison between the different statistics derived from each sub-region of the parameter configuration selected in the projection view.

in our system, and to key in a specific parameter configuration.

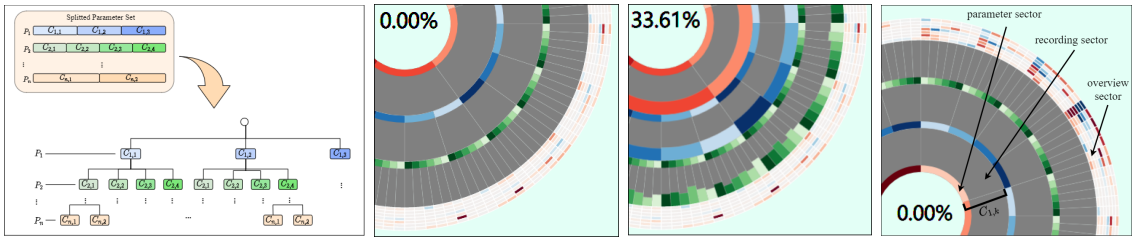
Projection View

Identifying parameter configuration clusters with similar simulation outputs and comparing their quantities in detail are important tasks. Because the quantities can be represented by a mix-descriptor that has been introduced in Section "Deep Learning Analysis", this task becomes a high-dimensional vector relation visualization task. UMAP [14] is used as our dimensional reduction technique because UMAP can project new data into 2-dimensional space without changing the existing positions of data points. This allows users to interactively add new parameter configurations into the projection view. We display the results of the UMAP process using a scatter plot to indicate the similarity of the simulation outputs relating to the given parameter configurations. The distance between two arbitrary points on the projection view shows the similarity that sums up the multiple quantities.

Because users are usually interested in comparing quantities of similar simulation outputs, we only show the difference in the quantity level when users lasso a group of the points by the mouse. The average descriptors are the *representative descriptors* of quantities of the selected group. We calculate the Euclidean distance between the descriptors of each quantity across all points and the representative descriptor. The distances are used to indicate the differences of quantities between a point and the selected group. We split each point evenly in the projection view into seven sectors, which we call a *quantity pie*, because we have seven quantities. Fig. 4 shows the notation of the quantity pies and the interaction of the projection view.

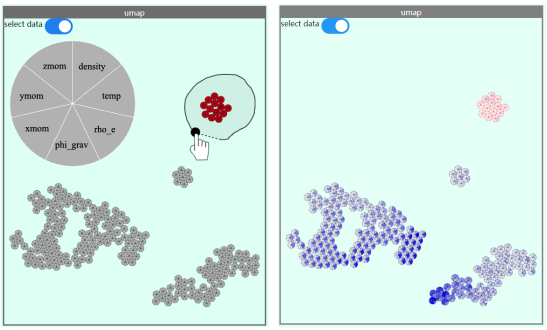
Comparison View

Our experts are interested in understanding the impact on subregions of simulation outputs if the parameter configuration changes and the



(a) (b) (c) (d)

Figure 3: (a) An example of the hierarchical tree. P_i is a parameter. $C_{i,k}$ is the k-th interval of P_i. A path from the root to any leaf node represents a combination of parameters. The images (b) and (c) are examples of the recording sector. (b) The recording sector while the overall visited rate is 0%. (c) The recording sector while the overall visited rate is around 33%. (d) The notations of our sunburst chart.



(a) (b)

Figure 4: The interactions facilitated on the projection view. (a) Users can lasso quantity pie-charts by dragging the mouse. The enlarged quantity pie in (a) shows the represented quantity in each sector. (b) This shows the differences between parameter configurations across quantities when the mean of lassoed pies (red contour) is used as the representative descriptor. A high color saturation sector indicates that the corresponding quantity has a more significant difference from the selected group.

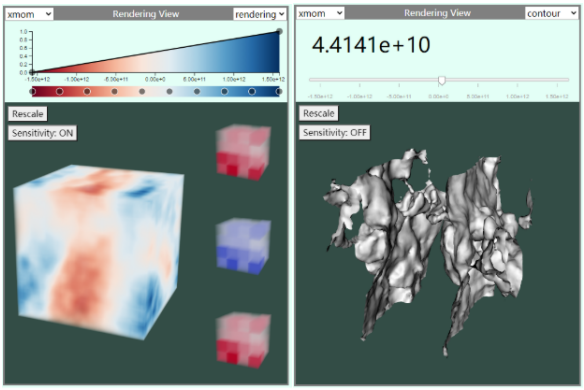
difference among subregions. Therefore, a bar chart is chosen to show the relation between the parameter value and data values of a subregion. Besides, we juxtapose bar charts of subregions for users to observe the difference among subregions.

Users can select parameter configurations by lassoing quantity pies in the projection view and then selecting a spatial region that they are interested in by entering 3D coordinates (the three sliders on the left) in *comparison view* (Fig. 2(d)). The selected subregion is then evenly divided into eight sub-blocks. The statistical values (e.g., max, mean, standard deviation, or sensitivity) relating to each selected parameter configuration in each block are computed and shown by bar charts. The length and value of bars represent the normalized value within each spatial block and the absolute value, respectively. The bars with the same vertical position are the values from different spatial blocks with the same parameter configuration. Users can select different statistical values from the top-left drag-down list to show and select a parameter from the top-right drop-down menu to reorder the bars along with the vertical position.

Rendering View

The *rendering view* (Fig. 2(c)) is used to visualize the output volume of a selected parameter configuration. After users select the parameter configuration they are interested in by clicking a parameter configuration displayed in the parameter table or a quantity pie displayed in the projection view, the backend surrogate model will generate the predicted volume for volume rendering or isosurface visualization. Users can specify a visualization technique to visualize a quantity from the drop-down lists (Fig. 5). In addition, we evenly divide the output volume into multiple sub-blocks and compute the sensitivity of each block. The sensitive volumes are placed at the right-hand side of the output volume with the order of parameters in the sunburst from top to bottom. The sensitivities are color-coded. The color map is the same as that used in the overview sectors in the section, 'Parameter View'.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCG.2022.3169554, IEEE Computer Graphics and Applications



(a) (b)

Figure 5: The rendering view in our system. Users can change the visualization method used by selecting from the drop-down list at the top righthand corner. (a) The volume rendering result. (b) The isosurface result.

Implementation and Performance

According to the experts we discussed the matter with, *Omega_m* (total matter density), *Omega_bar* (total density of baryons) and *h* (hubble constant) are the parameters they would most like to investigate. Because of cosmologists' efforts in the past, the known possible value ranges of these parameters have been shrunk to ever-smaller intervals. Thus, the domain ranges of interest are $0.55 < h < 0.85$, $0.17, < Omega\_m < 0.5$, and $0.03 < Omega\_bar < 0.08$. By considering the trade-off between the cost of the training data collection, the training time, and the prediction time on the commodity computer, plus the data quality for the parameter exploration task, the resolution $64 \times 64 \times 64$ is suggested for use in our system. The surrogate model was trained using 800 outputs of sampled parameter configurations from the Nyx simulation because the performance of the model does not significantly improve after collecting more simulation outputs than this. We also randomly collected an extra 200 outputs as a testing set to evaluate the model performance. Each simulation run with the resolution $64 \times 64 \times 64$ requires around 3 minutes of super-computer time. We trained the models on the Taiwan Computing Cloud with a single NVIDIA Tesla V100 GPU, and each model took around one day for training. We also measured the time taken by our deep learning model to generate a set of data on a commodity computer with an Intel Core i7-8700 and an NVIDIA 2060 GPU. The average data generation time for a single output quantity, derived from a parameter configuration, was 0.05 seconds. Note that the parameters are real values and users could be interested in any parameter configuration. Our emulator can predict not only simulation outputs of the parameter configurations used to train the emulator but also simulation outputs of parameter configuration from the parameter intervals of interest. To evaluate the quality of the output data generated from our model, we used the peak signal-to-noise ratio (PSNR) and the structural similarity index measures (SSIM) as quality metrics. We also implemented the surrogate model used by Hazarika et al. [9] to generate 3D cosmology data, for comparison purposes. Table 1 shows that our model yielded better PSNR and SSIM values with respect to all seven quantities. Even for *phi_grav*, our model still gets pretty high PSNR and SSIM values. The source code of the surrogate model and the visual tool are available at github.com/andy1213aa/DLA-VPS_3dGAN and github.com/andy1213aa/DLA-VPS_Visualization, respectively.

Use Cases

In this section, we introduce the use cases of our interactive system.

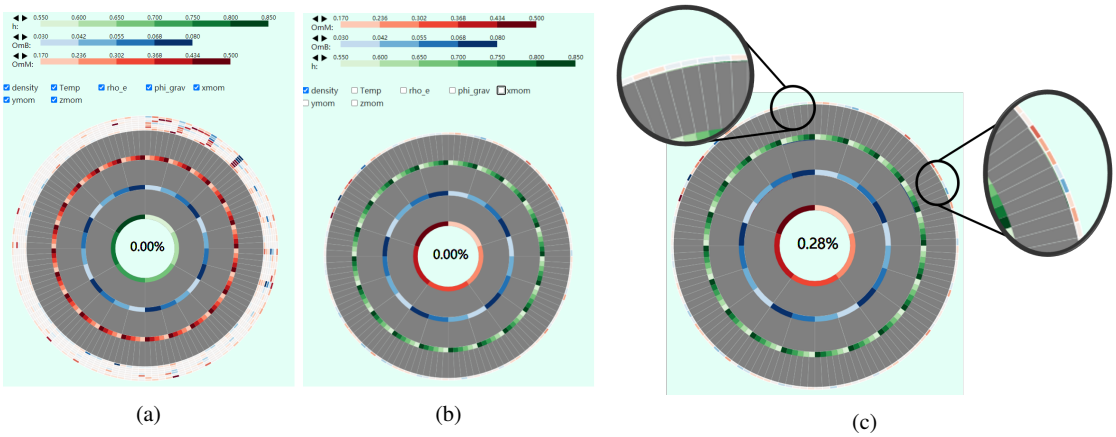
Parameter Sampling Strategy Calibration

In this first use case, the user wishes to calibrate their parameter sampling strategies using our system. A good sampling strategy is important for these kinds of study because this can help users achieve the goal of their data analysis with fewer simulation runs. However, the task of creating a good strategy is not easy because it relies on a deep understanding of the relation between the input (of multiple parameters) and the output (of multiple quantities). So, the user would wish to understand the level of difference within and among parameter intervals.

The user starts the exploration from Fig. 6(a). The overview sectors in Fig. 6(a) illustrate the sensitivity of *h* (hubble) to each parameter interval. The user first pays attention to density quantity and unchecks the other quantities. By dragging the legend to the reorder parameters area in the

Table 1: The performance of our GAN-based surrogate models. PSNR is directly computed from 3D data. The SSIM is calculated from volume rendering images of ten random viewpoints with a fixed transfer function.

	density		Temp		rho_e		phi_grav		xmom		ymom		zmom	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
DLA-VPS	**38.301**	**0.922**	**41.883**	**0.911**	**40.723**	**0.983**	44.679	0.990	**39.159**	**0.960**	**26.969**	**0.641**	**31.234**	**0.879**
Hazarika et al. [9]	37.163	0.893	38.005	0.910	33.771	0.952	**47.958**	**0.995**	20.192	0.741	10.554	0.572	21.308	0.737



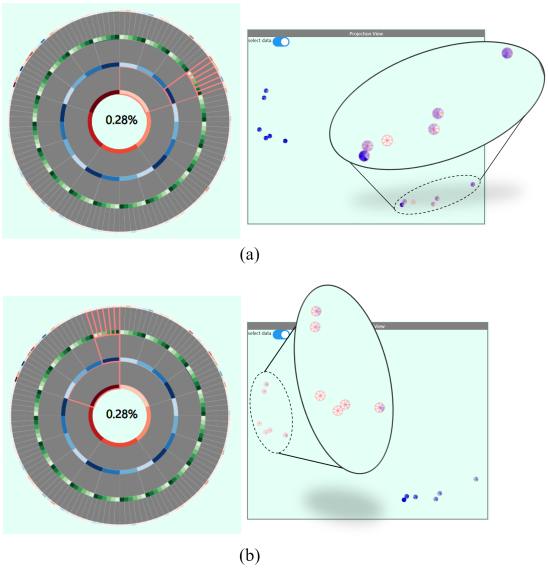
(a) (b) (c)

Figure 6: (a) and (b) are the sunburst charts with different settings: (a) is the normal setting, (b) focuses on density, (c) the parameter intervals at eleven o'clock are low sensitivity. In addition, the parameter intervals in the two o'clock direction are high sensitivity.

sunburst plot, the user can observe which overview sectors have high sensitivity aggregated together and form a few clusters when putting the *h* (hubble) to the outermost region (two o'clock and eleven o'clock directions in Fig. 6(b)). The user decides to attempt to obtain insights into the high sensitivity parameter intervals at the two o'clock direction first and then to pick up the low sensitivity parameter intervals in the eleven o'clock direction for reference. From the non-visited table, the user selects one parameter configuration from each parameter interval with different hubble values, while the other parameters *Omega_m* and *Omega_bar* are fixed. Then, our system runs the surrogate model and adds these parameter configurations to the projection view (Fig. 7). By lassoing the quantity pies from the high sensitivity intervals, the user observes not only that the *density* quantity is dissimilar but that the other quantities are also (Fig. 7(a)). Therefore, no matter what quantities are studied, more parameter configuration samples may be required for these parameters intervals. By lassoing the quantity pies from the low sensitivity intervals, the user finds that most of the sector colors of the quantities pies are the same, but that those of the quantities *temp* and *rho_e* are now showing a high similarity (Fig. 7(b)). Therefore, fewer parameter configuration samples may be sufficient when studying most of the quantities in this parameter interval. If the user wants to focus on studying the quantities *temp* or *rho_e*, it should be considered that the parameter configuration sample count should be increased.

In order to further verify the hypothesis visually, the user clicks on the quantity pies one by one to display the visualization results of the different quantities for each parameter configuration in the rendering view (Fig. 9). The user then confirms his hypothesis from the projection view by observing the differences in the visualizations yielded by the differing parameter configurations. The user repeats the above steps to study other parameter configurations and output quantities by observing the recording sector to find which parameter subspaces have not been visited sufficiently. Thus the domain expert can iteratively tune their parameter configuration sampling strategies.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCG.2022.3169554, IEEE Computer Graphics and Applications



(a)

(b)

Figure 7: The parameter intervals in the eleven o'clock direction are low sensitivity. The parameter intervals at the two o'clock are high sensitivity.

Spatial Subregion Analysis

After knowing the high sensitivity parameter intervals for the whole *density* quantity volume, the user might like to further explore what combination of spatial subregion and local parameter intervals have a relatively higher sensitivity. In this regard, the halo count is always what is of most interest because many domain questions can be answered by the halo count estimation. A dramatic density value change indicates a possibly dramatic halo count change in a local region. By applying this observation, users can focus on the particular spatial subregion and local intervals of parameters and this can result in an efficient data analysis.

Therefore, the user first investigates the subregion sensitivity visualization (Fig. 10) in the rendering view to find the high sensitivity subregions of a few selected parameter configurations. The user notices, in this case, that a corner always shows relatively high sensitivity (red color) even if the user changes the value of the parameter h (hubble). Thus, the user might well wish to explore the pattern of density values of different subregions and the parameter h with more details. The user asks the system to randomly select 200 parameter configurations with different h values but fixed $Omega\_m$ = 0.198 and $Omega\_bar$ = 0.074 in the high sensitivity intervals that we observed in the first use case. Fig. 8 shows the mean density value of each parameter configuration in each spatial sub-block. The user has found that the density values in the subregion, block0, are affected by h most (2×10^{10} down to 1.6×10^{10}) by observing the colors of the bars in block0. This block is also the high sensitivity corner in the subregion sensitivity visualization. In addition, although the density value change is nowhere as dramatic as that in block0, most of the subregions still have significant density value changes around h = 0.163. This observation gives the user a hint that it is necessary to spend more effort on the data around h = 0.163 if the answers to the data domain questions are indeed affected by the halo count.

Discussion and Future Work

Visual System

One limitation of the current design of the visualization system is the scalability in terms of the number of input parameters that can be used. If the number of parameters goes beyond five or six, the outer sections may not be readable. When a larger number of parameters is required, a zoomable sunburst implementation could be used to relieve this problem. In a zoomable sunburst, the center of the sunburst would always represent a node in the parameter hierarchical tree (but not necessarily the root) and would show a fixed number of levels (e.g., 3 or 4) from this.

Surrogate Model

The proposed surrogate model has the potential to be used on a wide range of simulations if the simulation is deterministic and the outputs are scalar fields. If the simulation is stochastic, the current model cannot capture the randomness within the simulation. In addition, the required computational resources and the time needed to train the surrogate model are another constraint and they can be influenced by multiple factors, such as the intervals of parameters, the number of parameters, and the simulation resolution. For example, if the parameter count or the resolution increase, more training datasets may be required to maintain a sufficient performance, and this might result in a non-realistic requirement for computational power, GPU memory, and training time. Based on the observation that data from

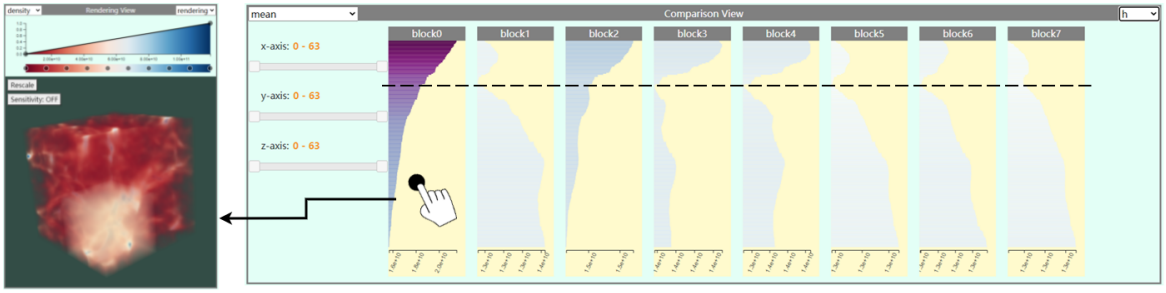


Figure 8: The mean value of the selected parameter configurations in each subregion. Users can click on the chart to see the corresponding spatial location in the rendering view.

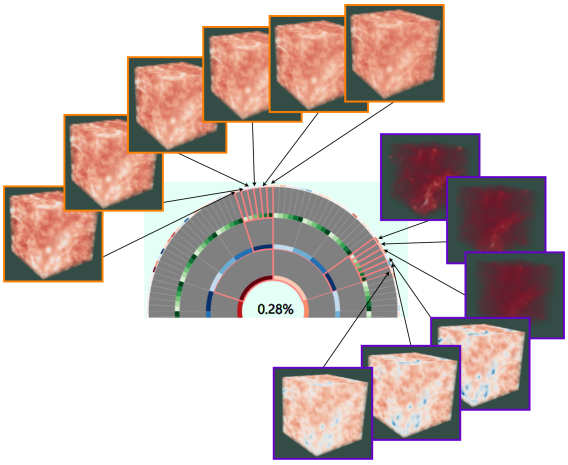


Figure 9: An example of comparing the density quantities derived from different parameter configurations. The visualizations with the same colored frames (orange or purple) use the same transfer function.

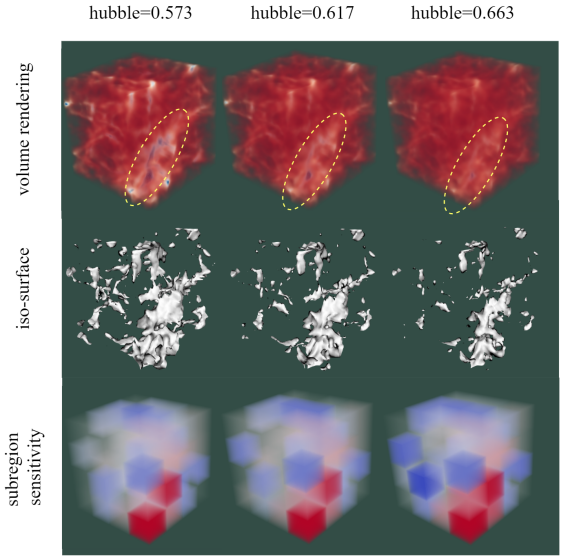


Figure 10: A comparison of the different *hubble* values in the density quantity. Top row: the volume rendering with the same transfer function. Middle row: isosurface with iso-value of 3.015×10^{10}. Bottom row: Regions influenced by the *hubble* value.

different subregions are usually similar, we could develop a transfer learning-based framework to train a surrogate model for just one subregion first and adapt it to other subregions. This framework might reduce the training requirements in terms of computational resources and time.

Conclusion

In this work, we have proposed a deep-learning-assisted visual analysis system for simulation parameter space exploration. Our system consists of two components: a GAN-based surrogate model, and an interactive visual system. The GAN-based surrogate model is trained to learn the mapping from the simulation parameter space to the 3D cosmological simulation outputs. With these trained models, we also provide several deep learning analysis techniques to assist parameter space exploration. The interactive visual system is driven by the trained surrogate models, which are designed to facilitate the data analysis process. Experts can utilize our system to explore different parameter intervals and calibrate their parameter sampling strategy.

REFERENCES

1. Ann S. Almgren, John B. Bell, Mike J. Lijewski, Zarija Lukić, and Ethan Van Andel. Nyx: A massively parallel AMR code for computational cosmology. *Astrophysical Journal*, 765(1), 2013.

2. Stefan Bruckner and Torsten Möller. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1468–1476, 2010.
3. Angelo Ciccazzo, Gianni Di Pillo, and Vittorio Latorre. A SVM Surrogate Model-Based Method for Parametric Yield Optimization. 35(7):1224–1228, 2016.
4. Ismail Demir, Christian Dick, and Rüdiger Westermann. Multi-charts for comparative 3D ensemble visualization. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2694–2703, 2014.
5. Brian Friesen, Ann Almgren, Zarija Lukić, Gunther Weber, Dmitriy Morozov, Vincent Beckner, and Marcus Day. In situ and in-transit analysis of cosmological simulations. *Computational Astrophysics and Cosmology*, 3(1):1–18, 2016.
6. Li Guo, Shaojie Ye, Jun Han, Hao Zheng, Han Gao, Danny Z. Chen, Jian Xun Wang, and Chaoli Wang. SSR-VFD: Spatial Super-Resolution for Vector Field Data Analysis and Visualization. *IEEE Pacific Visualization Symposium*, 2020-June:71–80, 2020.
7. Jun Han, Jun Tao, and Chaoli Wang. FlowNet: A Deep Learning Framework for Clustering and Selection of Streamlines and Stream Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 26(4):1732–1744, 2020.
8. Jun Han and Chaoli Wang. Tsr-tvd: Temporal super-resolution for time-varying data analysis and visualization. *IEEE transactions on visualization and computer graphics*, 26(1):205–215, 2019.
9. Subhashis Hazarika, Haoyu Li, Ko-Chih Wang, Han-Wei Shen, and Ching-Shan Chou. Nnva: Neural network assisted visual analysis of yeast cell polarization simulation. *IEEE transactions on visualization and computer graphics*, 26(1):34–44, 2019.
10. Wenbin He, Junpeng Wang, Hanqi Guo, Han Wei Shen, and Tom Peterka. CECAV-DNN: Collective Ensemble Comparison and Visualization using Deep Neural Networks. *Visual Informatics*, 4(2):109–121, 2020.
11. Wenbin He, Junpeng Wang, Hanqi Guo, Ko-Chih Wang, Han-Wei Shen, Mukund Raj, Youssef SG Nashed, and Tom Peterka. Insitunet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE transactions on visualization and computer graphics*, 26(1):23–33, 2019.
12. Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
13. Slawomir Koziel and John W. Bandler. Microwave device modeling using space-mapping and radial basis functions. *IEEE MTT-S International Microwave Symposium Digest*, 3:799–802, 2007.
14. Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.
15. Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.
16. Thong Nguyen and Jose Schutt-Aine. Gaussian Process surrogate model for variability analysis of RF circuits. *IEEE Electrical Design of Advanced Packaging and Systems Symposium*, 2020-December(6):2020–2022, 2020.
17. Daniel Orban, Daniel F Keefe, Ayan Biswas, James Ahrens, and David Rogers. Drag and track: A direct manipulation interface for contextualizing data instances within a continuous parameter space. *IEEE transactions on visualization and computer graphics*, 25(1):256–266, 2018.
18. Sohrab Rahvar. Cosmic initial conditions for a habitable universe. *Monthly Notices of the Royal Astronomical Society*, 470(3):3095–3102, 2017.
19. Junpeng Wang, Subhashis Hazarika, Cheng Li, and Han Wei Shen. Visualization and visual analysis of ensemble data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 25(9):2853–2872, 2019.
20. Junpeng Wang, Xiaotong Liu, Han Wei Shen, and Guang Lin. Multi-Resolution Climate Ensemble Parameter Analysis with Nested Parallel Coordinates Plots. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):81–90, 2017.