CS231N Final Project

Erika MacDonald
emacd@stanford.edu

Baptiste Brugerolle
bbrug2hu@stanford.edu

Nael Ghoundale
naelgh22@stanford.edu

Abstract

Motivated by experimental constraints in fluid mechanics, we aim to reconstruct 3D flow fields from sparse 2D planar velocity data. In this final project, we use a Denoising Diffusion Probabilistic Model (DDPM) to predict missing flow information. The model takes in a subset of 2D velocity fields and outputs the predicted velocity fields at previously unobserved planes. We train and evaluate our method using Direct Numerical Simulation data from the Johns Hopkins Turbulence Database and compare its performance to a baseline approach of linear interpolation and a simple 3D-UNet. We find that the DDPM outperforms both baselines on our dataset. These results suggest that generative models like DDPMs offer a promising method for reconstructing 3D flow from limited planar data, with applications in both experimental and computational fluid dynamics.

1. Introduction

Predicting accurate and complete 3D flow fields is essential in fluid dynamics research and engineering applications. However, obtaining full volumetric measurements remains experimentally challenging, costly, and often impractical. Standard techniques such as Particle Image Velocimetry (PIV) typically offer precise data only at specific planar cross-sections. Tomgographic PIV allows for full velocity data in a 3D volume, but requires multiple cameras, careful calibration, high computational cost, while typically achieving lower spatial resolution than 2D PIV.

In this final project, we are motivated to develop a method that infers 3D flow structure from sparse planar measurements, which could reduce experimental complexity and cost. Aside from experimental fluid mechanics, such a method would have implications for computational fluid dynamics. In computational fluid dynamics, Direct Numerical Simulation (DNS) of the Navier Stokes equation can produce high-fidelity and time-resolved 3D flow data, but the resulting datasets can be large and thus costly to store. If 3D fields could be reconstructed from a sparse set of 2D slices, this could allow for reduced storage without significant loss of information. Additionally, related techniques could be applied to super-resolve coarse spatial or temporal DNS data, which could lower the computational cost of simulations by inferring the fine-scale dynamics from lower resolution input data.

For this final project, we aim to reconstruct 3D flow fields from sparse 2D planar velocity data. Specifically, the input to our algorithm is a set of 2D velocity fields obtained from DNS data, with each 2D velocity field corresponding to a cross-section of the 3D domain. These inputs are spatially sparse, and we then use a Denoising Diffusion Probabilistic Model (DDPM) to infer the missing velocity fields at the unobserved cross-sections. The output of our model is a set of predicted 2D velocity fields at the previously unobserved planes, which allows for partial reconstruction of the full 3D velocity field.

2. Related Work

We organize the related work into three categories: machine learning in fluid mechanics, inpainting and generative modeling with diffusion models, and applications of DDPMs to volumetric or 3D data.

2.1. Machine Learning in Fluid Mechanics

Machine learning has been increasingly used in fluid mechanics to tackle a wide range of tasks, including reconstructing missing data, accelerating simulations, and developing reduced-order models [1, 10]. One relevant application is the reconstruction of high-resolution velocity fields from low-resolution measurements. Fukami et al. [3] used convolutional neural networks to reconstruct higher resolution 2D velocity fields from coarse 2D input data. Their architecture used skip connections similar to those used in U-Net models. These skip connections helped model the multi-scale nature of turbulent flow, which improved their results compared to a standard CNN.

Physics-informed methods have also been developed to help models learn with physical constraints. Raissi et al. [9], introduced physics-informed neural networks, which incorporates governing equations, like the Navier-Stokes equation, directly into training. Similarly, Jiang et al. [5] used a physics-informed method to learn turbulence closure

models.

These studies informed our approach by showing that machine learning can learn spatial and temporal correlations in fluid flow. Further, these studies show how an understanding of the governing equations can improve models. However, most of these methods include 2D reconstructions, whereas our goal is to reconstruct 3D structure from sparse slices of 2D data.

2.2. Inpainting and Generative Models for Missing Data

Our problem is also related to image inpainting, where the goal is to reconstruct missing parts of an image. Thus, we additionally investigated work related to data inpainting with diffusion models. The original Denoising Diffusion Probablistic Models (DDPM) paper by Ho et al. [4] introduced a method to learn the probability distribution of data by adding noise to data and having a neural network learn to reverse the noise process. Building off of this, Lugmayr et al. [8] developed a DDPM based inpainting method that can take in arbitrary masks and generalizes better for a variety of masks in 2D image inpainting. While our project involves a fixed set of masks (masking some 2D slices and not others), this paper was helpful for understanding how DDPMs can be used in the presence of missing data.

2.3. 3D extensions of Diffusion Models

Although DDPMs have been mostly used for 2D image generation, there have been a few applications to 3D data. Dorjsembe et al. [2] synthesized 3D medical images using a 3D DDPM. This study showed how DDPMs could be used in a 3D context by replacing all of the 2D operations in a U-Net architecture with the associated 3D operations. This was helpful in that it showed that a similar implementation could work for our problem.

Karnewar et al.[6] trained a 3D diffusion model using 2D images, but their 2D input images are 2D projections of 3D volumes. In contrast, our work uses 2D cross-sectional slices rather than projections. This more closely resembles the setup in experimental fluid mechanics, where slices of velocity fields are obtained using a thin laser sheet for illumination.

3. Problem Statement

In this final project, we address the problem of 3D inpainting for direct numerical simulation (DNS) data with the goal of reconstructing missing 2D slices of the velocity field. The input to our model will be a masked 3D velocity field where some 2D slices along the depth dimension are known and others are missing. We will train a 3D DDPM to reconstruct the full 3D volume given the available slices.

We will have two baseline methods to evaluate the effectiveness of our model. The first baseline will simply be a linear interpolation of the velocity fields between the unmasked 2D slices. The second baseline will use a 3D CNN without diffusion to assess the impact of using a DDPM. We will compare the performance between these methods using mean squared error.

4. Dataset

Although the goal of this project is to improve experimental methods, we will first build a model using results from direct numerical simulation (DNS) given the time constraints for the project and the availability of data. We are using data from the Johns Hopkins Turbulence Database, which includes space-time history of DNS of isotropic turbulence, fully developed turbulent channel flow, homogeneous buoyancy driven turbulence, and more [7]. There is about 100 TB of data available for most of these cases. To start with, we are starting with the fully developed turbulent channel flow. This dataset has a domain length of $L_x \times L_y \times L_z = 8\pi h \times 2h \times 3\pi h$ where h is the half channel height. x, y, and z are nondimensionalized by this half channel height such that

$$x \in [0, 8\pi], \quad y \in [-1, 1], \quad z \in [0, 3\pi]$$

In Fig. 1, we display the streamwise velocity fields for a slice of this data in the center of the channel ($y = 0$) and for an adjacent slice, when $y = 0.25$. The total stored grid resolution of this dataset is $N_x \times N_y \times N_z = 2048 \times 512 \times 1536$ with a DNS timestep of $\Delta t = 0.0013$ and a stored time step of $\delta t = 0.0065$, with a total of 4000 stored time steps.

However, for the sake of this project, we will only use $X \times Y \times Z = 32 \times 32 \times 32$ data points taken at $N = 64$ time snapshots from $t_{start} = 0.1$ to $t_{end} = 5$. We divided up the dataset into a training set (90%) and a validation set (10%) to help us counter potential overfitting.

To achieve training the model on only a few "observed" streamwise planes, we use a mask to designate these observed planes (mask = 1) and to treat the remaining planes as missing (mask = 0).

5. Methods

5.1. DDPM

From the previously discussed dataset, we extract a 5D tensor of shape (C, N, X, Y, Z) where $C = 6$ is the number of channels consisting of u_x, u_y, u_z, as well as the x, y and z meshgrid. N denotes the number of time snapshots these quantities have been recorded. X, Y and Z denote the 3 spatial dimensions of the problem. We globally preprocess the velocities to have zero mean and unit variance, a standard for diffusion-based conditional inpainting approach:

$$\overline{u_c} = \frac{u_c - \mu_c}{\sigma_c}, \quad \text{for } c \in \{x, y, z\}$$

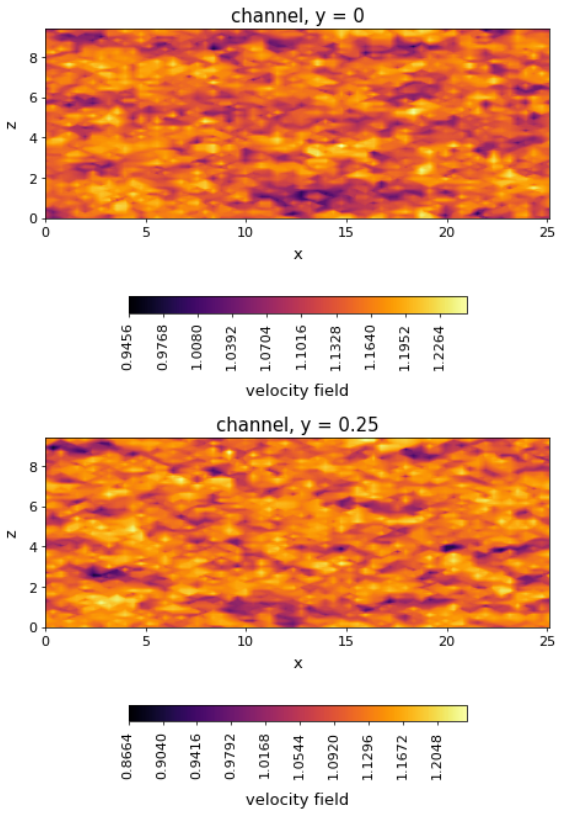

Figure 1. Two slices of nearby velocity fields.

where μ_c and σ_c are the global mean and standard deviation for channel c. When training, we extract the wanted time snapshot from this 5D tensor to obtain a part of the input of the model: a 4D tensor $\overline{u}$ of size (C, X, Y, Z).

To get the actual conditional input x_{cond} of our model, we first simulate incomplete observations by creating binary masks $M \in \{0,1\}^{1\times X\times Y\times Z}$ that randomly zero out a percentage of the cross-sectional slices along the Y-axis. The masked input is defined as:

$$u_{masked} = \overline{u} \cdot M$$

We concatenate the mask and spatial grid coordinates to the masked velocity, producing:

$$x_{input} = \text{concat}\,(u_{masked}, M, grid) \in \mathbb{R}^{7\times X\times Y\times Z}$$

Then, we define a forward diffusion process, inspired by DDPMs, that gradually perturbs the normalized ground truth velocity field $\tilde{u}$ with Gaussian noise. For a given timestep $t \in 1, \ldots, T$, the noisy sample is:

$$x_t = \sqrt{\bar{\alpha}_t} \times x_0 + \sqrt{1-\bar{\alpha}_t} \times \epsilon$$

where $\epsilon \sim \mathcal{N}(0, I)$ is Gaussian noise, $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ and β_t is such that it linearly increases from β_{start} to β_{end} over T steps. Note here that the subscript t represents the current diffusion step, and not physical time. This gives a x_{noisy} tensor of size $(3 \times X \times Y \times Z)$

Finally, we concatenate x_{noisy} with x_{input}, producing a conditional input:

$$x_{cond} = \text{concat}\,(x_{noisy}, x_{input}) \in \mathbb{R}^{10\times X\times Y\times Z}$$

Our denoising model ϵ_θ is a 3D U-Net designed for volumetric data. It takes the concatenated tensor x_{cond} as input and outputs a 3-channel tensor estimating the noise ϵ. The U-Net includes:

- Encoder Path: Two convolutional blocks with instance normalization and SiLU activations (a standard in DDPM), followed by max-pooling layers for downsampling.
- Bottleneck: A deeper convolutional block without downsampling.
- Decoder Path: Transposed convolutions for upsampling and skip connections from the encoder for spatial detail preservation.
- Output Layer: A $1 \times 1 \times 1$ convolution to produce a final 3-channel prediction.

To actually train the model, we want to minimize the following MSE:

$$\mathcal{L}_{MSE} = \mathbb{E}_{x_0,t,\epsilon}\left[\|\epsilon - \epsilon_0\,(x_t, x_{cond}, t)\,\|^2\right]$$

At each iteration, we sample a timestep $t \sim \text{Uniform}(1, T)$ and draw Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$. This loss encourages the model to learn to reverse the diffusion process and recover the original field from any noisy intermediate x_t.

Finally, to infer new slices using our model, we generate a sample starting from pure Gaussian noise $x_T \sim \mathcal{N}\,(0, I)$ and iteratively denoise it using the learned model. At each step t, we compute:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \cdot \epsilon_\theta(x_t, x_{cond}, t)\right) + \sqrt{\beta_t} \cdot z$$

where $z \sim \mathcal{N}(0, I)$ if $t > 1$, otherwise $z = 0$. After T steps, the output x_0 is a fully reconstructed velocity field consistent with the initial masked slices.

Training is done for 20 epochs using the AdamW optimizer (learning rate of 10^{-3}, decaying weights of 10^{-5}, with a $90/10$ train/validation set split. For initial testing, we mask 50% of the slices.

5.2. Baseline: Linear Interpolation

We apply a non-learning method based on 1D linear interpolation along the Y-axis. We start with the same preprocessed input from the DDPM.

For each missing slice $y = y_i$, we identify the two nearest observed slices y_a and y_b containing the slice, and interpolate linearly:

$$f(y_i) = \frac{y_b - y_i}{y_b - y_a} f(y_a) + \frac{y_i - y_a}{y_b - y_a} f(y_b)$$

If the missing slice lies outside the range of observed slices (i.e., before the first or after the last), we copy the nearest available observed slice $y_{nearest}$:

$$f(y_i) = f(y_{nearest})$$

where f represents the entire 2D slice of the velocity field at location y.

5.3. Baseline: 3D CNN Reconstruction with Partial Slices

We implemented a baseline using a 3D UNet to reconstruct missing velocity slices from partial observations. Each input volume contains 7 channels: the 3 velocity components, 3 spatial coordinates, and a binary mask indicating observed slices. During training, 4 random slices along the Y-axis are masked out per sample.

The model have a UNet 3D CNN architecture with skip connections and instance normalization. The model takes as input a volume with 7 channel, comprising masked velocity components (u_x, u_y, u_z), spatial coordinates (x, y, z), and a binary mask indicating observed slices. The architecture has three encoding and decoding blocks with skip connections, using InstanceNorm3d and ReLU activations. Training was done over 30 epochs using the Adam optimizer with learning rate 10^{-3}, batch size 2, and weight decay 10^{-5}. The loss function computes MSE only on the unobserved (masked) parts of the field:

$$\mathcal{L} = \frac{1}{\sum(1 - M)} \sum \left[(1 - M) \cdot (\hat{u} - u)^2\right]$$

Training is done for 30 epochs using Adam (learning rate 10^{-3}, batch size 2), with a 90/10 train/validation split. This baseline helps assess the performance of our DDPM by providing a supervised reconstruction reference under similar input constraints.

6. Results

Each of the three following sections is assigned a figure (4, 5, 6) that shows 9 plots, with all of them representing a slice at a fixed Y of the whole flow. The first column shows the Ground Truth u_x, u_y and u_z. The second column shows the Reconstructed / Predicted u_x, u_y and u_z. The last column shows a quantitative representation of the error between the Reconstructed / Predicted and the Ground Truth velocities, adding both qualitative and quantitative results.

6.1. Linear Interpolation

From figure 4, we can see that linear interpolation is a surprisingly efficient baseline. It assumes smooth variation of the velocity field along the Y axis and reconstructs missing slices by performing 1D linear interpolation between the nearest observed slices. Despite being purely algorithmic and requiring no training data, this method effectively enforces continuity in space. Given the nature of channel flow and the rough resolution in Y, this smoothness assumption can hold in some regions of the channel flow. The baseline was implemented using weighted averaging across neighboring slices, and it showed consistent performance in RMSE metrics across all velocity components.

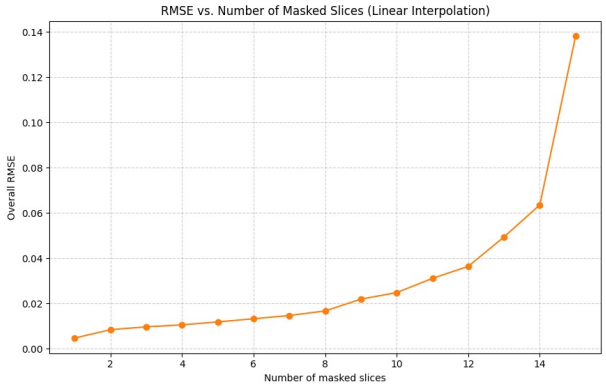

Figure 2. RMSE vs Number of Masked Slice

Masked Slices	RMSE
15	0.13810
14	0.06356
13	0.04940
12	0.03641
11	0.03111
10	0.02482
9	0.02194
8	0.01673
7	0.01472
6	0.01328
5	0.01189
4	0.01058
3	0.00972
2	0.00843
1	0.00475

Table 1. RMSE for varying numbers of masked slices

6.2. UNet

Despite access to a large dataset and spatial coordinates, the UNet baseline underperformed relative to linear interpo-

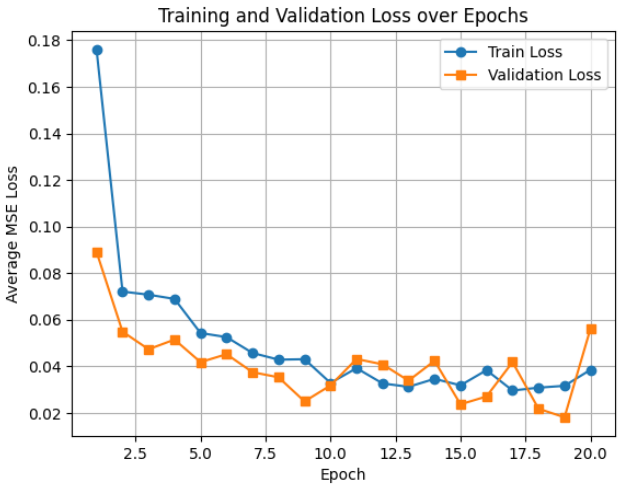

Figure 3. Loss vs Epochs (Test and Validation sets)

Epoch	Train Loss	Val Loss
1	0.175989	0.089104
2	0.072076	0.054862
3	0.070736	0.047305
4	0.068906	0.051460
5	0.054268	0.041779
6	0.052523	0.045186
7	0.045705	0.037464
8	0.042841	0.035262
9	0.043038	0.024918
10	0.032847	0.031779
11	0.039190	0.043129
12	0.032643	0.040832
13	0.031174	0.033826
14	0.034595	0.042339
15	0.031821	0.023664
16	0.038325	0.027038
17	0.029598	0.041786
18	0.030816	0.021771
19	0.031609	0.018090
20	0.038495	0.056204

Table 2. Training and validation loss over 20 epochs

lation according to figure 5. This is likely due to the ”general” nature of CNNs like UNet: the model learns patterns in data but has no understanding of physical quantities like velocity continuity or smoothness. In contrast, linear interpolation ”unwillingly” assumes a smooth gradient between slices, which inadvertently aligns better with physical expectations in certain flow regions. Moreover, the UNet must generalize across random masking configurations, which increases the difficulty of training. Its local convolution kernels and lack of long range context may limit its ability to resolve large field structures from sparse 2D slices inputs.

6.3. DDPM

The DDPM baseline outperforms both UNet and linear interpolation despite lacking explicit physical constraints. Its strength lies in modeling the full distribution of plausible velocity fields from sparse observations. The iterative denoising process captures long-range spatial dependencies, enabling globally coherent reconstructions. By learning from realistic turbulent flow samples, DDPM develops an implicit physical prior, yielding smoother and more accurate results than linear or convolutional methods. The Loss vs Epochs we obtain in figure 3 shows that the DDPM is well implemented: clear downward trend of the loss; a relatively smooth and stable training curve, meaning no exploding gradients; no major overfitting phenomenon present with the validation loss remaining rather low. With this, we can be confident about the validity of our DDPM.

7. Conclusions

In this project, we explored the reconstruction of 3D velocity fields from sparse 2D slices using generative modeling. We implemented and compared three approaches: a linear interpolation baseline, a supervised 3D UNet, and a Denoising Diffusion Probabilistic Model (DDPM).

Despite its simplicity, linear interpolation provided a strong baseline by implicitly assuming spatial smoothness, which can be partially valid in turbulent channel flows. The 3D CNN baseline, trained on a relatively large dataset and augmented with spatial coordinates, underperformed due to its lack of physical inductive bias and inability to capture long range dependencies across slices.

The DDPM was able to progressively refine its predictions through a learned denoising process, using the statistical structure of the data to generate coherent and better quality velocity fields. Rather than relying on hard coded physics, the model achieved to implicitly captured more global flow patterns by training on the dataset.

These results highlight the potential of DDPM for volumetric inpainting in fluid mechanics. Future work could involve adding physical constraints directly into the diffusion process, training on much larger datasets and testing the model on experimental PIV data to assess its generalization beyond simulation.

8. Contributions

- **Erika MacDonald**: Dataset preprocessing, Literature review, Report, Baselines and DDPM Design.
- **Nael Ghoundale**: Implementation of the 3D CNN (U-Net) baseline, development of the linear-interpolation baseline, Report and dataset preprocessing.
- **Baptiste Brugerolle**: Design, implementation of the full DDPM codebase, Report, dataset preprocessing.

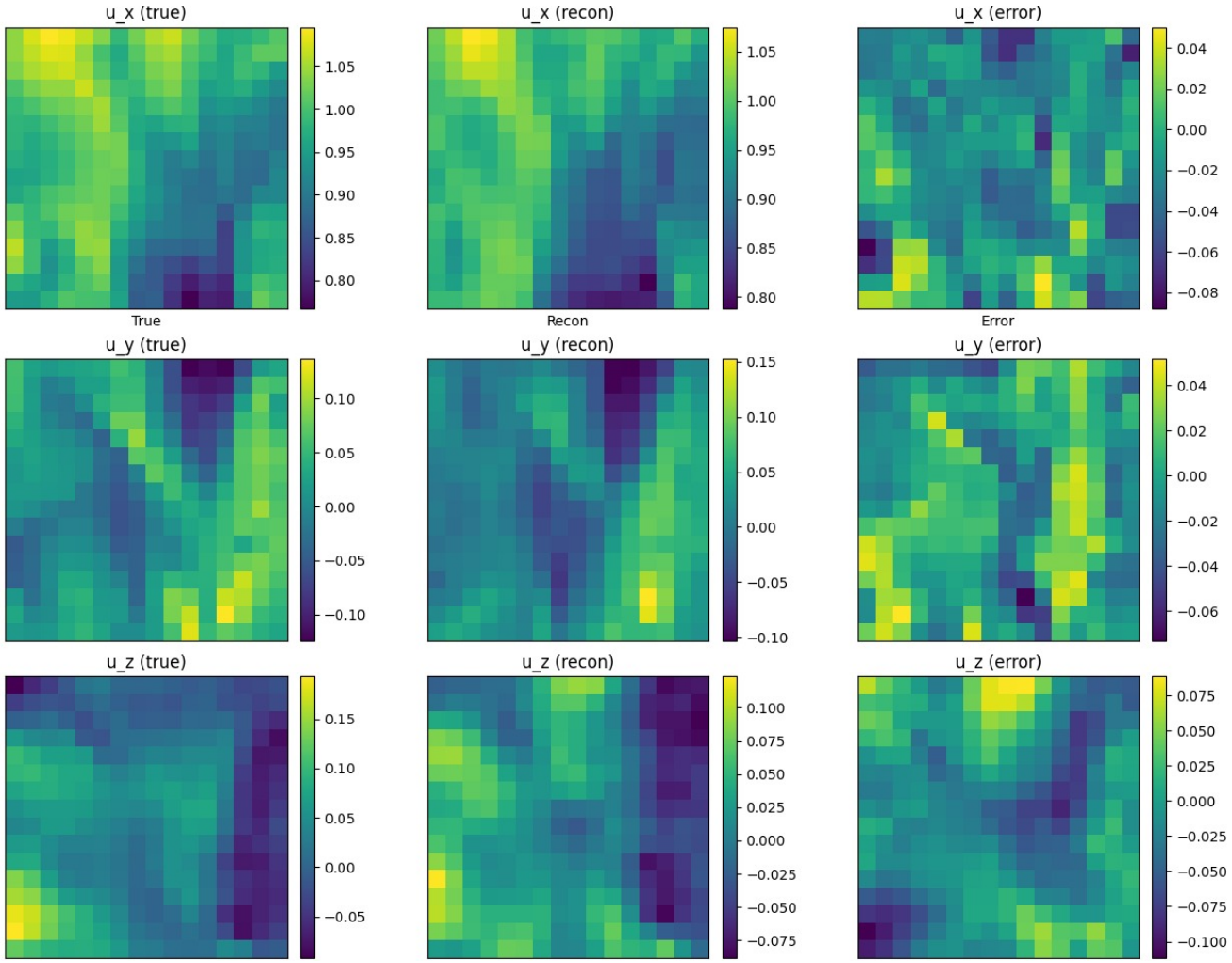

Figure 4. **LINEAR BASELINE:** Evaluating u_x, u_y, u_z: Ground Truth vs. Reconstructed vs. Error

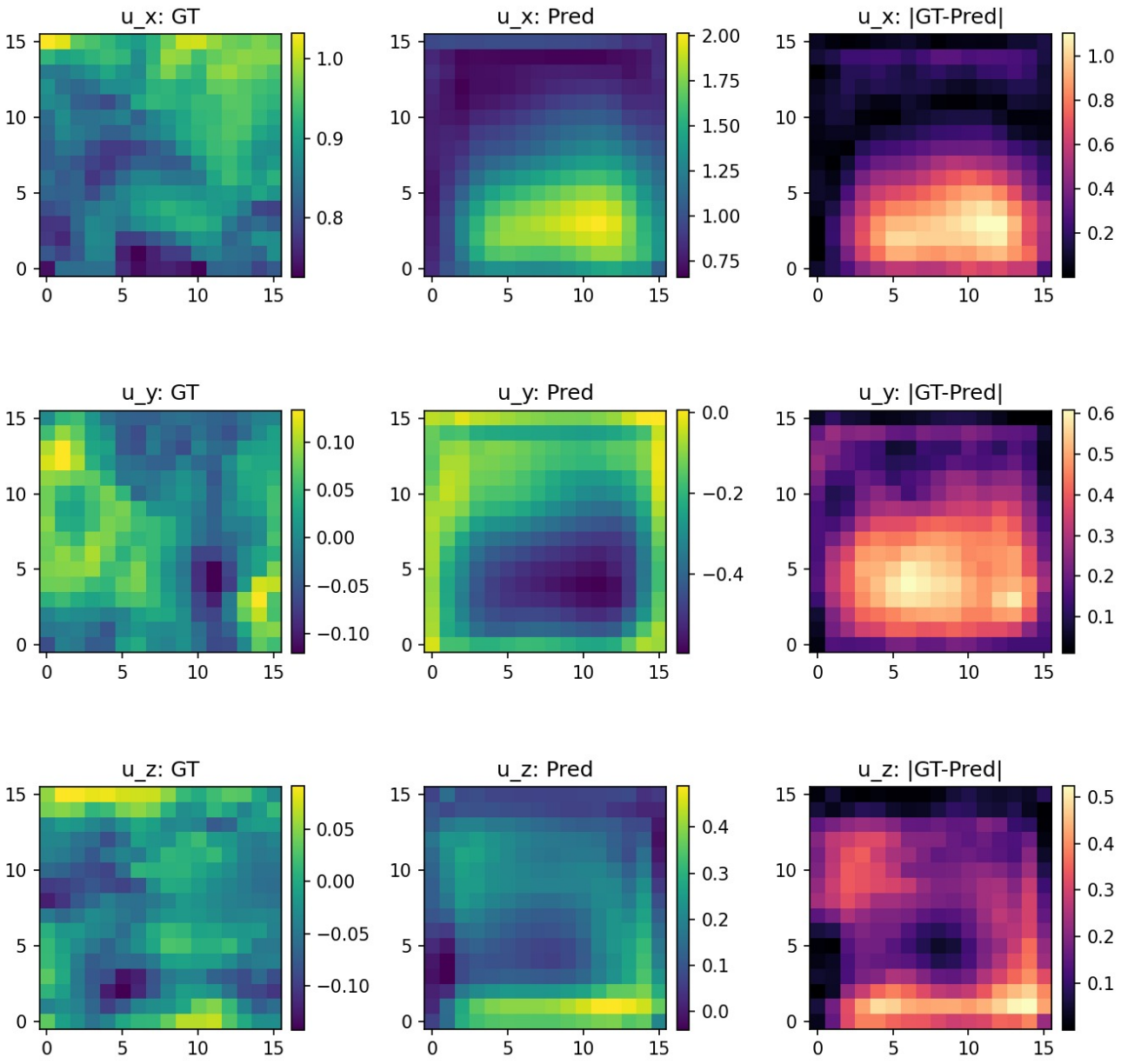

Figure 5. **UNET BASELINE:** Evaluating u_x, u_y, u_z: Ground Truth vs. Reconstructed vs. Error

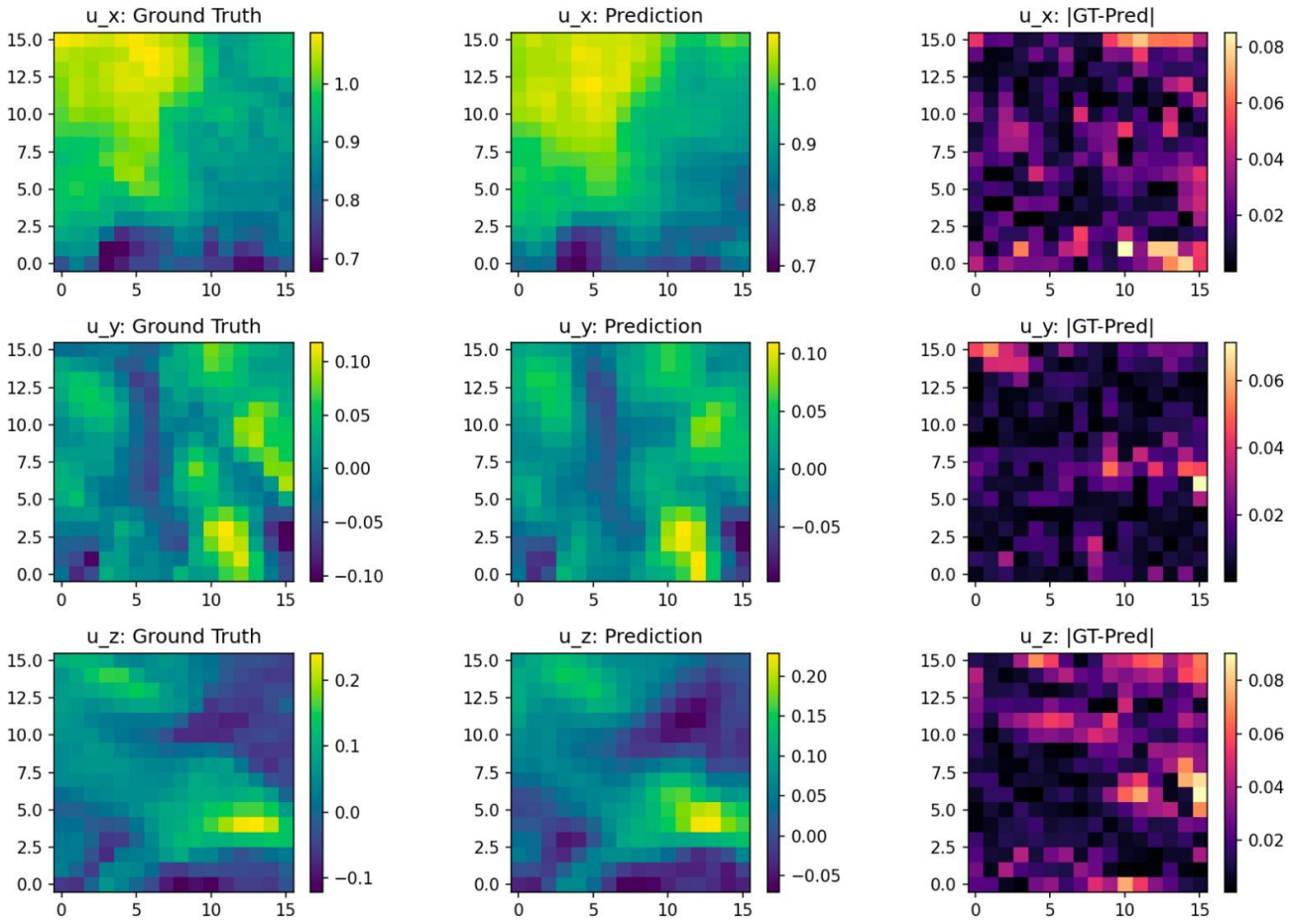

Figure 6. **DDBM BASELINE:** Evaluating u_x, u_y, u_z: Ground Truth vs. Reconstructed vs. Error

References

[1] S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52(1):477–508, 2020.

[2] Z. Dorjsembe, S. Odonchimed, and F. Xiao. Three-dimensional medical image synthesis with denoising diffusion probabilistic models. In *Medical imaging with deep learning*, 2022.

[3] K. Fukami, K. Fukagata, and K. Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019.

[4] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[5] C. Jiang, R. Vinuesa, R. Chen, J. Mi, S. Laima, and H. Li. An interpretable framework of data-driven turbulence modeling using deep neural networks. *Physics of Fluids*, 33(5), 2021.

[6] A. Karnewar, A. Vedaldi, D. Novotny, and N. J. Mitra. Holodiffusion: Training a 3d diffusion model using 2d images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18423–18433, 2023.

[7] Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, and G. Eyink. A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence*, (9):N31, 2008.

[8] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022.

[9] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[10] R. Vinuesa and S. L. Brunton. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366, 2022.