DOI: 10.1111/cgf.70134
Eurographics Conference on Visualization (EuroVis) 2025
W. Aigner, N. Andrienko, and B. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 3

HyperFLINT: Hypernetwork-based Flow Estimation and Temporal Interpolation for Scientific Ensemble Visualization

Hamid Gadirov[1], Qi Wu[3], David Bauer[2], Kwan-Liu Ma[2], Jos B.T.M. Roerdink[1], and Steffen Frey[1]

[1]University of Groningen, Netherlands
[2]University of California, Davis, USA
[3]NVIDIA, USA

Abstract

We present HyperFLINT (Hypernetwork-based FLow estimation and temporal INTerpolation), a novel deep learning-based approach for estimating flow fields, temporally interpolating scalar fields, and facilitating parameter space exploration in spatio-temporal scientific ensemble data. This work addresses the critical need to explicitly incorporate ensemble parameters into the learning process, as traditional methods often neglect these, limiting their ability to adapt to diverse simulation settings and provide meaningful insights into the data dynamics. HyperFLINT introduces a hypernetwork to account for simulation parameters, enabling it to generate accurate interpolations and flow fields for each timestep by dynamically adapting to varying conditions, thereby outperforming existing parameter-agnostic approaches. The architecture features modular neural blocks with convolutional and deconvolutional layers, supported by a hypernetwork that generates weights for the main network, allowing the model to better capture intricate simulation dynamics. A series of experiments demonstrates HyperFLINT's significantly improved performance in flow field estimation and temporal interpolation, as well as its potential in enabling parameter space exploration, offering valuable insights into complex scientific ensembles.

CCS Concepts

• ***Computing methodologies*** → *Flow Estimation; Interpolation; Deep Learning;* • ***Human-centered computing*** → *Spatiotemporal Data; Ensemble Parameter Space Exploration; Scientific visualization;*

1. Introduction

Advancements in simulation and capture technologies now enable the observation of time-dependent processes at extremely high spatial and temporal resolutions. This generates vast spatio-temporal datasets, offering researchers opportunities to analyze parameter variations and stochastic effects. However, the sheer data volumes often exceed storage capacities, requiring selective preservation of timesteps or variables [CBGH19]. Additionally, experimental data is constrained by specific modalities, limiting its comprehensiveness. Addressing these challenges, several recent methods were proposed for reconstructing scalar fields [HW22, WBCM23, TW24, GRF25] and estimating flow fields [GRF25] in 2D and 3D scientific data. FLINT's [GRF25] student-teacher architecture achieved state-of-the-art results in density interpolation and flow estimation, proving effective for reconstructing missing data. This is particularly valuable for applications like flow visualization [JWC*11], optimal timestep selection [FE17], and ensemble member comparisons [TFE07]. However, FLINT's and the above-mentioned recent methods' limitations in generalizing across parameter space and neglecting ensemble parameters reduced their effectiveness in dynamic simulation scenarios.

In this paper, we introduce HyperFLINT (Hypernetwork-based FLow estimation and temporal INTerpolation), a deep learning approach that uses hypernetworks to estimate missing flow fields in scientific ensembles. HyperFLINT leverages the simulation parameters via hypernetworks to generate accurate flow fields for each timestep, even in scenarios where the flow could not be captured or was omitted (see overview in Fig. 1). Additionally, HyperFLINT is capable of producing high-quality temporal interpolants between scalar fields, providing a comprehensive solution for reconstructing both flow and scalar data. HyperFLINT improves upon existing state-of-the-art methods, such as FLINT [GRF25], by focusing on scientific ensembles and incorporating hypernetworks to account for simulation parameters, resulting in better temporal interpolation and flow estimation. Unlike previous approaches, HyperFLINT handles complex spatio-temporal datasets without requiring domain-specific assumptions, pre-training, or fine-tuning on simplified datasets.

The ability of HyperFLINT to incorporate parameter-driven transformations into its model architecture opens up new possibilities for parameter space exploration. By learning the relationships between simulation parameters and output data, HyperFLINT can generate approximations of data for configurations that were not explicitly simulated, allowing for a broader investigation of possible simulation outcomes without rerunning the full simulation for

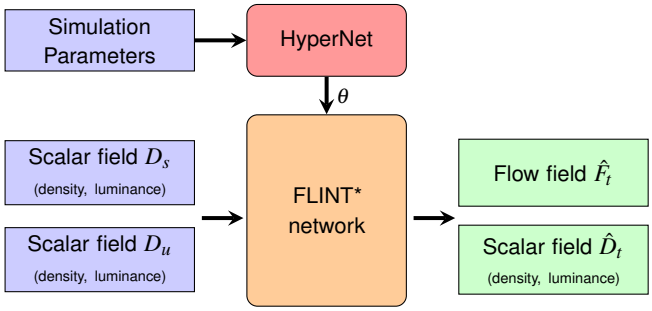


Figure 1: *Overview of HyperFLINT pipeline during inference. The FLINT* deep neural network, whose weights are generated by the HyperNet, performs flow field estimation $\hat{F}_t$ and temporal (scalar) field interpolation $\hat{D}_t$, where $s < t < u$, by utilizing the available densities D_s and D_u from the previous and following timesteps, and their simulation parameters.*

each parameter set. This capacity is particularly valuable for understanding complex dependencies between parameters in ensemble simulations, as it enables researchers to interpolate or extrapolate within the parameter space. With HyperFLINT, users can systematically explore how changes in parameters impact the generated fields, offering insights into the underlying phenomena that would otherwise require extensive computational resources to simulate directly. Moreover, HyperFLINT's parameter-driven capabilities allow for the reconstruction of missing data, including variables and timesteps, which is especially useful in large-scale simulations where only a subset of the data can be saved [CAA*20]. The HyperFLINT code will be made available publicly.

Our key contributions can be summarized as follows:

- To the best of our knowledge, HyperFLINT is the first approach that employs a hypernetwork to adaptively estimate flow fields and produce state-of-the-art temporal interpolations of density fields for scientific ensembles by dynamically conditioning on simulation parameters.
- HyperFLINT effectively handles spatio-temporal ensembles utilizing a hypernetwork, requiring no specific assumptions, making it versatile for diverse scientific applications.
- With the introduction of the hypernetwork, HyperFLINT achieves several key advancements: it dynamically generates simulation parameter-aware weights facilitating the understanding of spatio-temporal scientific ensembles, enhancing model quality and performance by adapting to varying simulation parameters, enabling significantly improved flow estimation and scalar field interpolation, even in scenarios with sparse or incomplete data.

2. Related Work

Hypernetworks in deep learning. Hypernetworks [HDL16] have gained traction in the deep learning community as an effective mechanism for generating the weights of another neural network, thereby offering greater flexibility in capturing complex relationships between data and model parameters. These networks excel in tasks where parameter space exploration is crucial, as they can dynamically adjust the parameters of a target network based on external inputs, such as simulation parameters. Hypernetworks have been successfully applied in various fields, including neural architecture search [BLRW17], generative modeling [SIE21], and meta-learning [PPT*22].

Flow field estimation in scientific visualization. Several methods have been developed for flow field estimation and visualization in scientific datasets. Kappe *et al.* [KSG*15] focused on estimating 3D local flow in microscopy data using a combination of image processing and optical flow techniques, while Kumpf *et al.* [KRRW18] applied ensemble sensitivity analysis with optical flow-based feature tracking to study changes in geo-spatial data. Manandhar *et al.* [MBW*18] proposed dense 3D optical flow estimation for microscopy image volumes using displacement vectors. However, these approaches often rely on specific assumptions, limiting their applicability to real-world scientific ensembles. In contrast, HyperFLINT operates without dataset-specific assumptions, making it versatile for diverse scientific visualization tasks. Sahoo *et al.* [SB22] introduced Integration-Free Learning of Flow Maps, which estimates flow directly from state observations, enhancing efficiency for large-scale datasets where ground-truth (GT) flow may not be available. FLINT [GRF25] emerged as a state-of-the-art approach for reconstructing scalar fields and simultaneously estimating flow fields in spatio-temporal scientific datasets. FLINT's student-teacher architecture and flexible loss function helped achieve high accuracy in density interpolation and flow estimation tasks, particularly in 2D+time and 3D+time datasets. FLINT's modular design features a series of convolutional and deconvolutional layers grouped into neural blocks, which iteratively refine scalar field and flow field outputs. Temporal consistency is enforced through specific loss components, ensuring alignment with GT data and improving interpolation accuracy. These capabilities made FLINT particularly effective for addressing missing data in large-scale simulations. However, FLINT exhibits limitations in dynamic simulation scenarios where varying ensemble parameters affect data behavior. Without mechanisms to explicitly incorporate these parameter variations, FLINT's adaptability and generalization remain constrained.

Machine learning-based upscaling & super-resolution. Across multiple fields, including image processing, computer vision, and scientific visualization [LTH*17, SCH*16], ML-based upscaling and super-resolution approaches have gained considerable attention. These techniques generally target either spatial, temporal, or combined spatio-temporal enhancements, forming three main categories: spatial super-resolution (SSR), temporal super-resolution (TSR), and spatio-temporal super-resolution (STSR). SSR approaches, exemplified by models such as SRCNN [DLHT15], SRFBN [LYL*19], and SwinIR [LCS*21], focus on increasing spatial detail by generating realistic textures and enhancing fine structures. TSR methods, on the other hand, aim to fill in intermediate frames in time-subsampled sequences without degrading spatial quality. Methods like phase-based interpolation [MWZ*15], SepConv [NML17], and SloMo [JSJ*18] are representative TSR techniques that focus on temporal resolution improvements in videos. While some prior approaches, such as STNet [HZCW21], address either spatial or temporal resolution, they typically do not tackle both simultaneously. For example, TSR-TVD [HW19], a recurrent generative network proposed by Han *et al.*, was designed to temporally upscale a variable different from the given one, without explicitly addressing both spatial and temporal dimensions. Volume scene representation networks

(V-SRN) [LJLB21, PFS*19, SZW19, MST*21] have significantly advanced neural representations for volumetric data, enabling high-quality rendering and reconstruction through implicit neural representations. Building on these advancements, fV-SRN [WHW22] leverages GPU tensor cores to integrate neural reconstruction into on-chip ray tracing kernels, reducing computational complexity and accelerating training and inference for real-time volume rendering. Recent advancements, such as Filling the Void [MHBB22], SSR-TVD [HW20], FFEINR [JBY23], HyperINR [WBCM23], CoordNet [HW22], and STSR-INR [TW24], have shown improvements in handling either TSR or SSR of data fields at arbitrary resolutions. FLINT [GRF25] can perform temporal interpolation for both 2D+time and 3D+time datasets but does not consider the parameter space of scientific ensembles, which limits its adaptability to different scenarios. In contrast, HyperFLINT introduces hypernetworks to incorporate ensemble parameters directly into the interpolation process, enabling a more adaptive and data-driven approach.

3. Method

To handle diverse parameter settings in spatio-temporal scientific ensembles, we introduce HyperFLINT, a method that integrates a hypernetwork to dynamically adapt the main neural network, FLINT*. The hypernetwork generates weights based on simulation parameters. The pipeline consists of three main components: (1) the hypernetwork (HyperNet), which takes simulation parameters as input and produces weights for FLINT*, which is a simplified and streamlined version of the original FLINT network; (2) the FLINT* network, which estimates flow fields and interpolates scalar data across time steps; and (3) a training framework that optimizes both flow estimation and interpolation through a combination of loss functions. Unlike traditional methods, HyperFLINT does not require pre-training or fine-tuning on simplified datasets, enabling efficient adaptation to new scenarios. The following sections elaborate on the neural network architecture of HyperFLINT (Sec. 3.1), describe the temporal interpolation and flow estimation pipeline used in both training and inference (Sec. 3.3), and detail our proposed loss function (Sec. 3.4). A detailed comparison between HyperFLINT and FLINT can be found in Sec. 3.5.

3.1. HyperFLINT Network Architecture

The architecture of HyperFLINT integrates two neural networks: HyperNet and FLINT*. HyperNet (Fig. 2, top), described in Sec. 3.2, dynamically generates weights for the FLINT* network (Fig. 2, bottom), which is inspired by the FLINT method [GRF25]. The architecture of the FLINT* network comprises $N = 3$ stacked convolutional blocks (*Conv Block*), each incorporating convolutional (*Conv*) and deconvolutional (*Deconv*) layers. The expanded view of a single convolutional block is presented in the middle column of the orange box in Fig. 2. The first five *Conv* layers (red outlines, middle column) dynamically adjust their weights based on the output from HyperNet. The FLINT* network starts with 128 feature channels in the first block, reduces to 96 channels in the second block, and finishes with 64 channels in the final block. PReLU activation [HZRS15] is applied to all layers except the last one, ensuring efficient learning and non-linear transformations. The training process is driven by loss components illustrated on the right side of Fig. 2, guiding both flow estimation and scalar field interpolation. FLINT* and HyperNet are trained jointly, ensuring that the hypernetwork optimizes the weights of the main network dynamically throughout training.

3.2. HyperNet

The HyperNet architecture, as depicted in Fig. 2 top part, is an integral part of HyperFLINT. It takes numerical input parameters that characterize the simulation (such as physical quantities or configuration settings) and processes them through two main components: a Multi-Layer Perceptron (MLP) and a Convolutional Neural Network (CNN). First, the MLP, consisting of three linear layers with PReLU activation functions and two dropout layers, transforms the input parameters into a higher-dimensional representation. This embedding allows the network to interpret the simulation parameters in a way that is beneficial for downstream tasks. Once processed by the MLP, the intermediate representation is passed into the CNN, where additional transformations are applied, refining the information derived from the parameters. Finally, the output of the CNN is reshaped into a one-dimensional vector and passed through a final linear layer to generate the weights (θ) for the convolutional layers of the FLINT* network. The choice of one-dimensional convolutions (Conv1D) within the CNN component is motivated by the fact that the input simulation parameters are represented as one-dimensional sequences. Conv1D layers provide the most efficient way to capture local relationships between the parameters, preserving their spatial or sequential structure. Compared to fully connected layers, Conv1D can model these relationships in a way that is more computationally efficient, with fewer parameters, and can capture local patterns in parameters that are missed by dense layers. This structured approach helps the HyperNet leverage the inherent relationships between parameters to refine the generated weights. The proposed architecture enables parameter-aware learning, where the simulation's governing parameters directly influence the model's internal weights. This not only improves the model's quantitative and qualitative performance but also facilitates parameter space exploration (Sec. 7).

3.3. Flow Estimation and Scalar Field Interpolation

HyperFLINT takes as input the simulation parameters (for HypetNet), two scalar fields D_s and D_u of the same ensemble member at timesteps $s < u$, and an intermediate timestep t, where $s < t < u$ (for FLINT*). The goal is to predict the corresponding flow field $\hat{F}_t$ and generate interpolated scalar fields $\hat{D}_t$ for any intermediate time $t \in [s,u]$. To accomplish this, FLINT* first computes intermediate flow fields, $\hat{F}_{t\to s}$ and $\hat{F}_{t\to u}$. The *time-backward* flow field, $\hat{F}_{t\to s}$, represents the flow vectors from the frame at time t to an earlier frame at s. Conversely, the *time-forward* flow, $\hat{F}_{t\to u}$, represents flow vectors from the frame at t to a later frame at u. These intermediate flow fields are then used to warp scalar fields towards the target time t, generating estimates for that time step. In the final step, HyperFLINT combines the intermediate warped scalar fields using a fusion mask M learned by FLINT*, where $M(i,j) \in [0,1], \forall i,j$, which ensures smooth blending and high-quality interpolation.

Warping. In HyperFLINT, we implement the volumetric *backward warping*, where each target voxel v_t identifies its corresponding source voxel v_s based on the flow fields. This mapping, guided by the intermediate flow fields, ensures smooth resampling using trilinear interpolation around v_s to compute the value for v_t. We denote the

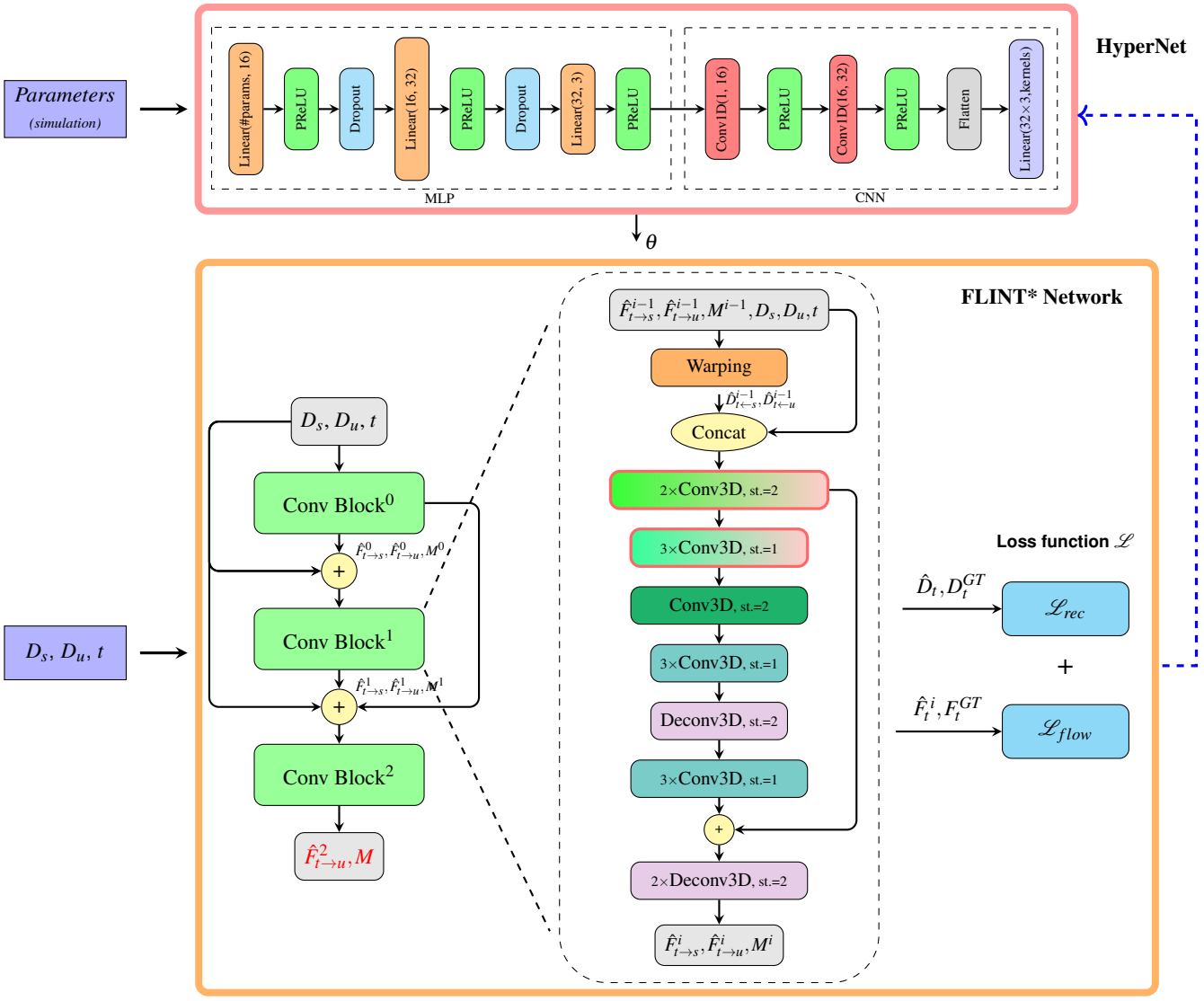


Figure 2: *HyperFLINT network architecture and pipeline during training: Given the input fields D_s and D_u, and their simulation parameters, HyperFLINT predicts the $\hat{D}_t$ scalar field and $\hat{F}^i_t$ flow fields used in the loss function for optimizing network parameters. The HyperFLINT model consists of two key components: the HyperNet and the main network, FLINT*. The HyperNet, depicted within the red box, generates weights for the convolutional layers of FLINT* (with red outlines in the middle column of the orange box). The FLINT* model architecture and loss function are shown in the orange box. The model consists of several stacked blocks of the convolutional network, which takes D_s, D_u, and t as input and in the i^{th} Conv Block computes estimated flows $\hat{F}^i_{t\to s}, \hat{F}^i_{t\to u}$, and fusion mask M^i used for interpolation. The zoomed-in view on the right highlights the structure of a generic Conv Block. The GT density D^{GT}_t and flow F^{GT}_t is only used in the loss function $\mathscr{L}$. The blue dashed arrow in the right of the figure represents the gradient propagation during training, from the output of FLINT* back to the HyperNet.*

combined effect of reverse mapping and interpolation by the warping operator $\overleftarrow{W}$. Specifically, the mappings are guided by the flow fields, producing the warped scalar fields $\hat{D}_{t\leftarrow s} = \overleftarrow{W}(D_s, \hat{F}_{t\to s})$ and $\hat{D}_{t\leftarrow u} = \overleftarrow{W}(D_u, \hat{F}_{t\to u})$, which represent the values at time t based on the source volumes D_s and D_u, respectively, see Fig. 3.

In HyperFLINT, iterative refinement within FLINT*'s convolutional blocks is applied similarly to the original FLINT method [GRF25]. However, FLINT* is optimized specifically for use with a hypernetwork (see Sec. 3.5), the architecture of which was constructed based on a thorough hyperparameter search (Sec. 6.3) to ensure efficiency and compatibility. The interpolated scalar field $\hat{D}_t$, intermediate flow fields $\hat{F}^i_t$, and estimated final flow field $\hat{F}_t$ are computed as follows:

$$\hat{D}_t = \hat{D}^{N-1}_{t\leftarrow s} \odot M + \hat{D}^{N-1}_{t\leftarrow u} \odot (\mathbf{I} - M), \tag{1a}$$

$$\hat{F}^{i+1}_t = \hat{F}^i_{t\to u}, \text{ for } i = 0, \ldots, N-2, \quad \hat{F}_t = \hat{F}^{N-1}_t \ (N = 3). \tag{1b}$$

Inference. During the inference phase, HyperFLINT processes the input scalar fields D_s and D_u, along with their associated simulation parameters using the fully trained FLINT* and HyperNet networks. Here, GT information and loss functions are absent, as

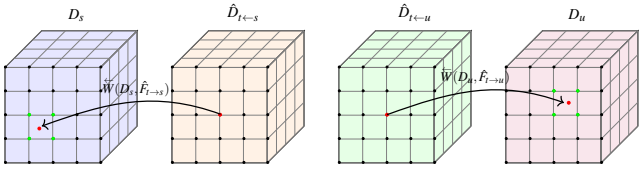


Figure 3: *Illustration of the 3D backward warping $\overleftarrow{W}$: (scalar) fields D_s and D_u are reversely mapped according to the flow fields $\hat{F}_{t\to s}$ and $\hat{F}_{t\to u}$. The fields $\hat{D}_{t\leftarrow s}$ and $\hat{D}_{t\leftarrow u}$ are then reconstructed using trilinear interpolation considering the values at the coordinates shown with green dots (for the visible front surface of the cube).*

the network operates solely with its learned parameters. The hypernetwork plays a crucial role by dynamically generating the convolutional layer weights (i.e., kernels) for the FLINT* network based on the provided simulation parameters, as illustrated in Fig. 2. This allows FLINT* to adapt its operations according to the specific characteristics of each input subset. The final outputs, namely the interpolated scalar field $\hat{D}_t$ and the reconstructed flow field $\hat{F}_t$, are computed according to Eqs. (1a) and (1b) respectively, see Fig. 1.

3.4. Loss Function

The total HyperFLINT loss is a linear combination of reconstruction loss $\mathscr{L}_{rec}$ and flow loss $\mathscr{L}_{flow}$:

$$\mathscr{L} = \mathscr{L}_{rec} + \lambda_{flow}\mathscr{L}_{flow}, \tag{2}$$

where $\lambda_{flow} = 0.2$ for balancing total loss scale w.r.t. the reconstruction component (determined experimentally, see Sec. 6.3).

Scalar field interpolation. To interpolate scalar fields temporally, we include a loss component to improve the accuracy of the interpolated density field from Eq.(1a). The reconstruction loss $\mathscr{L}_{rec}$ measures the L_1 distance between the GT density D_t^{GT} and the interpolated field $\hat{D}_t$ produced by the HyperFLINT network:

$$\mathscr{L}_{rec} = \|D_t^{GT} - \hat{D}_t\|_1. \tag{3}$$

This ensures that the interpolated scalar field closely matches the GT, improving the accuracy of the temporal interpolation.

Flow estimation. To improve the quality of the learned flow field, we incorporate a flow loss component calculated as the L_1 distance between the estimated flow at each network block and the GT flow (used only during training). Accumulating this measure across all blocks, rather than only the final one, yields better results. Additionally, we adopt exponentially increasing weights for the loss from RAFT [TD20], resulting in the following flow loss equation:

$$\mathscr{L}_{\text{flow}} = \sum_{i=1}^{N} \gamma^{N-i} \|F_t^{GT} - \hat{F}_t^i\|_1, \tag{4}$$

where F_t^{GT} is the GT flow at time t, $\hat{F}_t^i$ is the flow output from the corresponding i^{th} block of the FLINT* network (Eq. (1b)), and $N = 3$ is the number of blocks in the model. We experimentally established the value of γ as 0.8, aligning with the RAFT loss and validating this choice through hyperparameter search.

3.5. Comparison between HyperFLINT and FLINT methods

The key innovation of HyperFLINT is its integration of a hypernetwork that dynamically generates weights for the FLINT* network, a variant of the original FLINT model where the student-teacher setup has been removed, thus reducing complexity while being optimized to work with the hypernetwork. This enables HyperFLINT to condition convolutional layers on simulation parameters, aligning the network more precisely with data characteristics. The result is enhanced accuracy in flow estimation and scalar field interpolation, capturing physical phenomena with greater fidelity than the original FLINT approach. Furthermore, the inclusion of a hypernetwork introduces a novel capability to HyperFLINT: parameter space exploration. By conditioning the model on input parameters, HyperFLINT generates predictions for unseen configurations, enabling researchers to estimate outcomes for unsimulated parameter sets. This approach is particularly valuable in computationally expensive scientific studies, offering insights without requiring exhaustive simulations. This flexibility distinguishes HyperFLINT from traditional methods like FLINT, which lack parameter-driven adaptability.

4. Study Setup

In this section, we describe the training setup, provide an overview of the datasets used in our experiments and discuss the evaluation methods employed to assess the results obtained by HyperFLINT.

4.1. Training

We apply the standard prepossessing step of normalization to [0, 1] for the scalar fields and to [-1, 1] for the vector fields before the start of the training. HyperFLINT is optimized using AdamW [LH*17] with early stopping, similarly to [GRF25] to prevent overfitting on the training data. We use an experimentally determined learning rate of 10^{-4} with a cosine annealing scheduler that gradually decreases the learning rate to 10^{-5} by the end of the training. We train HyperFLINT with mini-batches of size 4 for the 3D ensemble datasets used in our study (see Sec. 4.2). We split the set of all available data into training, validation, and test subsets. To support arbitrary interpolation and flow estimation during training, $t \in [s,u]$ is chosen randomly within a maximum time window of size 12, similarly to [GRF25]. This window, confirmed through hyperparameter search, efficiently determines the maximum time gap between sampled timesteps s and u for constructing the training set, allowing for effective interpolation. Our proposed HyperFLINT model is trained for 200 epochs for 8 hours on a single Nvidia Titan V GPU with 12GB of VRAM to converge for all datasets.

4.2. Datasets and Evaluation

We consider two scientific ensemble datasets in our study.

Nyx. The first dataset is a 3D+time ensemble based on the compressible cosmological hydrodynamics simulation Nyx, developed by Lawrence Berkeley National Laboratory [SLA*21]. We consider an ensemble comprising 36 members, each consisting of a maximum of 1600 timesteps with a spatial resolution of $128 \times 128 \times 128$. It contains density and the x, y, z components of velocity. Akin to InSituNet [HWG*19], we vary three parameters for ensemble generation: the total matter density ($\Omega_m \in [0.1, 0.2]$), the total density of baryons ($\Omega_b \in [0.0215, 0.0235]$), and the Hubble constant ($h \in [0.55, 0.75]$). We randomly sample a training subset of 500 timesteps and utilize different ensemble members for training, validation, and testing.

Castro. The second dataset consists of a 3D+time ensemble

based on the astrophysical hydrodynamics simulation Castro, simulating the merger of two white dwarfs, developed by the Lawrence Berkeley National Laboratory [ABB*10]. This ensemble contains 12 members, each with up to 800 timesteps, and a spatial resolution of $128 \times 128 \times 128$. The dataset includes the density field and the x, y, and z velocity components. For ensemble generation, we vary two parameters based on the authors' suggestions, such as the masses of the primary (M_P) and secondary (M_S) white dwarfs ($M_P, M_S \in [0.8, 0.95] M_\odot$), where $M_\odot$ represents the solar mass. A training subset of 400 timesteps is randomly sampled, with different ensemble members designated for training, validation, and testing.

We evaluate HyperFLINT's performance in reconstructing scalar and vector fields both qualitatively and quantitatively. For density field evaluation, we use *peak signal-to-noise ratio* (PSNR), while the accuracy of the flow field is assessed with *endpoint error* (EPE), which calculates the average Euclidean distance between estimated and GT flow vectors—lower EPE values indicating higher accuracy. For qualitative assessment, we visualize flow field outcomes for the simulation ensembles. Given the 3D nature of our datasets, PSNR and EPE are calculated in the volume domain, ensuring the metrics align with the spatial characteristics of the data and provide a robust evaluation of HyperFLINT's performance.

5. Qualitative Results

We evaluate HyperFLINT on the two datasets (Sec. 4.2) with respect to flow estimation and density interpolation (i.e., temporal super-resolution), and compare it with FLINT and STSR-INR.

5.1. Nyx

First, we consider the scenario where GT density and velocity fields are available for some members of the entire 3D ensemble. As Fig. 4a illustrates, HyperFLINT achieves accurate performance in terms of both density field interpolation and flow field estimation. Visually, the difference between the renderings of the reconstructed density field (second row) and its GT is minimal. Moreover, even at a relatively high interpolation rate of $5\times$, HyperFLINT effectively learns a flow field that structurally resembles the GT flow. Both HyperFLINT and FLINT (third row) produce results that are visually very close to the GT for density interpolation; however, when compared to STSR-INR (fourth row), HyperFLINT demonstrates more accurate density field interpolation. At $t = 303$, $t = 403$, and $t = 503$, the purple and orange zoom-ins clearly reveal a different structure and less dark matter density in STSR-INR's results compared to HyperFLINT, which reconstructs density more effectively.

Moreover, HyperFLINT not only reconstructs the density field but also supplements it with accurate flow information, as shown in Fig. 4a (sixth row), a feature that STSR-INR lacks. When examining the flow, we observe circular swirling patterns, indicating the complex dynamics of the baryonic gas. As these flows intensify, we see evidence of dark matter moving outward—reflected in both the GT and HyperFLINT density, especially in the orange zoom-ins—consistent with an expanding universe. This underlines the utility of HyperFLINT in capturing not just the static density fields but also the dynamic evolution of the cosmic structures. Furthermore, HyperFLINT outperforms the FLINT model (last row) in capturing flow information, producing more accurate representations of the underlying dynamics. For example, at $t = 303$ and $t = 503$, in the purple and orange zoom-ins, the FLINT flow exhibits structural differences compared to the GT, whereas HyperFLINT more effectively preserves the intricate flow dynamics.

A domain expert from astronomy specializing in cosmological simulations and observational data highlighted the advantages of estimating both density and velocity fields. This capability is crucial in cosmology for estimating distances between astronomical objects and analyzing their spatial relationships. This would help in bridging simulations with real-world observations. Velocity estimation is particularly valuable for constructing "lightcones", in which simulation data are used to model the evolution of the universe.

5.2. Castro

For the Castro ensemble dataset, as Fig. 4b illustrates, HyperFLINT achieves results that are visually very close to the GT for both density field interpolation and flow field estimation. When comparing HyperFLINT to FLINT and STSR-INR for temporal density interpolation (third and fourth row), HyperFLINT demonstrates superior accuracy. For instance, at $t = 103$ and $t = 203$, STSR-INR and FLINT show noticeable deviations in structure and reduced matter density around the merging white dwarfs. In contrast, HyperFLINT preserves these density structures more effectively, aligning well with the GT. For flow estimation, HyperFLINT produces more coherent and spatially accurate flow fields than FLINT. This is particularly evident at $t = 3$ and $t = 203$ in Fig. 4b, where FLINT introduces artifacts and inconsistencies in motion direction. In contrast, HyperFLINT better captures the developing flow dynamics, especially in the detailed areas in the purple and orange zoom-ins, where it maintains a more accurate representation of the evolving flows.

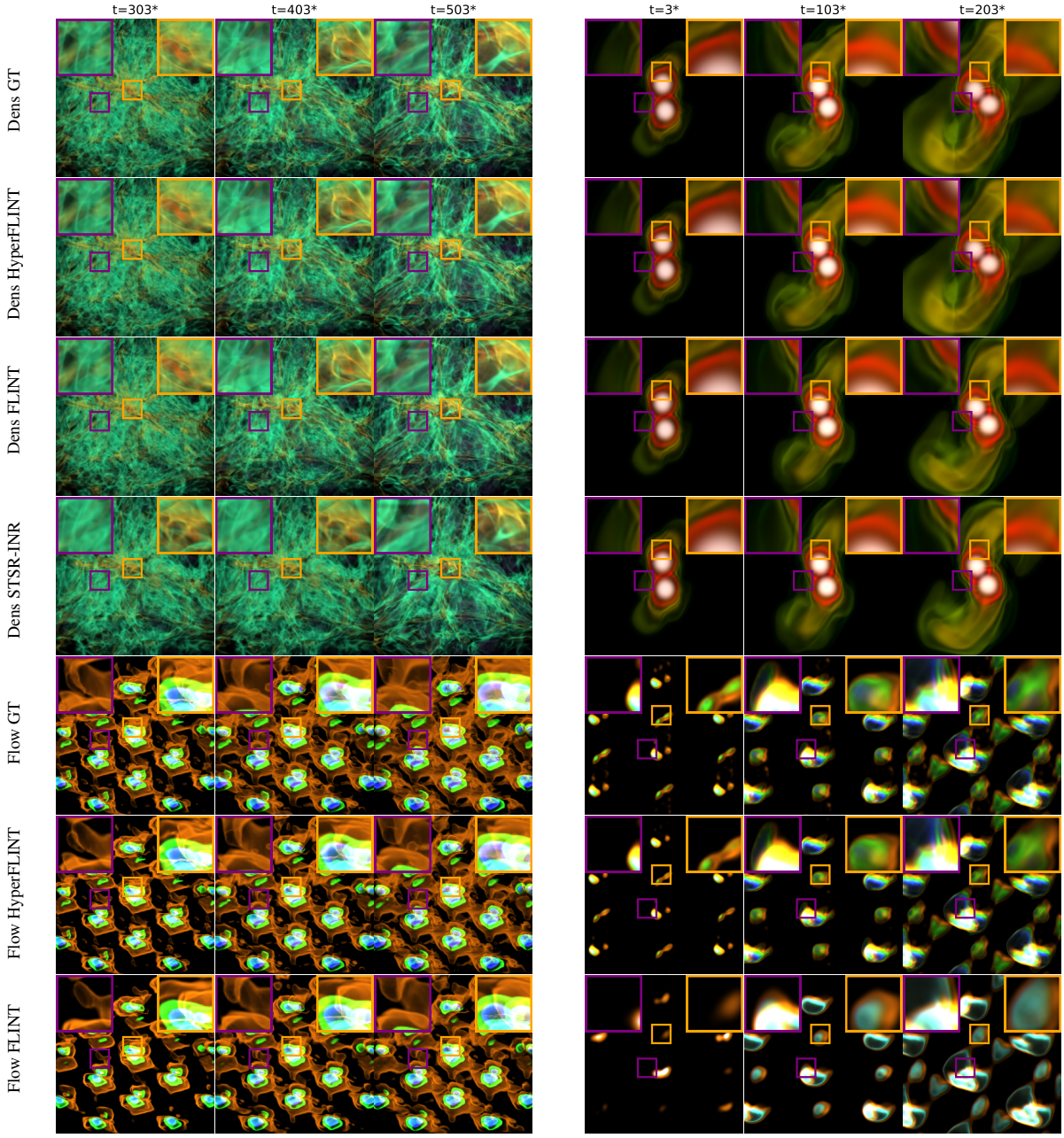
6. Quantitative and Comparative Evaluation

In this section, we present quantitative results and compare against baseline methods to demonstrate the improvement achieved with our proposed method, followed by ablation and parameter studies to explore different configurations and hyperparameters.

6.1. Comparison Against Baselines

STSR-INR [TW24] and CoordNet [HW22] were evaluated as key benchmarks for temporal super-resolution (TSR) tasks on the 3D+time Nyx and Castro datasets, with results shown in Table 1 and Table 2. STSR-INR employs a variational auto-decoder to optimize latent vectors for variable interpolation in the latent space, building on prior works like STNet [HZCW21]. It has demonstrated notable improvements in density interpolation but focuses solely on scalar field interpolation, lacking capabilities for flow field estimation. Similarly, CoordNet, a benchmark for visualizing time-varying volumetric data, also improves upon TSR-TVD [HW19] as detailed in Sec. 2. While CoordNet achieves strong results in PSNR scores for density interpolation across various rates, it shares STSR-INR's limitation in addressing only TSR of scalar fields without extending to flow field estimation. In contrast, HyperFLINT not only outperforms both methods in density interpolation but also introduces flow estimation, a capability absent in both CoordNet and STSR-INR.

FLINT [GRF25], based on a pure CNN architecture with a



(a) *Nyx results* **(b)** *Castro results*

Figure 4: *Nyx and Castro: HyperFLINT flow field estimation and temporal density interpolation, 5×. From top to bottom, the rows show GT density, HyperFLINT interpolated density, FLINT interpolation, STSR-INR interpolation, GT flow, HyperFLINT flow estimation, and FLINT flow estimation. 3D rendering was used for the density and flow visualization (● ● ● colors representing x, y, and z flow directions respectively).*

student-teacher training mechanism, serves as another valuable benchmark for comparison. FLINT demonstrated improved results over STSR-INR and CoordNet in temporal interpolation and provided reasonable flow estimation, making it a strong competitor. Our evaluation against FLINT shows that HyperFLINT slightly outperforms FLINT in terms of TSR for scalar fields. More notably, HyperFLINT demonstrates superior performance in flow estimation, yielding significantly better results. This is evidenced by the lower EPE scores reported for both the Nyx and Castro datasets, as shown in Table 1 and Table 2. Furthermore, HyperFLINT offers the added advantage of enabling parameter space exploration, a feature neither FLINT nor other baselines can achieve. This makes HyperFLINT a more versatile and powerful tool for analyzing scientific ensembles.

We conducted an inference time comparison of HyperFLINT against FLINT, CoordNet, and STSR-INR across the two 3D ensemble datasets, measuring inference time over 300 timesteps from the test set. HyperFLINT significantly outperforms CoordNet and STSR-INR in speed, achieving an average of 0.18 seconds per

Table 1: *Comparison against baselines, Nyx*

Method	3×		5×		8×	
	PSNR ↑	EPE ↓	PSNR ↑	EPE ↓	PSNR ↑	EPE ↓
HyperFLINT	**53.32**	**0.0237**	**52.70**	**0.0238**	**51.07**	**0.0242**
FLINT	53.17	0.0310	52.31	0.0310	49.39	0.0309
STSR-INR	49.63	—	44.21	—	41.09	—
CoordNet	49.37	—	44.02	—	40.78	—
Linear	47.49	—	41.92	—	37.51	—

Table 2: *Comparison against baselines, Castro*

Method	3×		5×		8×	
	PSNR ↑	EPE ↓	PSNR ↑	EPE ↓	PSNR ↑	EPE ↓
HyperFLINT	**49.48**	**0.0275**	**47.39**	**0.0276**	**45.41**	**0.0278**
FLINT	47.89	0.0502	46.16	0.0506	43.83	0.0513
STSR-INR	44.83	—	42.38	—	40.26	—
CoordNet	44.52	—	42.29	—	40.08	—
Linear	42.11	—	37.28	—	33.64	—

timestep compared to 2.1 seconds for CoordNet and 1.5 seconds for STSR-INR. Remarkably, HyperFLINT even slightly outpaces FLINT, which averages 0.2 seconds per timestep. This improvement arises from HyperFLINT's efficient CNN utilization, which, despite the added hypernetwork complexity, maintains performance advantage. Unlike INR-based methods that solve implicit functions at each point, HyperFLINT leverages CNNs' parallel processing capabilities and optimized memory access, resulting in faster inference.

6.2. Ablation Studies

Our proposed HyperFLINT method is a deep neural network that has various loss components. To assess the impact of these in achieving optimal results, we conducted a series of ablation studies.

Table 3: *Ablation of HyperFLINT*

Method	Nyx, 5×		Castro, 5×	
	PSNR ↑	EPE ↓	PSNR ↑	EPE ↓
HyperFLINT no *flow*	51.94	0.1348	46.92	0.7432
HyperFLINT no *rec*	44.78	0.0254	37.89	0.0335
HyperFLINT w/o hyper	50.89	0.0357	46.04	0.0516
HyperFLINT	**52.70**	**0.0238**	**47.39**	**0.0276**

The ablation studies on HyperFLINT reveal the impact of various loss components on performance, as shown in Table 3 for the Nyx and Castro datasets. Four variants are compared:

- HyperFLINT no *flow*: omits the flow loss, leading to poor flow estimation (EPE) despite decent density interpolation (PSNR).
- HyperFLINT no *rec*: omits the reconstruction loss, resulting in decent flow learning but compromised density interpolation.
- HyperFLINT w/o hyper: removes the hypernetwork, leading to subpar interpolation and flow estimation due to the absence of dynamic adaptation enabled by simulation parameters.
- HyperFLINT: the full model with all components, achieves the highest scores in both PSNR and EPE.

6.3. Hyperparameter Studies

Our proposed HyperFLINT method involves several hyperparameters, and we conducted extensive hyperparameter optimization to identify the optimal configuration for HyperFLINT, testing variations in model width, depth, and loss functions. Table 4 displays the outcomes of these studies regarding spatio-temporal datasets.

Table 4: *Hyperparameter search for HyperFLINT*

Method	Nyx, 5×		Castro, 5×	
	PSNR ↑	EPE ↓	PSNR ↑	EPE ↓
HyperFLINT *64*	51.68	0.0315	47.12	0.0430
HyperFLINT *Lapl*	51.73	0.0319	44.95	0.0480
HyperFLINT *hyper all*	48.19	0.0281	42.79	0.0419
HyperFLINT *w teacher*	49.31	0.0265	44.43	0.0393
HyperFLINT $\lambda_{flow} = 0.3$	52.29	0.0311	45.39	0.0316
HyperFLINT $\lambda_{flow} = 0.1$	52.19	0.0320	47.01	0.0325
HyperFLINT $\gamma = 0.9$	52.14	0.0310	46.98	0.0345
HyperFLINT $\gamma = 0.7$	52.28	0.0317	47.07	0.0371
HyperFLINT *stride = 1*	52.19	0.0418	46.94	0.0348
HyperFLINT *2 Blocks*	49.72	0.0245	46.87	0.0407
HyperFLINT *3 Blocks*	**52.70**	**0.0238**	**47.39**	**0.0276**
HyperFLINT *4 Blocks*	52.51	0.0249	46.98	0.0314
HyperFLINT *5 Blocks*	52.09	0.0256	46.97	0.0341
HyperFLINT *hyper no MLP*	52.27	0.0261	46.06	0.0363
HyperFLINT *hyper no CNN*	51.74	0.0282	45.46	0.0378
HyperFLINT *hyper no dropout*	52.48	0.0259	46.61	0.0298

The best-performing setup, labeled "HyperFLINT *3 Blocks*", includes three blocks with convolutional layers having channel counts decreasing from 128 to 64, balancing capacity and avoiding overfitting. Alternatives like "HyperFLINT *hyper all*" (all layers affected by the hypernetwork) and "HyperFLINT *w teacher*" (using a teacher block similar to FLINT) resulted in convergence issues and degraded performance. Similarly, the "HyperFLINT *stride = 1*" variant underperformed due to reduced expressiveness from fixed strides. We explored loss functions, comparing "HyperFLINT *Lapl*" using a Laplacian pyramid, and "HyperFLINT *3 Blocks*" which applies L_1 loss. The simpler L_1 loss proved more effective. Additionally, we show the roles of HyperNet components, as configurations like "HyperFLINT *hyper no MLP*," "HyperFLINT *hyper no CNN*," and "HyperFLINT *hyper no dropout*" exhibited reduced performance. These findings underscore the importance of the proposed architecture and individual components. In Table 5, we present the hyperparameter search results for Nyx at 64^3 resolution, where "HyperFLINT *3 Blocks*" remains the best-performing model, consistent with what has been determined in Table 4 at 128^3. This suggests good generalization across resolutions, with a slight performance drop at higher resolutions due to increased data complexity.

Table 5: *Hyperparameter search for HyperFLINT, Nyx, 5×, 64^3*

Method	PSNR ↑	EPE ↓
HyperFLINT *hyper all*	48.87	0.0279
HyperFLINT *3 Blocks*	**52.93**	**0.0220**
HyperFLINT *4 Blocks*	52.81	0.0231
HyperFLINT *hyper no MLP*	52.33	0.0254
HyperFLINT *hyper no CNN*	51.65	0.0277

7. Simulation Parameter Space Exploration

This section highlights two key examples demonstrating HyperFLINT's utility in parameter space exploration. First, Sec. 7.1 examines the correlation between HyperNet-generated weights and simulation data, showcasing a strong alignment with simulation characteristics. Second, Sec. 7.2 demonstrates HyperFLINT's ability to interpolate within the parameter space, accurately predicting outputs for configurations beyond the training set. These functionalities are validated using experiments with the Nyx simulation ensemble.

7.1. Hypernetwork weights as Proxy for Data Similarity

To better understand the broader capabilities of HyperNet beyond generating weights for the FLINT* network, we explored its potential for parameter space analysis. Specifically, we constructed

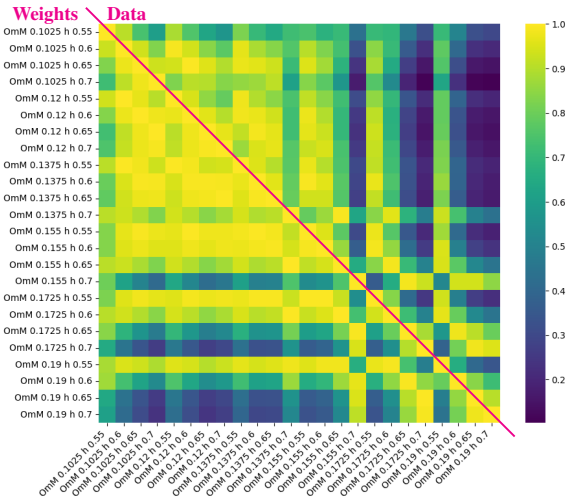


Figure 5: *Similarity matrix: lower-left—HyperNet's weight similarity, upper-right—Nyx simulation parameter similarity.*

similarity matrices to examine how the weights generated by HyperNet correlate with the underlying data dynamics, reflecting the influence of simulation parameters. This analysis highlights how HyperNet weights, apart from serving as weight generation for the main network, can provide insights into parameter-driven variations within the dataset. Specifically, we compute similarity matrices for (i) the weights produced by HyperNet and (ii) the original dataset volumes (both in Fig. 5), both of which are naturally influenced by the simulation parameters. For this analysis, we considered 24 members from the Nyx ensemble under variation of $\Omega_m = [0.1025, 0.12, 0.1375, 0.155, 0.1725, 0.19]$, $h = [0.55, 0.6, 0.65, 0.7]$, while Ω_b was fixed at 0.0225. Our proposed HyperFLINT model was trained on 25% of the data, specifically on six members with the combinations of parameters $\Omega_m = [0.1025, 0.1375, 0.19]$, $h = [0.55, 0.7]$, with $\Omega_b = 0.0225$.

The similarity matrices reveal a strong correlation between the HyperNet-generated weights and the dataset volumes, confirming that the HyperNet effectively learns parameter-dependent representations. This correlation suggests that differences in HyperNet weights can serve as a meaningful surrogate for differences in the underlying data. Upon closer examination, certain members stand out with distinguishable characteristics. For instance, for parameter configurations such as $\Omega_m = 0.155$ and $h = 0.7$, and $\Omega_m = 0.1725$ with $h = 0.65$ or $h = 0.7$, the data shows lower similarity compared to other ensemble members. Similarly, for $\Omega_m = 0.19$ with $h = 0.6$, $h = 0.65$, and $h = 0.7$, the matrix highlights notable deviations in the data structure. Despite these variations, HyperNet's weight similarities consistently maintain strong correlations across all configurations, enabling insights into data dynamics without requiring generation and access to the original data fields. We also conducted a quantitative evaluation using a *triplet loss* approach [SJ03] to validate the alignment between HyperNet-generated weights and parameter-dependent data dynamics. We analyzed triplets comprising an anchor, a more similar member, and a less similar member to evaluate how well distances in the HyperNet-generated weight space align with those in the data space. Specifically, we computed the distance between the anchor and both the more similar and less similar members in the data space and performed the same calculation in the HyperNet-generated weight space, confirming that weight similarity strongly correlates with data similarity. Achieving a 96% triplet correlation, this result underscores HyperFLINT's utility in parameter space analysis, where learned representations guide tasks like constructing characteristic maps of parameter impacts and optimizing sampling strategies.

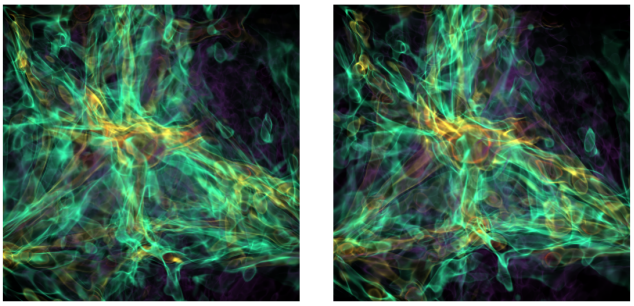


Figure 6: *The rendering of two volumes that were used for Nyx simulation parameter space exploration.*

7.2. Parameter-Driven Data Synthesis

Leveraging the HyperNet architecture, which conditions the FLINT* network on specific simulation parameters, HyperFLINT can dynamically adapt its outputs based on changes in simulation settings. As shown in Fig. 6, which presents renderings of the input volumes at timesteps s and u, these volumes serve as the foundation for interpolating the target timestep while varying only the simulation parameters through HyperNet. As illustrated in Fig. 7, this approach effectively generates outputs that closely match the ensemble volumes associated with the specified parameter values. Even when starting with identical initial data, HyperFLINT effectively adapts its predictions solely based on the altered parameter inputs. For instance, the density reconstruction and flow estimation are particularly accurate for the parameter setting $\Omega_m = 0.1375$ (third column), corresponding to the data from the ensemble member used as inputs. Notably, HyperFLINT maintains the structural integrity and fine-grained details of the density field (second row) while accurately capturing the dynamic flow patterns (fifth row), demonstrating its capability to preserve both shape and texture. By using the same input volume and altering only the simulation parameters, HyperFLINT generates new data outputs effectively for settings like $\Omega_m = 0.12$, 0.155, and 0.1725, achieving visually close reconstructions, as the difference plots confirm, showcasing its generalization capabilities. While novel (unseen) simulation features cannot be generated, these results suggest that the model is capable of capturing parameter-dependent relationships, as reflected in the smooth density transitions and flow field predictions that generally follow GT dynamic trends. The reconstruction remains robust despite these novel parameters. As expected, we note a slight increase in error for more distant parameter values such as $\Omega_m = 0.1025$ and 0.19 (first and last columns, respectively). This is likely due to these configurations being further away from the range of parameters the model received as input during inference. Nonetheless, the results remain decent, with distinct features preserved, such as central galaxy density formation and distinct flow patterns, showcasing HyperFLINT's parameter-dependent generalization capabilities.

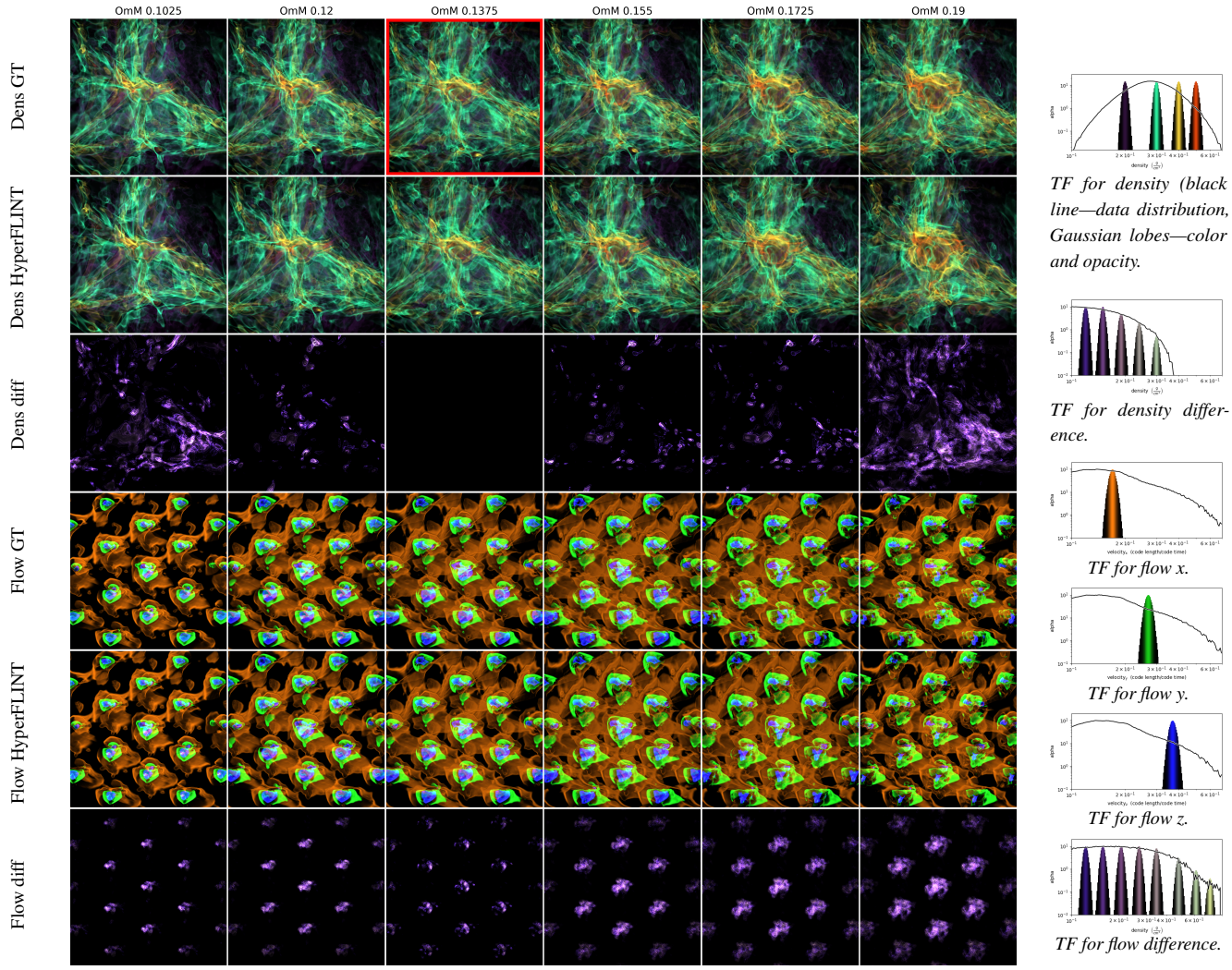


Figure 7: *Nyx simulation parameter space exploration and transfer functions for density and flow field components. From top to bottom, the rows show GT density, HyperFLINT interpolated density, difference between GT and HyperFLINT density, GT flow, HyperFLINT flow estimation, and difference between GT and HyperFLINT flow.*

8. Conclusion and Future Work

In this work, we have proposed HyperFLINT, a hypernetwork-based method for flow estimation and scalar field interpolation in spatio-temporal scientific ensembles. By leveraging hypernetworks, HyperFLINT adapts to varying simulation parameters, enabling accurate flow estimation and high-quality temporal interpolants, even with limited data. It outperforms recent state-of-the-art methods while achieving fast inference without requiring extensive pre-training or fine-tuning on simplified datasets. Validated across diverse simulation ensembles, HyperFLINT demonstrates robust and versatile performance, making it a valuable tool for scientific visualization.

The integration of hypernetworks in HyperFLINT establishes a robust framework for analyzing how simulation parameters influence flow and density dynamics, enabling efficient parameter space exploration and deeper insights into complex systems. Integrating skill score metrics that account for spatial and temporal biases, along with adaptivity based on flow characteristics and simulation constraints, could enhance robustness and applicability in future work. Incorporating stable diffusion techniques [RBL*22] could further enhance HyperFLINT's ability to generate high-quality vector and scalar fields by leveraging simulation parameters. Known for producing smooth outputs, stable diffusion methods could complement hypernetworks, improving generalization in intricate simulations.

Acknowledgments

We thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Hábrók high performance computing cluster. We also thank the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) for supporting this work by funding SFB 1313 (Project Number 327154368). We thank dr. Maxime Trebitsch (Institut d'Astrophysique de Paris) for his help in interpreting the flow visualizations of the Nyx cosmological simulation and assessing their usefulness for data analysis.

References

[ABB*10] ALMGREN A. S., BECKNER V. E., BELL J. B., DAY M. S., HOWELL L. H., JOGGERST C. C., LIJEWSKI M. J., NONAKA A., SINGER M., ZINGALE M.: CASTRO: A New Compressible Astrophysical Solver. I. Hydrodynamics and Self-gravity. *The Astrophysical Journal 715*, 2 (June 2010), 1221–1238. arXiv:1005.0114, doi:10.1088/0004-637X/715/2/1221. 6

[BLRW17] BROCK A., LIM T., RITCHIE J. M., WESTON N.: SMASH: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344* (2017). 2

[CAA*20] CHILDS H., AHERN S. D., AHRENS J., BAUER A. C., BENNETT J., BETHEL E. W., BREMER P.-T., BRUGGER E., COTTAM J., DORIER M., ET AL.: A terminology for in situ visualization and analysis systems. *The International Journal of High Performance Computing Applications 34*, 6 (2020), 676–691. 2

[CBGH19] CHILDS H., BENNETT J., GARTH C., HENTSCHEL B.: In situ visualization for computational science. *IEEE Comput. Graph. Appl. 39*, 6 (2019), 76–85. 1

[DLHT15] DONG C., LOY C. C., HE K., TANG X.: Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell. 38*, 2 (2015), 295–307. 2

[FE17] FREY S., ERTL T.: Flow-based temporal selection for interactive volume visualization. *Comput. Graph. Forum 36*, 8 (2017), 153–165. doi:https://doi.org/10.1111/cgf.13070. 1

[GRF25] GADIROV H., ROERDINK J. B. T. M., FREY S.: FLINT: Learning-based flow estimation and temporal interpolation for scientific ensemble visualization. *IEEE Transactions on Visualization and Computer Graphics* (2025). 1, 2, 3, 4, 5, 6

[HDL16] HA D., DAI A., LE Q. V.: Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016). 2

[HW19] HAN J., WANG C.: TSR-TVD: Temporal super-resolution for time-varying data analysis and visualization. *IEEE Trans. Vis. Comput. Graph. 26*, 1 (2019), 205–215. 2, 6

[HW20] HAN J., WANG C.: SSR-TVD: Spatial super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics 28*, 6 (2020), 2445–2456. 3

[HW22] HAN J., WANG C.: CoordNet: Data generation and visualization generation for time-varying volumes via a coordinate-based neural network. *IEEE Trans. Vis. Comput. Graph.* (2022). 1, 3, 6

[HWG*19] HE W., WANG J., GUO H., WANG K.-C., SHEN H.-W., RAJ M., NASHED Y. S., PETERKA T.: InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Trans. Vis. Comput. Graph. 26*, 1 (2019), 23–33. 5

[HZCW21] HAN J., ZHENG H., CHEN D. Z., WANG C.: STNet: An end-to-end generative framework for synthesizing spatiotemporal super-resolution volumes. *IEEE Trans. Vis. Comput. Graph. 28*, 1 (2021), 270–280. 2, 6

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)* (2015), pp. 1026–1034. 3

[JBY23] JIAO C., BI C., YANG L.: FFEINR: Flow feature-enhanced implicit neural representation for spatio-temporal super-resolution. *arXiv preprint arXiv:2308.12508* (2023). 3

[JSJ*18] JIANG H., SUN D., JAMPANI V., YANG M.-H., LEARNED-MILLER E., KAUTZ J.: Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)* (2018), pp. 9000–9008. 2

[JWC*11] JÄNICKE H., WEIDNER T., CHUNG D., LARAMEE R. S., TOWNSEND P., CHEN B. M.: Visual reconstructability as a quality metric for flow visualization. *Comput. Graph. Forum 30*, 3 (2011), 781–790. URL: http://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2011.01927.x, doi:https://doi.org/10.1111/j.1467-8659.2011.01927.x. 1

[KRRW18] KUMPF A., RAUTENHAUS M., RIEMER M., WESTERMANN R.: Visual analysis of the temporal evolution of ensemble forecast sensitivities. *IEEE Trans. Vis. Comput. Graph. 25*, 1 (2018), 98–108. 2

[KSG*15] KAPPE C. P., SCHÜTZ L., GUNTHER S., HUFNAGEL L., LEMKE S., LEITTE H.: Reconstruction and visualization of coordinated 3D cell migration based on optical flow. *IEEE Trans. Vis. Comput. Graph. 22*, 1 (2015), 995–1004. 2

[LCS*21] LIANG J., CAO J., SUN G., ZHANG K., VAN GOOL L., TIMOFTE R.: SwinIR: Image restoration using swin transformer. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)* (2021), pp. 1833–1844. 2

[LH*17] LOSHCHILOV I., HUTTER F., ET AL.: Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101 5* (2017). 5

[LJLB21] LU Y., JIANG K., LEVINE J. A., BERGER M.: Compressive neural representations of volumetric scalar fields. In *Computer Graphics Forum* (2021), vol. 40, Wiley Online Library, pp. 135–146. 3

[LTH*17] LEDIG C., THEIS L., HUSZÁR F., CABALLERO J., CUNNINGHAM A., ACOSTA A., AITKEN A., TEJANI A., TOTZ J., WANG Z., ET AL.: Photo-realistic single image super-resolution using a generative adversarial network. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)* (2017), pp. 4681–4690. 2

[LYL*19] LI Z., YANG J., LIU Z., YANG X., JEON G., WU W.: Feedback network for image super-resolution. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)* (2019), pp. 3867–3876. 2

[MBW*18] MANANDHAR S., BOUTHEMY P., WELF E., ROUDOT P., KERVRANN C.: A sparse-to-dense method for 3D optical flow estimation in 3D light-microscopy image sequences. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI)* (2018), IEEE, pp. 952–956. 2

[MHBB22] MISHRA A., HAZARIKA S., BISWAS A., BRYAN C.: Filling the Void: Deep learning-based reconstruction of sampled spatiotemporal scientific simulation data. *arXiv preprint arXiv:2205.12868* (2022). 3

[MST*21] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM 65*, 1 (2021), 99–106. 3

[MWZ*15] MEYER S., WANG O., ZIMMER H., GROSSE M., SORKINE-HORNUNG A.: Phase-based frame interpolation for video. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)* (2015), pp. 1410–1418. 2

[NML17] NIKLAUS S., MAI L., LIU F.: Video frame interpolation via adaptive separable convolution. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)* (2017), pp. 261–270. 2

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 165–174. 3

[PPT*22] PRZEWIEŹLIKOWSKI M., PRZYBYSZ P., TABOR J., ZIEBA M., SPUREK P.: HyperMAML: few-shot adaptation of deep models with hypernetworks. *arXiv preprint arXiv:2205.15745* (2022). 2

[RBL*22] ROMBACH R., BLATTMANN A., LORENZ D., ESSER P., OMMER B.: High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 10684–10695. 10

[SB22] SAHOO S., BERGER M.: Integration-free learning of flow maps. *arXiv preprint arXiv:2211.03192* (2022). 2

[SCH*16] SHI W., CABALLERO J., HUSZÁR F., TOTZ J., AITKEN A. P., BISHOP R., RUECKERT D., WANG Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)* (2016), pp. 1874–1883. 2

[SIE21] SKOROKHODOV I., IGNATYEV S., ELHOSEINY M.: Adversarial generation of continuous images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 10753–10764. 2

[SJ03] SCHULTZ M., JOACHIMS T.: Learning a distance metric from relative comparisons. *Advances in neural information processing systems 16* (2003). 9

[SLA*21] SEXTON J., LUKIC Z., ALMGREN A., DALEY C., FRIESEN B., MYERS A., ZHANG W.: Nyx: A massively parallel AMR code for computational cosmology. *The Journal of Open Source Software 6*, 63 (2021), 3068. 5

[SZW19] SITZMANN V., ZOLLHÖFER M., WETZSTEIN G.: Scene representation networks: Continuous 3D-structure-aware neural scene representations. *Advances in Neural Information Processing Systems 32* (2019). 3

[TD20] TEED Z., DENG J.: RAFT: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Proceedings, Part II 16* (2020), Springer, pp. 402–419. 5

[TFE07] TKACHEV G., FREY S., ERTL T.: Local prediction models for spatiotemporal volume visualization. *IEEE Trans. Vis. Comput. Graph. 27*, 7 (2021-07), 3091–3108. doi:10.1109/TVCG.2019.2961893. 1

[TW24] TANG K., WANG C.: STSR-INR: Spatiotemporal super-resolution for multivariate time-varying volumetric data via implicit neural representation. *Computers & Graphics 119* (2024), 103874. 1, 3, 6

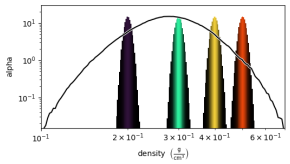
[WBCM23] WU Q., BAUER D., CHEN Y., MA K.-L.: HyperINR: A fast and predictive hypernetwork for implicit neural representations via knowledge distillation. *arXiv preprint arXiv:2304.04188* (2023). 1, 3

[WHW22] WEISS S., HERMÜLLER P., WESTERMANN R.: Fast neural representations for direct volume rendering. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 196–211. 3

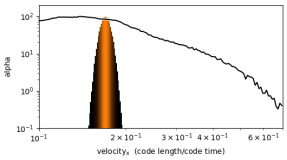
Appendix A: Supplementary Material

Nyx and Castro Transfer Functions

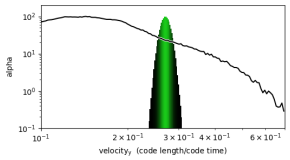
In Fig. 8 and Fig. 9 we present transfer functions for visualizing the density and velocity components of the Nyx and Castro ensembles, respectively. These are applied to the density field and the x, y, and z components of the flow field, offering an intuitive means to explore and analyze the spatial and dynamic characteristics of the simulation data. The utilized transfer functions help to better understand the structural patterns and flow dynamics inherent to each ensemble.



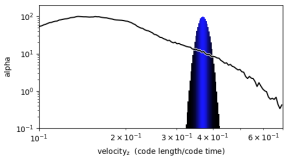
(a) *TF for density field.*



(b) *TF for flow field in x direction.*

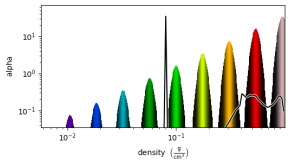


(c) *TF for flow field in y direction.*

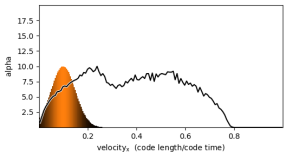


(d) *TF for flow field in z direction.*

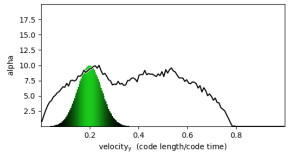
Figure 8: *Nyx ensemble: transfer function for density and x, y, z components of the flow field.*



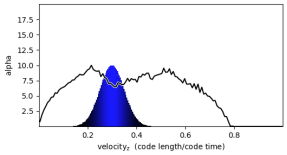
(a) *TF for density field.*



(b) *TF for flow field in x direction.*



(c) *TF for flow field in y direction.*



(d) *TF for flow field in z direction.*

Figure 9: *Castro ensemble: transfer function for density and x, y, z components of the flow field.*

3D Flow Estimation and Density Interpolation Results

This subsection presents the complete results for 3D flow estimation and temporal interpolation tasks on the Nyx (Fig. 10) and Castro (Fig. 11) datasets. The figures include visualizations of the estimated flow fields and interpolated density fields. Additionally, the results highlight comparisons with baseline methods, including FLINT and STSR-INR.

The 3D PSNR metric used in our evaluation is computed after normalizing the scalar data to [0,1], using the following formulation:

$$\mathrm{MSE} = \frac{1}{mno}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{k=0}^{o-1}(D^{GT}_{t,i,j,k} - \hat{D}_{t,i,j,k})^2, \tag{5}$$

$$\mathrm{PSNR} = 20\log_{10}(1.0) - 10\log_{10}(\mathrm{MSE}), \tag{6}$$

where the mean squared error (MSE) is calculated over the entire 3D volume, with m,n,o representing the spatial dimensions of the volume along the x-, y-, and z-axes, respectively.

For flow estimation, we compute the 3D endpoint error (EPE) as:

$$\mathrm{EPE} = \frac{1}{mno}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{k=0}^{o-1}\left\|F^{GT}_{t,i,j,k} - \hat{F}_{t,i,j,k}\right\|_2. \tag{7}$$

Here, F^{GT} and $\hat{F}$ represent the ground-truth and predicted flow fields, respectively, with components along the x-, y-, and z-axes. The EPE measures the Euclidean distance between the predicted and true flow vectors across the entire 3D volume.

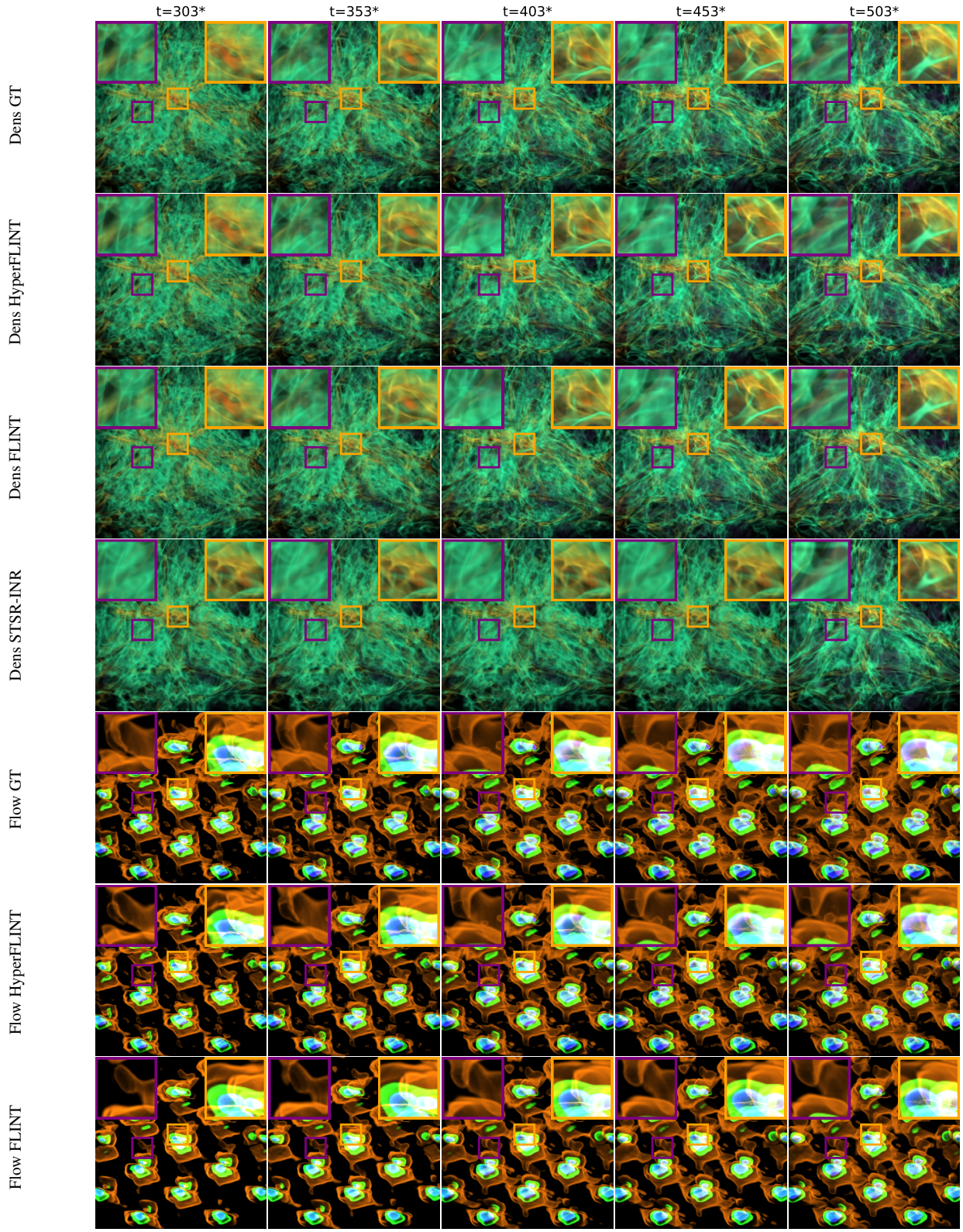


Figure 10: *Nyx: HyperFLINT flow field estimation and temporal density interpolation, 5×. From top to bottom, the rows show GT density, HyperFLINT interpolated density, FLINT interpolation, STSR-INR interpolation, GT flow, HyperFLINT flow estimation, and FLINT flow estimation. 3D rendering was used for the density and flow visualization (● ● ● colors representing x, y, and z flow directions respectively).*

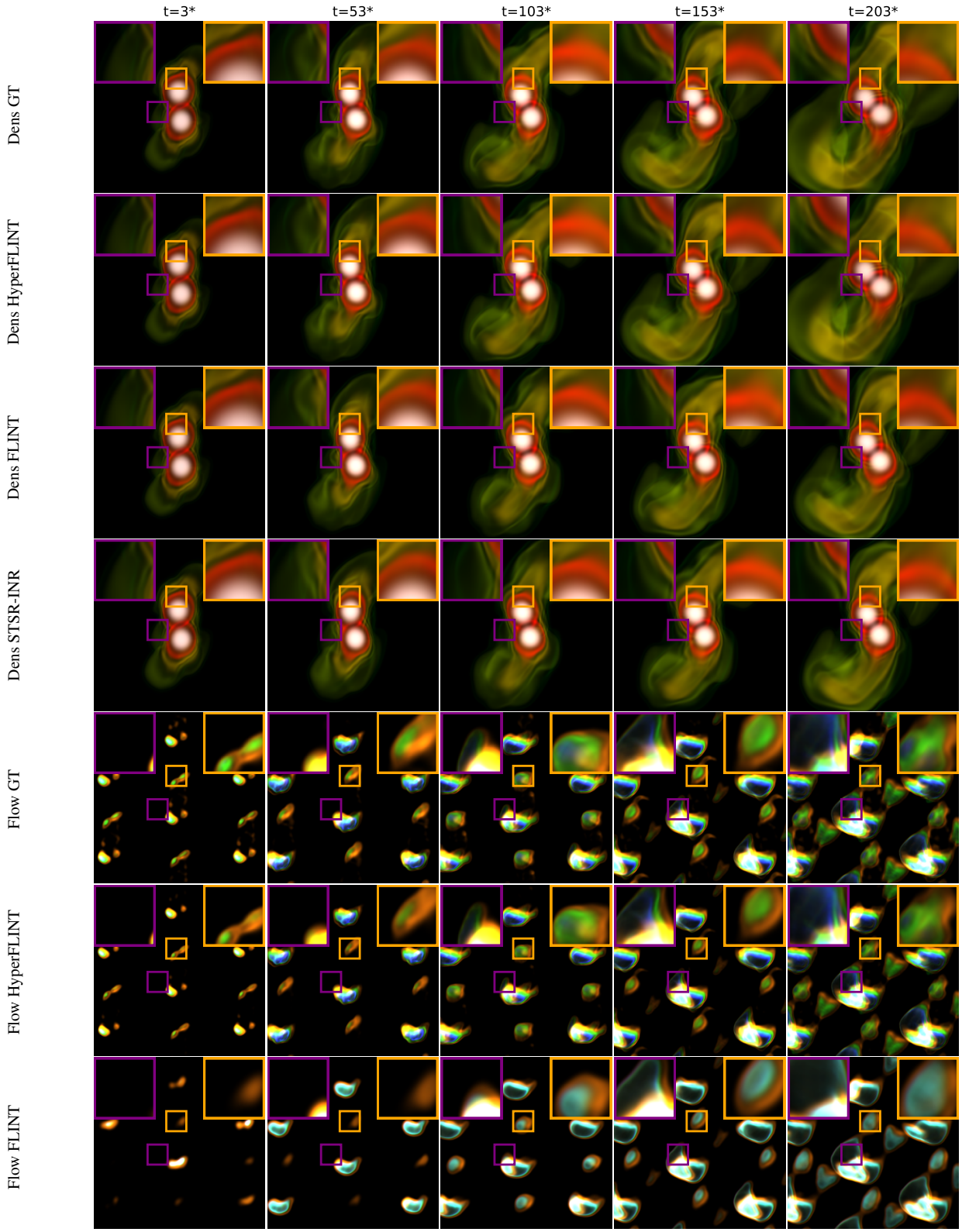


Figure 11: *Castro: HyperFLINT flow field estimation and temporal density interpolation, 5×. From top to bottom, the rows show GT density, HyperFLINT interpolated density, FLINT interpolation, STSR-INR interpolation, GT flow, HyperFLINT flow estimation, and FLINT flow estimation. 3D rendering was used for the density and flow visualization (● ● ● colors representing x, y, and z flow directions respectively).*