Utah State University

DigitalCommons@USU

All Graduate Theses and Dissertations, Fall 2023 to Present

Graduate Studies

12-2024

Interpreting Neural Networks for Particle Tracing in Fluid Simulation Ensembles: An Interactive Visualization Framework

Maanav Choubey
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd2023

Part of the Artificial Intelligence and Robotics Commons

Recommended Citation

Choubey, Maanav, "Interpreting Neural Networks for Particle Tracing in Fluid Simulation Ensembles: An Interactive Visualization Framework" (2024). *All Graduate Theses and Dissertations, Fall 2023 to Present*. 333.
https://digitalcommons.usu.edu/etd2023/333

This Thesis is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Theses and Dissertations, Fall 2023 to Present by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.

INTERPRETING NEURAL NETWORKS FOR PARTICLE TRACING IN FLUID SIMULATION ENSEMBLES: AN INTERACTIVE VISUALIZATION FRAMEWORK

by

Maanav Choubey

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Steve Petruzza, Ph.D. Major Professor	Soukaina Filali Boubrahimi, Ph.D. Committee Member
John Edwards, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2024

Copyright © Maanav Choubey 2024

All Rights Reserved

ABSTRACT

Interpreting Neural Networks for Particle Tracing in Fluid Simulation Ensembles: An Interactive Visualization Framework

by

Maanav Choubey, Master of Science

Utah State University, 2024

Major Professor: Steve Petruzza, Ph.D.
Department: Computer Science

Understanding the internal mechanisms of neural networks, particularly Multi-Layer Perceptrons (MLP), is essential for their effective application in a variety of scientific domains. In particular, in the scientific visualization domain their adoption has recently shown to be a promising tool to predict particle trajectories in fluid dynamics simulation and aid the interactive visualization of flows. This research addresses the critical challenge of interpretability of such models.

While interpretability has been extensively explored in fields like computer vision and natural language processing, its application to time series data, particularly for particle tracing (or prediction of trajectories), has not garnered sufficient attention.

The overarching objective of this thesis is to augment the interpretability of MLP networks through interactive and comparative visualization of model ensembles. We aim to contribute to address the challenges associated with the "black-box" nature of neural networks in this specific context. Our primary contribution lies in the development of a comprehensive visualization tool that integrates multiple linked views, including gradient visualization, particle trajectories, layer-wise activation, and weights visualization. This tool facilitates a more profound understanding of the intricate relationships between model components and model predictions.

In particular, the proposed framework provides a user-friendly interface for comparing different models trained to predict particle trajectories in fluid dynamics simulation ensembles.

This tool not only aids in understanding the MLP network behaviour, but also serves as a practical resource for researchers and practitioners wanting to analyze and use similar models.

Finally, we test our framework using a variety of different 2D flows with different degrees of complexity. This helps understanding the effectiveness of the tool in providing insights about which components of the model affects a particular prediction and also what the network is learning at different training epochs.

(60 pages)

PUBLIC ABSTRACT

Interpreting Neural Networks for Particle Tracing in Fluid Simulation Ensembles: An Interactive Visualization Framework

Maanav Choubey

Understanding the internal mechanisms of neural networks, particularly Multi-Layer Perceptrons (MLP), is essential for their effective application in a variety of scientific domains. In particular, in the scientific visualization domain their adoption has recently shown to be a promising tool to predict particle trajectories in fluid dynamics simulation and aid the interactive visualization of flows. This research addresses the critical challenge of interpretability of such models.

While interpretability has been extensively explored in fields like computer vision and natural language processing, its application to time series data, particularly for particle tracing (or prediction of trajectories), has not garnered sufficient attention.

The overarching objective of this thesis is to augment the interpretability of MLP networks through interactive and comparative visualization of model ensembles. We aim to contribute to address the challenges associated with the "black-box" nature of neural networks in this specific context. Our primary contribution lies in the development of a comprehensive visualization tool that integrates multiple linked views, including gradient visualization, particle trajectories, layer-wise activation, and weights visualization. This tool facilitates a more profound understanding of the intricate relationships between model components and model predictions.

In particular, the proposed framework provides a user-friendly interface for comparing different models trained to predict particle trajectories in fluid dynamics simulation ensembles.

This tool not only aids in understanding the MLP network behaviour, but also serves as a practical resource for researchers and practitioners wanting to analyze and use similar models.

Finally, we test our framework using a variety of different 2D flows with different degrees of complexity. This helps understanding the effectiveness of the tool in providing insights about which components of the model affects a particular prediction and also what the network is learning at different training epochs.

To the tales shared, the lessons learned, and the quiet shadows who will not read this.

ACKNOWLEDGMENTS

I am deeply grateful to my supervisor, Dr. Steve Petruzza, whose exceptional support and guidance were fundamental throughout this research endeavor. His expertise and encouragement have been invaluable in shaping this work. I would also like to express my gratitude to Dr. Soukaina Filali Boubrahimi, Dr. John Edwards, and Dylan Cashman for their encouragement and engaging collaborations, which have pushed the boundaries of my research.

I express my deepest appreciation to my parents, whose constant support and encouragement have been the cornerstone of my journey. I am also thankful to the master's students from the 2023 and 2024 batches for their guidance and camaraderie, which have consistently motivated me. Their perspectives and experiences have significantly contributed to my development.

I am grateful to the CS department at USU for their support and resources, which have played a crucial role in facilitating this research.

Lastly, I am thankful to the One who has pushed me to achieve this milestone and become the person I am today.

Maanav Choubey

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

In the field of predictive modeling, neural networks, such as multilayer perceptrons (MLPs), have found adoption in various fields. Their applications in scientific visualization encompass a wide range of uses beyond predicting particle trajectories in fluid dynamics simulations. For instance, neural networks are utilized in tasks such as feature extraction, anomaly detection, and dimensionality reduction in scientific data visualization. Notably, there exists a paper by Wang and Han [3] which systematically reviews the use of deep learning methods in scientific visualization tasks, offering insights into the current trends, challenges, and opportunities in this rapidly evolving field.

This research addresses the huge challenge of improving the interpretability of neural networks, with a special focus on particles tracking in fluid simulation groups. Although interpretation methods have been widely studied in areas such as computer vision and natural language processing, their adaptation to time series data, especially in predicting trajectories, has remained relatively unexplored.

The main goal of this thesis is to enhance the interpretability of MLP networks by creating an interactive web-based visualization framework. The goal of this framework is to provide researchers and practitioners with more insight into model behavior and predictions, thereby enhancing confidence and facilitating informed decision making in scientific endeavors.

Our tool includes interactive views to: (i) summarize differences in models parameters; (ii) explore similarity and evolution (over different epochs) of gradients, predicted vector fields and trajectories using different models; (iii) compare weights and activation values across models.

The contributions of this project are summarized as follows:

- metrics to compare activation of neurons between two models;

- development of a versatile visualization tool in the guise of a multi-views application that facilitates comparative studies of a given neural network model for particle tracing;
- an evaluation study using the tool to understand the learning process of models trained on different flow simulation ensembles.

This visualization tool not only facilitates the understanding of MLP network behavior but also serves as a pragmatic resource for researchers and practitioners working in the scientific visualization arena. By providing insight into model predictions and internal mechanisms, it enables users to verify, interpret, and refine neural network models to improve accuracy and reliability in scientific fields.

In the following sections, we delve into the underlying methodology of our visualization framework and explore methods used to improve interpretability and facilitate comparative analysis of neural network models. Next, we present experimental results that demonstrate the effectiveness of our approach in various scenarios. Finally, we consider potential implications and future research opportunities in the context of explanatory artificial intelligence and scientific visualization.

CHAPTER 2
BACKGROUND

In this section, we provide an overview of existing work related to the interpretability of neural networks and visualization techniques, with a specific focus on particle tracing in fluid dynamics simulations.

Existing work in the field of neural network interpretability has made significant strides, with numerous studies exploring methods to elucidate model behavior and predictions. Molnar et al. [4] offer a comprehensive review of interpretability techniques across various domains. They discuss the evolution of interpretability methods, from simple linear models to complex deep neural networks, emphasizing the importance of interpretability in ensuring trust and reliability in AI systems.

Guidotti et al. [5] provide a survey of methods for explaining black box models, focusing on techniques applicable to a wide range of domains, including healthcare, finance, and telecommunications. Their work highlights the need for interpretable AI systems, particularly in critical applications where model decisions directly impact human lives.

Visualization techniques play a crucial role in understanding neural network models. Zeiler and Fergus [6] proposed groundbreaking methods for visualizing and understanding convolutional neural networks (CNNs), notably multi-layered Deconvolutional Network (deconvnet), as proposed by (Zeiler et al., 2011) [7], to project the feature activations back to the input pixel space. Activation maximization involves optimizing input images to maximize the activation of specific neurons, thereby visualizing what features in the input data the network responds to. On the other hand, deconvolutional networks reverse the process of convolutional layers to generate visualizations of the features learned by different layers of the network. Their work not only shed light on the inner workings of CNNs but also provided insights into how these networks make decisions in image classification tasks. The techniques introduced by Zeiler and Fergus enabled researchers to gain a deeper understanding of CNNs' hierarchical feature representations and contributed to advancements in computer vision and image understanding.

While the work of Zeiler and Fergus primarily focuses on understanding CNNs for image classification tasks, this thesis extends the scope to the interpretability of Multi-Layer Perceptrons (MLPs) for time-series prediction tasks (i.e., predicting particles trajectories in a fluid dynamic simulation). Inspired by Zeiler and Fergus's approach of examining activation features from different layers, we adopt a similar strategy to analyze MLPs, aiming to understand how information is processed across various layers of the network. Unlike image classification, where CNNs excel in learning hierarchical features from visual data, time series prediction involves analyzing sequential data and capturing temporal dependencies. Therefore, the visualization techniques and interpretability challenges in this context are distinct from those in image classification tasks.

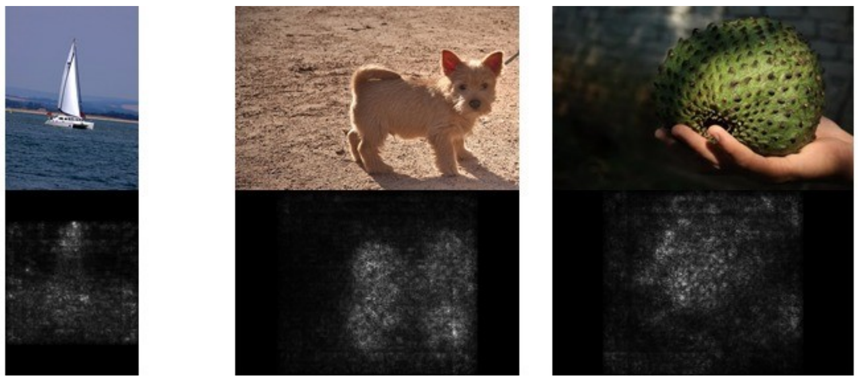


Fig. 2.1: Example of Saliency Maps for Image Classification. Image source: Simonyan et al. [1].

Selvaraju et al. [8] introduce Grad-CAM, a technique for generating visual explanations from deep networks via gradient-based localization. By visualizing class-specific activation maps, Grad-CAM provides insights into the regions of input images that contribute most to model predictions. This method has been widely adopted for interpreting CNNs in various computer vision tasks.

Simonyan et al. [1] and Olah et al. [9] delve into the use of gradients for interpretability in deep neural networks. They propose visualization techniques, such as saliency maps and feature visualization, to understand how changes in input data or model parameters influence model predictions. These methods offer valuable insights into the inner workings of neural networks, allowing researchers to interpret model decisions more effectively.

While numerous solutions exist for understanding the behavior of neural networks for tasks like classification and object detection, limited attention has been given to understanding how neural network models learn from time series data in general, and particularly for tasks such as fluid particle prediction using MLPs. In this work, we aim to bridge this gap by developing an interactive visualization framework tailored to the interpretability of MLP networks for particle tracing in both fluid simulation ensembles and double gyre datasets.

The proposed framework draws upon established techniques in interpretability and visualization, including activation maximization and deconvolutional networks as discussed by Zeiler and Fergus [6], which inspired the analysis of activation features across layers in Multi-Layer Perceptrons (MLPs). Additionally, techniques such as Grad-CAM introduced by Selvaraju et al. [8] for generating visual explanations from deep networks via gradient-based localization, provide a foundation for understanding the contributions of different parts of the input sequence to model predictions. Building on these techniques, our framework adapts them to the specific challenges and characteristics of time series prediction tasks

Furthermore, we introduce novel visualization techniques for the analysis of MLP networks for time series prediction. For example, inspired by existing work such as [10] by Chen et al, which visualizes the training of convolutional neural networks using Paraview, we develop interactive visualization tools for comparing models' performance and understanding the evolution of model predictions over time. Chen's work on TensorView provides insights into the visualization of neural network training processes, which we leverage to adapt visualization techniques to the analysis of MLP networks in time series prediction tasks. Additionally, we incorporate metrics such as histogram intersection similarity and performance and weight comparison, guided by the insights from TensorView and other related works.

In summary, while leveraging established techniques in interpretability and visualization, this thesis introduces novel methods tailored to the challenges of time series prediction in fluid dynamics simulations and contributes to advancing the understanding of neural networks' behavior in complex temporal domains.

2.0.1 Reference Multi-Layer Perceptron (MLP) Model

Before delving into the application of the Multi-Layer Perceptron (MLP) architecture for particle tracing in fluid simulation ensembles, it's essential to grasp the fundamental components of neural networks.

A neural network is a computational model inspired by the biological structure of the brain, composed of interconnected nodes or neurons organized in layers. Each neuron receives input signals, processes them, and generates an output signal. In the context of an MLP, neurons are arranged in layers, with each layer fully connected to the next. The basic building blocks of an MLP include:

- **Activations**: Neurons apply an activation function to the weighted sum of their inputs to introduce non-linearity into the model's predictions. Common activation functions include the rectified linear unit (ReLU), which allows the model to learn complex relationships in the data.

- **Weights**: Neurons in each layer are associated with weights, which represent the strength of connections between neurons. These weights are adjusted during the training process to minimize prediction errors.

- **Biases**: Each neuron is also associated with a bias term, which allows the model to capture patterns that may not be directly related to the input features. Biases are adjusted during training to fine-tune the model's predictions. Biases, which are constant, are an additional input into the next layer that will always have the value of 1. However, unlike the input features or weighted inputs, bias units are not influenced by the previous layer; they do not have any incoming connections. Hence, bias units are not visualized in our analysis, as they do not contribute to the visual representation of the network's learned features. The inclusion of biases in the neuron's computation is encapsulated in the equation:

$$y = \sigma(Wx + b)$$

where W represents the weights, x denotes the input features, b is the bias term, and σ is the activation function applied element-wise to the weighted sum.

With this understanding, we can now discuss the application of the MLP architecture in predicting particle trajectories within fluid simulations. The `MLP` model described here is designed to capture complex relationships between input features and predict particle trajectories. It comprises several fully connected layers, interleaved with normalization layers and ReLU activation functions to facilitate non-linear transformations of the input data.

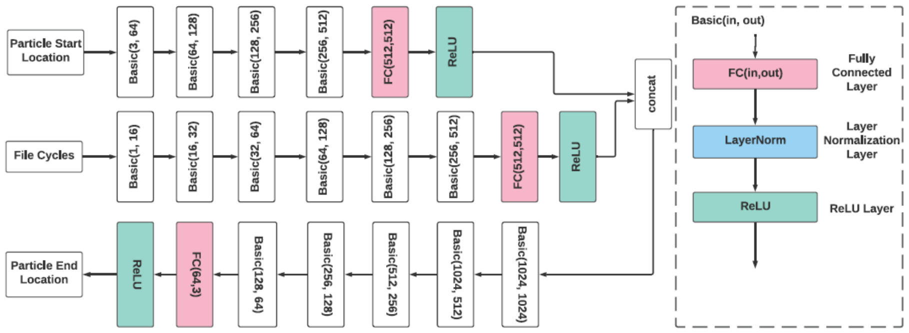


Fig. 2.2: The architecture of our neural network built with multi-layer perceptrons (MLP). The network takes the particle start location and the file cycles as input, and outputs the particle end locations. [2]

2.0.1.1 Network Architecture

The architecture and model design are adapted from the work of Han et al [2], which explores exploratory Lagrangian-based particle tracing using deep learning techniques. The network architecture, shown in Figure 2.2, consists of a latent encoder E and a latent decoder D. The latent encoder E and decoder D are built with MLP, a series of fully connected layers. The latent encoder E takes a particle's start location `start` and a queried file cycle C_j as inputs. These two parameters are separately fed into two sequences of fully connected layers of size (64, 128, 256, 512) and (16, 32, 64, 128, 256, 512). The two outputs are then concatenated together as a latent vector. Next, the latent decoder D, also a series of fully connected layers of size (512, 256, 128, 64), maps the latent vector to predicted end location `pred` at the queried file cycle. We added layer normalization Ba et al [11] after each fully connected layer except output layers to stabilize the training process. Moreover, we used the rectified linear unit (ReLU) Nair et al [12] as the activation function for each output from the fully connected layer.

2.0.1.2 Training Process

The training data sets are saved in the NPY file format for efficient loading in Python. We created a three-dimensional (3D) array, with dimensions of $[n+1; N; 3]$, for saving start seed locations and corresponding end locations at various file cycles. When loading the data sets, the data are organized into training samples, as shown in Equation 2.1. One training sample contains start location $start_i$ (where $i = 0, 1, \ldots, N-1$), the queried file cycle C_j (where $j = 0, 1, \ldots, n-1$), and the target end location at the queried file cycle $target_{i,j}$ (where $i = 0, 1, \ldots, N-1$ and $j = 0, 1, \ldots, n-1$). The start location and the queried file cycle are inputs to the network. The target end locations are used for calculating the loss (Equation 2.2). In addition to training data, we generated validation data by using $0.1 \times N$ seeds (10% of training samples) and following the same process.

$$Inputs = \left\{ \begin{array}{l} \{start_0, C_0, target_0, C_0\} \\ \{start_0, C_1, target_0, C_1\} \\ \vdots \\ \{start_0, C_{n-1}, target_0, C_{n-1}\} \\ \vdots \\ \{start_{N-1}, C_{n-1}, target_{N-1}, C_{n-1}\} \end{array} \right\} \tag{2.1}$$

[1] **Input:** Data set shown in Equation 2.1 **Initial weights of the network** w **Output:** Optimized weights w Load training data set each epoch each batch of training samples **model.train()** $pred \leftarrow model(start; queried\ file\ cycle)$ $loss \leftarrow L1Loss(pred; target)$ Backpropagation and update weight w each batch of validation samples **model.eval()** $pred \leftarrow model(start; queried\ file\ cycle)$ $loss \leftarrow L1Loss(pred; target)$ call learning rate scheduler to adjust the learning rate if needed

In this work we used the MLP neural network developed by Han et all [2] which is implemented using PyTorch [13]. The training process aims to find the optimized weights w of the network. The weights are initialized by PyTorch. We created a custom PyTorch Dataset class to load and store all training samples. We then loaded the PyTorch Dataset object into a PyTorch DataLoader for iterating through the training samples. At the beginning of each epoch, the training samples are shuffled and split into batches. Given a batch of training samples, the forward process computes the output following the network architecture and computes the loss as defined by the loss function. The backpropagation process is done automatically using PyTorch by calling `loss.backward()`, and the weights are updated by the optimizer. For our experiments, we trained the network for 100 epochs using the Adam optimizer Kingma and Ba [14] with the hyperparameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e - 6$. Further, in our training process, we set the initial learning rate to 10^{-5} and used a learning rate scheduler provided by PyTorch to reduce the current learning rate by a factor of 2 if the validation loss had not decreased for five epochs. We applied L1 loss as loss functions in our method. L1 loss calculates the mean absolute error between target and predicted end locations by the network.

$$L1\_Loss = |target - pred| \tag{2.2}$$

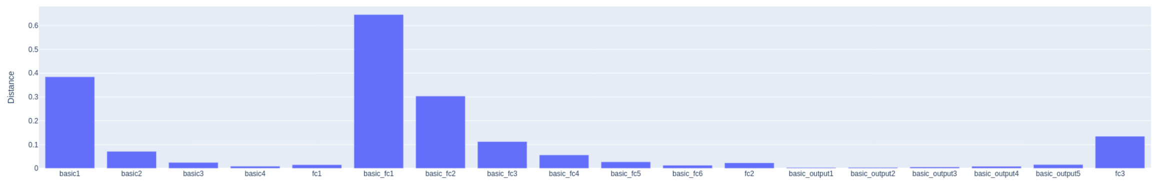
2.0.1.3 Similarity metrics

Exsisting metrics Existing methods for comparing neural network model layers, such as Euclidean distance or cosine similarity, are not suitable for our case due to several reasons. Firstly, these methods focus solely on the magnitude or angle between vectors representing layer activations, which may not adequately capture the similarities in activation distributions. Additionally, these methods do not consider the shapes of activation distributions, which are crucial for understanding the functional similarities between layers. As depicted in Figure 2.3, our analysis using cosine similarity initially suggested some degree of similarity across models, though at a basic level. However, further investigation, as illustrated in Figure 2.4, revealed discrepancies in detailed activation comparison. This discrepancy arises due to the nature of the cosine similarity, which is calculated as:

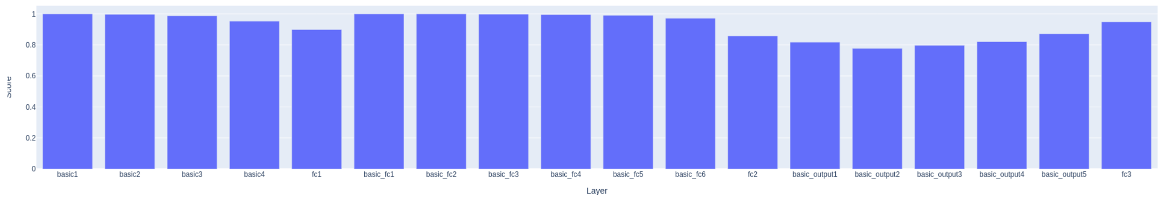
$$CosineSimilarity = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

where **A** and **B** are the vectors representing the activation values of two layers. When one of the activation values is zero, the denominator becomes zero, leading to undefined values (NaN) in the cosine similarity calculation. This phenomenon results in underestimated divergence between models, as the similarity metric tends towards infinity when one of the vectors is entirely zero. This limitation highlights the inadequacy of traditional metrics in capturing the intricate differences in activation patterns between models.

In Chen et. al [10], the authors utilize metrics such as Pearson correlation coefficient to quantify similarities between activation patterns in convolutional neural networks (CNNs) trained on image data. They flatten the 2D feature maps into 1D vectors and then normalize the activation values to [0,1], allowing for a comparison of accumulated activation values for each filter. However, our analysis involves Multi-Layer Perceptrons (MLPs) on temporal data, which poses distinct challenges and necessitates a different approach.



(a) Euclidean Distance



(b) Cosine Similarity

Fig. 2.3: Comparison of all layers using different measures

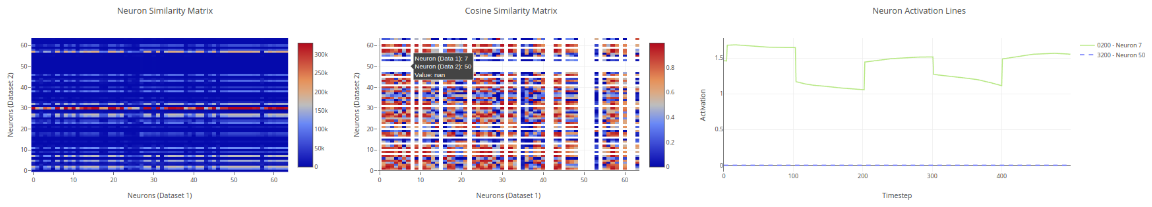


Fig. 2.4: Cosine Similarity comparison for individual layer

CHAPTER 3

Comparison and Visualization System

3.0.1 System Overview

The visualization system is designed as a comprehensive tool to enhance the interpretability of MLP networks for particle tracing in fluid simulation ensembles. At a high level, the system consists of the followingcomponents:

1. **Data Pre-processing Module**: This module converts raw model outputs into a format compatible with the the visualization and analysis components of our system. Upon receiving the output of a trained neural network model, the module undertakes essential tasks such as data formatting, trajectory prediction extraction, feature extraction (gradients for input coordinates and time), and weight extraction. Subsequently, this processed data is organized into dictionaries, wherein the keys serve to distinguish datasets from each other. These prepared datasets are then seamlessly integrated into our visualization system, facilitating efficient analysis and interpretation of MLP network behaviors for particle tracing in fluid dynamics simulations.

2. **Model Comparison Module**: Responsible for computing similarity metrics between different models.

3. **Visualization Interface**: Once the data has been pre-processed and model comparisons have been computed, the visualization interface provides users with interactive visualizations and tools to explore and analyze the model behavior and predictions. The visualization interface consists of multiple linked views, including histogram intersection for comparing complete layers based on activation values, gradient visualization, particle trajectories, layer-wise activation, and weights visualization. Users can interact with these views to compare models, visualize activation patterns, and

understand the impact of different model configurations on particle tracing in fluid dynamics simulations.

By integrating the data pre-processing module, model comparison module, and visualization interface, the visualization system enables researchers and practitioners to gain a comprehensive understanding of MLP networks for particle tracing, facilitating informed decision-making and interpretation in scientific applications.

3.0.1.1 Pre-processing

After loading the necessary libraries and defining utility functions, we perform pre-processing tasks to prepare the data and model predictions for further analysis. The following steps outline the pre-processing pipeline:

1. **Data Loading and Model Initialization:** Following the initialization of essential variables which stores activations and gradients, the test data and the trained model are retrieved from their designated directories. Subsequently, we utilize a custom PyTorch Dataset class, as detailed in 2.0.1.2, to load the data. This class facilitates efficient handling of the data, ensuring compatibility with our subsequent analysis.

2. **Layer Activation Capture:** We define a hook function, `hook_fn`, which allows us to capture the activity within each layer of our model. When we call this function, we specify the name of the layer we want to monitor. As data flows through that layer during the model's operation, the function records what's happening and stores that information for later analysis. This enables us to gain insights into the behavior of individual parts of our model.

3. **Model Inference and Gradient Calculation:** The code performs model inference on the test data using a DataLoader to efficiently process batches of data. For each batch, the model predicts the future states of particles based on their initial positions and timestamps. Additionally, gradients of the model predictions with respect to the

input data and timestamps are computed using ‘torch.autograd.grad‘ to analyze the sensitivity of predictions to input variations.

4. **Normalization and Reconstruction:** The predicted future states of particles are normalized based on predefined bounding values to ensure consistency across datasets. This normalization process, known as min-max normalization, scales the data to fit within a specific range, typically between 0 and 1. Specifically, each dimension (e.g., x and y coordinates) undergoes normalization using the formula:

$$normalized\_value = \frac{value - min\_value}{max\_value - min\_value} \times (max\_bound - min\_bound) + min\_bound$$

   This ensures that the minimum value becomes 0 and the maximum value becomes 1, while preserving the relative scale of the other values. Additionally, the results are reconstructed to match the original data format for visualization and analysis.

5. **Feature Importance and Gradient Analysis:** We compute the absolute values of gradients to assess the significance of input features, which include particle positions and time. This analysis helps us understand how changes in particle positions and time influence the model predictions.

The pre-processing steps ensure that the data and model predictions are appropriately formatted and analyzed before further exploration and visualization. These steps are essential for extracting meaningful insights from the raw model outputs and facilitating comparative analysis across different datasets and model configurations.

3.0.2 New Similarity Metrics

In addition to developing visualization techniques, we employ quantitative methods to compare neural network models. Two primary comparison metrics are utilized: histogram intersection for comparing complete layers based on activation values and the difference of weights across all neurons for weight comparison. We choose to compare activation values to compare complete layers because activation values reflect the output of each neuron after processing input data through the network, capturing the patterns and features extracted by the model during training.

Comparing activation values allows us to evaluate not only the structural similarities between layers but also the functional similarities in terms of how well the models capture and process information. In contrast, comparing weights or biases directly might not provide a comprehensive understanding of model similarities, as weights and biases are internal parameters that influence the transformation of input data but may not directly reflect the learned representations or the output of the model.

Given those limitations described in the previos Section we propose two new metrics to compare activation values across neurons of a given neural network layer. One metric is less computationally intensive and is used to produce summary comparisons for all the layers of two neural networks. The second metric is used to compare activations of two given neurons and is inspired by signal comparison metrics.

Model Similarity Metric The comparison of all the layers in a model is a way to summarize the similarity of two models. For comparing complete layers based on activation values, we chose to utilize the histogram intersection method as proposed in Bicciato et al [15]. This method is particularly relevant in our context due to its ability to measure the similarity between two histograms representing the activation distributions of corresponding neurons in different models. Unlike other comparison methods such as Euclidean distance or cosine similarity, histogram intersection directly compares the shapes of activation distributions rather than their magnitudes. This makes it well-suited for capturing similarities in activation patterns across different datasets, even if their scales or magnitudes vary. The histogram intersection similarity (I_H) between two histograms H_1 and H_2 is calculated using the formula:

$$I_H(H_1, H_2) = \sum_{i=1}^{n} \min(H_1(i), H_2(i))$$

where n is the number of bins in the histograms.

Layer Similarity Metric We introduce a custom similarity index function for computing similarity of activation functions. In particular, rather than directly comparing activation values between individual neurons, we treat activations as signals evolving over time. This approach allows us to capture nuanced differences in activation patterns between different models. Our similarity index function considers various aspects such as time domain, frequency domain, and power similarities, providing a more comprehensive comparison of activation dynamics. In particular, we treat activations as signals spread across entire batches for different datasets and models and adopt a methodology commonly used to compare signal similarity.

In doing so, we employ three primary metrics for comparing activation signals, and we combine these metrics using a weighted sum approach to provide a comprehensive assessment of similarity.

The three metrics utilized for comparing activation signals are:

1. **Cross-correlation**: Cross-correlation measures the similarity between two signals by computing the correlation of one signal with a time, frequency, and power-shifted version of the other signal. This method is suitable for comparing signals that exhibit similar temporal patterns. For instance, if we want signals that look visually similar, we perform time domain analysis.

2. **Histogram Intersection**: Histogram Intersection measures the similarity between two histograms representing the distributions of activation values in neural network layers. It quantifies the overlapping area between the histograms, providing a robust measure of similarity that is invariant to differences in scale and magnitude.

3. **Difference in Weights**: Difference in weights across two datasets provides a measure of dissimilarity between the weights of corresponding neurons in the neural network models. By computing the absolute difference in weights for each neuron between two datasets, we obtain a metric indicative of the divergence or convergence of weight values.

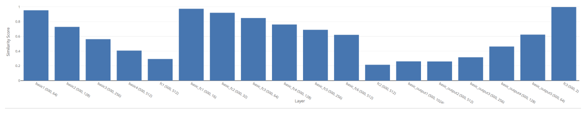
To compare activation signals from different neurons in a particular layer, we compute these metrics for each pair of signals and then combine the findings using a weighted sum method. The weights used for aggregation are usually chosen by the relative importance of each metric in capturing the desired characteristics of activation pattern comparison. In our situation, we allocated equal weights to each metric, ensuring that the total similarity score reflects a balanced evaluation of time domain, frequency domain, and power similarities. The weighted sum formula for aggregation is as follows:

$$WeightedSum = w_1 \times diff\_time + w_2 \times diff\_freq + w_3 \times diff\_power$$

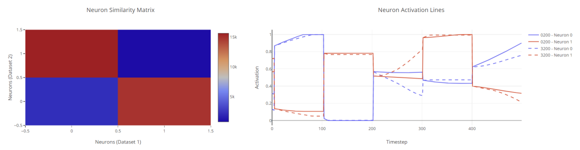
where w_1, w_2, and w_3 are the weights allocated to the metrics for time domain, frequency domain, and power similarity, respectively. In our scenario, each weight w_i is set to 0.33, ensuring equal relevance across all measures. This method allows us to measure the similarity of activation patterns while taking into account a wide range of signal properties.

Weight comparison The comparison of weights is performed by calculating the difference of weights across all neurons in the network. This metric provides insights into the divergence or convergence of weight values between different models. In Figure 3.1c (c) we report an example of weight similarity heatmap. With this metric low values indicate high similarity.

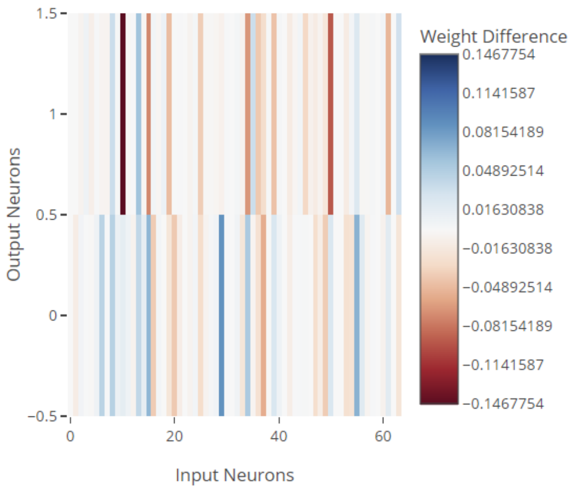
This elevated weights similarity suggests that neural networks are adept at learning generalizable features crucial for the task at hand. If the datasets represent similar tasks (e.g., classification, regression), the model may acquire similar weights to optimize its performance on those tasks, irrespective of the specific dataset. This assertion finds support in Wang et al [16], which examines the extent to which different neural networks learn comparable representations and discusses the implications for understanding the behavior of deep learning models. Additionally, Glorot and Bengio [17] delves into the challenges of training deep neural networks and discusses how distinct architectures can learn similar representations despite differences in depth.



(a) Layers similarity



(b) Activation Similarity



(c) Weight Similarity

Fig. 3.1: Similarity metrics example visualization.

In summary, the combination of layer activations and weights similarity help understand how similar the learned representations of two models really are.

3.0.3 Model Comparison

The Model Comparison component of our visualization system aims to provide researchers and users with the ability to compare models trained on different datasets. By examining various features and parameters of these models, users can gain insights into the training process, learning dynamics, and the effects of dataset variations on model performance. This component serves to facilitate a high-level understanding of the differences between two given models trained on distinct datasets, offering both an overview/summary of model disparities and a detailed comparison of parameters at the individual layer level.

By offering a range of comparison features, the visualization system enables users to explore and analyze various aspects of model behavior, including:

- Differences in activation patterns across layers and datasets.
- Variances in weight distributions and values between models.
- Changes in gradient magnitudes and directions over training epochs.
- Evolution of model predictions and trajectories in response to dataset variations.

3.0.3.1 Overview and Summary

At a high level, the Model Comparison component offers users an overview or summary of the differences between two given models. This summary may include key metrics such as overall accuracy, loss functions, and performance on specific evaluation criteria. Additionally, visualizations such as comparative histograms or line plots can illustrate how model predictions or trajectories vary across datasets.

The summary view aims to provide users with a quick and intuitive understanding of the broader differences between models, enabling them to identify areas of interest for further investigation.

3.0.3.2 Detailed Comparison

In addition to the 3.0.3.1, the Model Comparison component allows users to delve into more detailed comparisons related to the parameters of individual layers in the neural network models. This detailed comparison enables users to explore how specific components of the models differ across datasets, providing insights into the underlying mechanisms of model behavior.

Users can interactively select layers or specific parameters to compare, and the visualization system generates comparative plots or figures illustrating differences in activation patterns, weight distributions, gradient values, and other relevant parameters. These visualizations enable users to identify subtle variations in model behavior and understand how different datasets influence the learning process.

By offering both overview and detailed comparison capabilities, the Model Comparison component empowers users to comprehensively analyze and interpret differences between models trained on different datasets, facilitating informed decision-making and further research in scientific domains.

Summary Comparison The Summary Comparison section provides an overview of the comparative analysis of all layers across two different models. This analysis is facilitated using the Histogram Intersection method, as described in Section 3.0.2. In this method, the similarity between activation distributions of corresponding neurons in different layers is measured by quantifying the overlapping area between their histograms.

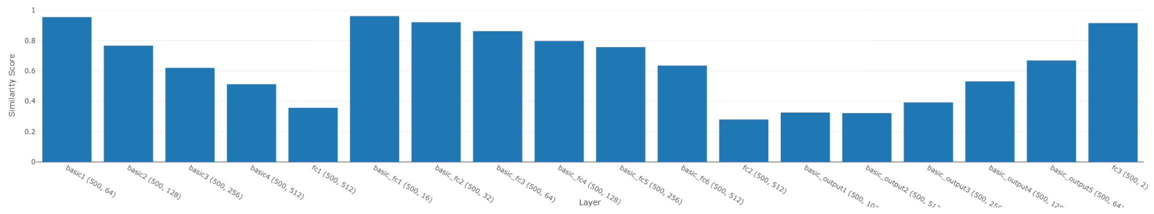


Fig. 3.2: Summary comparison of two models

In Figure 3.2, the x-axis of the summary comparison represents the different layers of the neural network, including input layers, hidden layers, and output layers. On the other hand, the y-axis indicates the similarity score for each layer, ranging from 0 to 1. A value closer to 1 denotes high similarity, indicating that the corresponding layers in the two models are highly similar, while a value closer to 0 signifies low similarity, suggesting significant differences between the layers.

To generate the summary comparison, histograms of activation distributions are computed for each layer in both models. These histograms capture the distribution of activation values across neurons within each layer. The Histogram Intersection method then quantifies the similarity between these histograms by calculating the overlapping area, providing a robust measure of similarity between corresponding layers in different models.

The resulting summary comparison provides researchers and users with a quick and intuitive understanding of the overall differences between the two models across all layers. By visualizing the similarity scores for each layer, users can identify areas of high and low similarity, guiding further investigation and analysis.

Activation and Weights Comparing activations and weights across different models is essential for understanding how the models learn and make predictions. We utilize the previously described metrics in Section 3.0.2 to compare activations and weights between two given models trained on different datasets.

For activation comparison, we compute an all-to-all comparison for all neurons in a given layer, visualized as a matrix as depicted in Figure 3.3 (left). Each cell in the matrix represents the similarity score between the activation patterns of two neurons. A higher similarity score indicates that the neurons have different activation patterns (in the left of the Figure), while a lower score suggests similarities in activation behavior. This visualization provides insights into how neurons in different models respond to input stimuli and how their activation patterns evolve during training.

Weights comparison is performed using individual visualizations, differential analysis, and all-to-all comparisons. Individual visualizations highlight the importance of different weights in the network as shown in Figures 3.4a and 3.4b, showing which connections

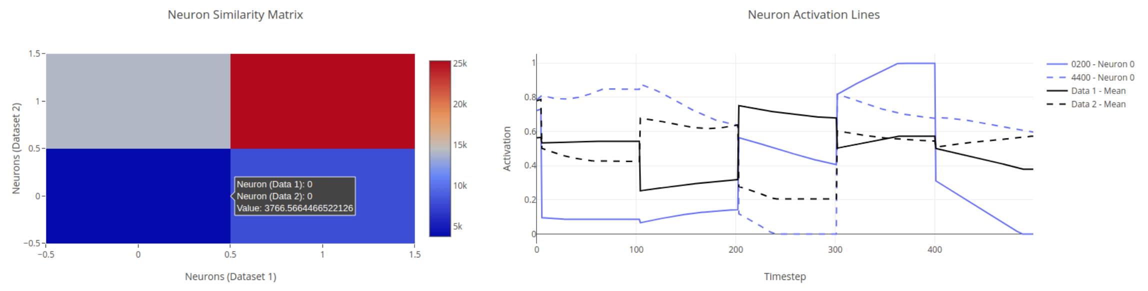
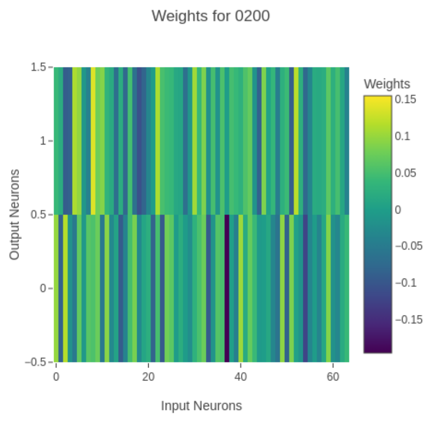


Fig. 3.3: Layer-wise activation heatmap (left) with hover functionality connecting. On the right, activation plots

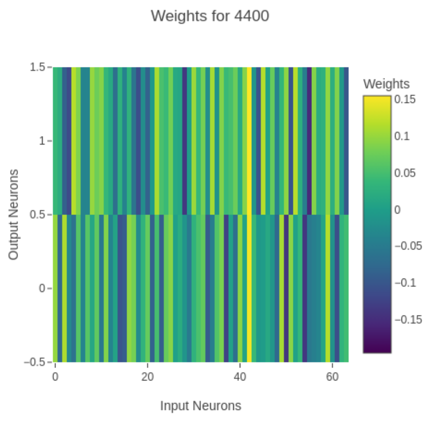
are most significant for predicting particle trajectories. Differential analysis compares the differences in weights between corresponding neurons in different models like in Figure 3.4c, revealing how the models adapt their weights to different datasets. All-to-all comparisons provide a comprehensive view of weight similarities across models as shown in Figure **??**, showing which weight configurations are consistent and which diverge between models.

Layer-wise Activation Understanding the activation patterns of different layers in the MLP network is crucial for interpreting model decisions. We visualize layer-wise activations to gain insights into how information flows through the network and which features are extracted at each stage of processing. By visualizing the activation patterns of all neurons for a selected layer and dataset across the entire batch and identify overall trends in activation behavior. Utilizing interactive features such as hover functionality (see Figure 3.3), users can selectively focus on and examine the activations of specific neurons, facilitating effective exploration of activation similarities between datasets.

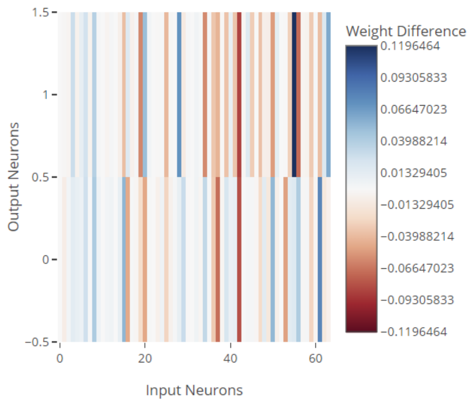
Weights Visualization Visualization of model weights plays a crucial role in understanding how neural networks learn and represent information. In this section, we employ visualization techniques, such as heatmaps, to elucidate the importance of different weights within the network architecture. These techniques enable us to highlight the connections that are most significant for predicting particle trajectories, which is the primary objective of our study.



(a) Weights of `fc3` layer for Model 1



(b) Weights of `fc3` layer for Model 2



(c) Difference in Weights of Model 1 and 2

Fig. 3.4: Different Weights Visualization for two datasets

By utilizing heatmaps, we can visually represent the importance of individual weights in the network, providing clear insights into which connections hold the most significance. This visualization method facilitates a deeper understanding of the network's learning process and how it contributes to accurate predictions.

Furthermore, we emphasize the significance of visualizing the distribution of weights across different layers of the neural network.

3.0.4 Gradients and Prediction Visualization

3.0.4.1 Vector Field Visualization

Visualization of vector fields plays a pivotal role in comprehending the intricate dynamics of fluid systems and their influence on the trajectories of particles. In this section, we delve into the detailed analysis of vector fields derived from various datasets, aiming to glean profound insights into how fluid motion shapes the movement of particles over time.

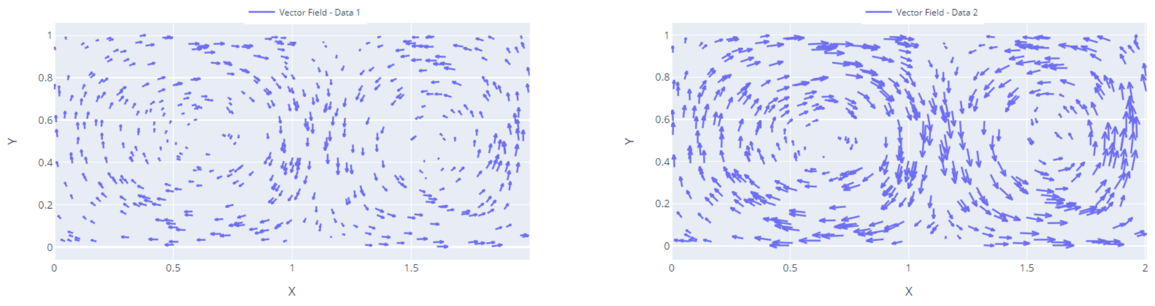
When investigating datasets with static vector fields, we adopt a comprehensive approach by scrutinizing vector fields spanning from the initial epoch (epoch 0) to the culmination of model training. This longitudinal analysis allows us to track the evolution of the vector field assimilation by the model over time, offering invaluable insights into the alterations in gradients and predictive patterns.

To illustrate the complexity of the vector fields across different spatial points within the simulation domain, Figure **??** presents an exemplary visualization. Here, vectors are depicted at various spatial locations, providing a comprehensive view of the ensemble datasets.

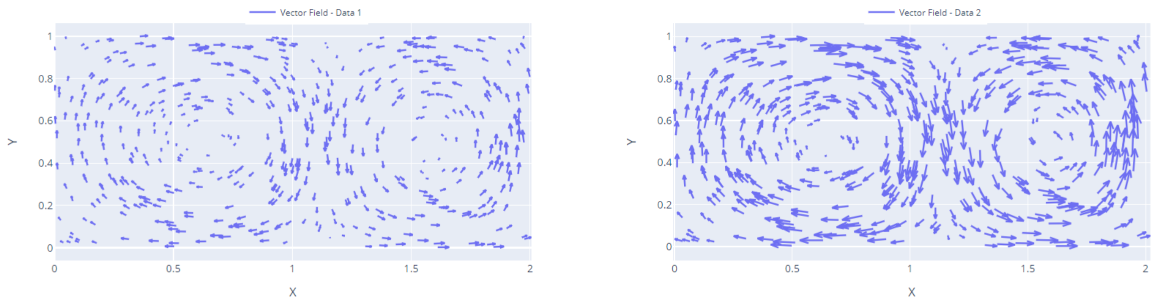
Moreover, for a more focused examination of the double gyre static datasets, we present specific visualizations in Figures **??**, 3.5a, and 3.5b. These figures delineate the progression of vector fields from the initial epoch to the final stages of model training, offering a nuanced understanding of how the vector field evolves throughout the modeling process.

3.0.4.2 Gradient Visualization

Interpreting the inner workings of neural network models is crucial for understanding



(a) Vector Field comparison of two similar models at epoch 24 of training



(b) Vector Field comparison of two similar models at epoch 50 (complete training).

Fig. 3.5: Comparison of vector fields derived by the predicted trajectories of two similar models.

their predictions and improving their reliability. In our research, we tackle the challenge of enhancing the interpretability of MLP networks, with a specific focus on particle tracking in fluid simulation groups.

Gradients play a fundamental role in understanding how changes in input features affect model predictions. In the context of neural networks, gradients represent the rate of change of a model's output with respect to its input parameters. Essentially, they tell us how much the output of the model will change if we make a small change to its input. By visualizing gradients using line plots, where the y-axis represents time and the x-axis represents the gradient value, we can observe how the gradient values evolve over time. This provides insights into the dynamics of the model's sensitivity to changes in input parameters as the simulation progresses.

Figure 3.6a illustrates an example of gradient visualization using line plots. Each line in the plot represents the gradient values of a specific input feature (such as particle position or time) over time. By analyzing these plots, researchers can identify trends and patterns in the model's sensitivity to different input parameters. For example, spikes or sudden changes in gradient values may indicate critical points in the simulation where small changes in input parameters have a significant impact on model predictions.

Another important aspect of our visualization is the interactive hover functionality, demonstrated in Figure 3.6b. When users hover over specific points on the line plot, the corresponding gradient value at that time step is highlighted. This feature allows for a detailed analysis of gradient values at different stages of the simulation, providing insights into how the model's sensitivity to input parameters changes over time.

Moreover, when comparing static datasets from double gyre simulations across different epochs, users can observe the gradients and their correlation with trajectories, providing deeper insights into the dynamics of fluid motion.

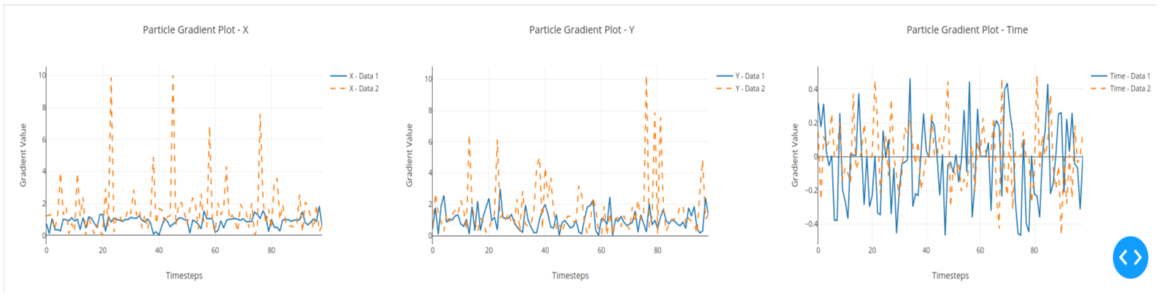
Overall, our visualization framework empowers researchers and practitioners with a versatile tool to enhance the interpretability of MLP networks, facilitating informed decision-making in scientific endeavors involving fluid dynamics simulations.

3.0.4.3 Particle Trajectories

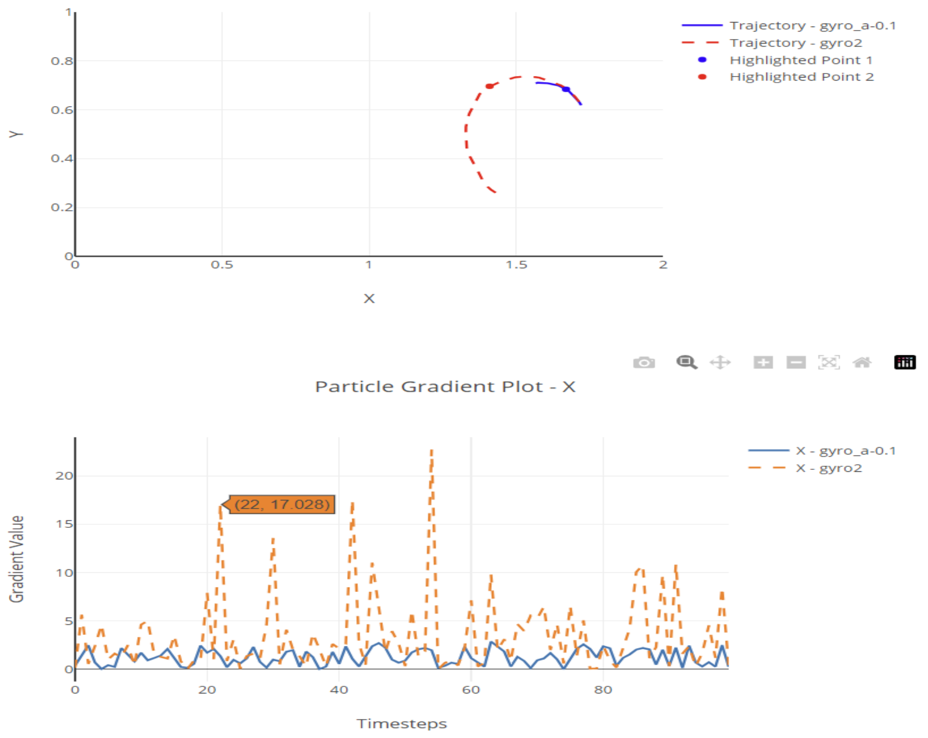
Visualization of particle trajectories is instrumental in comprehending the dynamics of fluid flow predicted by neural network models. By plotting the predicted paths of fluid particles over time, researchers gain valuable insights into how the model responds to different flow conditions.

By examining the particle trajectories alongside the vector field visualizations, researchers can observe how the model predicts particle movement relative to the flow patterns depicted by the vector fields. This juxtaposition enables the assessment of the model's ability to capture and simulate fluid dynamics accurately.

Furthermore, this integrated visualization approach also allows for a comparative analysis of predicted trajectories across different epochs of double gyre datasets with varying speeds. Figures 3.7a, 3.7b, and 3.7c showcase the visualization of particle trajectories using line plots, comparing ensemble datasets with Reynolds numbers of 200 and 4400, and double gyre 0.2 and 0.4 datasets at epochs 0 and 50, respectively. By observing how the



(a) Visualization of gradients using line plots. The plot compares gradient values for all inputs of a particle across different time steps.



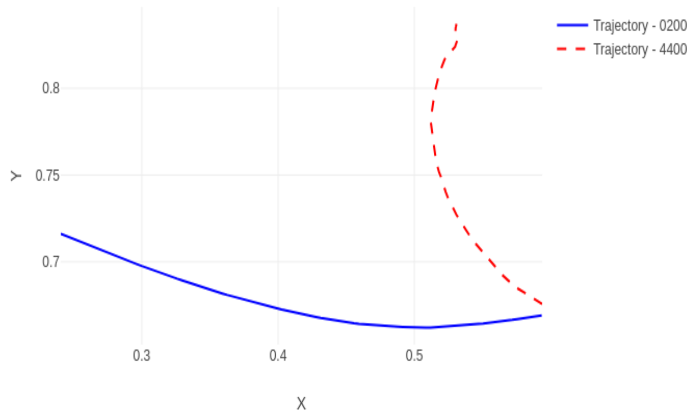
(b) Hover functionality demonstrating gradient values. The plot showcases how hovering over specific gradient values over a timestep highlights the corresponding particle position on the trajectory plot.

Fig. 3.6: Gradient Visualization using different inputs and associated functionality

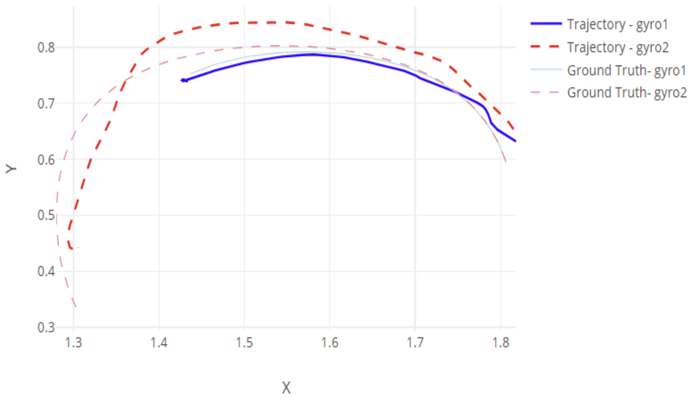
predicted trajectories change over training, researchers can gain insights into how the model learns and adapts to different flow scenarios, enhancing comprehension of its behavior and performance.

By visually inspecting particle trajectories and vector fields side by side, researchers can gain a holistic understanding of the model's behavior and performance in simulating fluid dynamics. This comprehensive analysis enhances comprehension by providing insights into how the model integrates various components of the prediction process and responds to different flow scenarios.

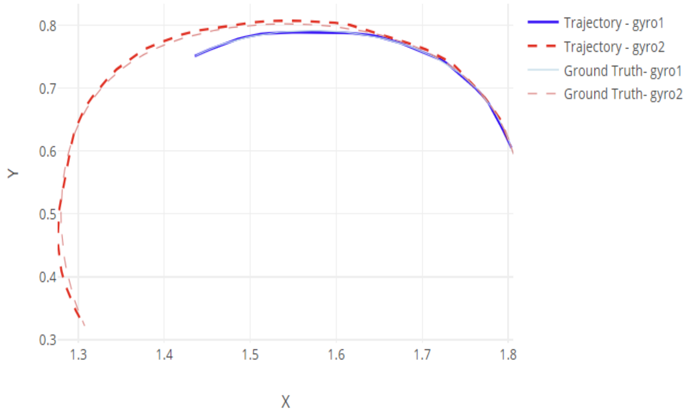
Overall, the visualization of particle trajectories facilitates a deeper understanding of neural network predictions in fluid dynamics simulations. By visually examining how particles move through the simulated flow field and comparing them with the underlying vector fields, researchers can identify patterns, anomalies, and areas for improvement in model predictions, ultimately enhancing the interpretability and reliability of the model.



(a) Comparison of particle trajectories in two different turbulence regimes: low (blue) and high (orange).



(b) Initial positions of particles with double gyro vector fields with speeds 0.2 and 0.4 are represented at epoch 0.



(c) Updated positions of particles after 50 epochs, with double gyro vector fields showing the change in direction and magnitude of particle movement. Speeds 0.2 and 0.4 are represented.

Fig. 3.7: Visualization of particle trajectories and dynamics

CHAPTER 4

Experimental study

4.0.1 Datasets

We utilize two main types of datasets in our experimental study:

- **Double Gyre Datasets**: These datasets provide simplified yet realistic scenarios for fluid dynamics simulations. The double gyre is a periodic time-dependent vector field in which a separating boundary oscillates horizontally between two oppositely rotating vortices. The flow was introduced by Shadden et al [18]. and became the prime benchmark for finite-time Lyapunov exponents. The analytic formula has several parameters that steer magnitude (A), oscillation frequency (omega), and oscillation amplitude (eps). The resampled versions use the standard parameters listed below. The flow is also spatially periodic, which allows variations that contain saddles in the interior of the domain, such as the quad gyre with [0,2] x [-1,1] x [0,10].

  Grid resolution (X x Y x T): 256 $\times$128 $\times$ 512

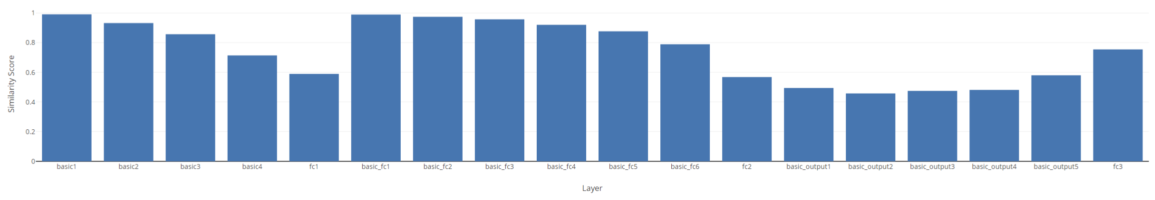
  $Space - timeDomain: \quad [0,2] \times [0,1] \times [0,10]$

  $Standardparameters: \quad A = 0.1, \quad \omega = \frac{\pi}{5}, \quad \varepsilon = 0.25$

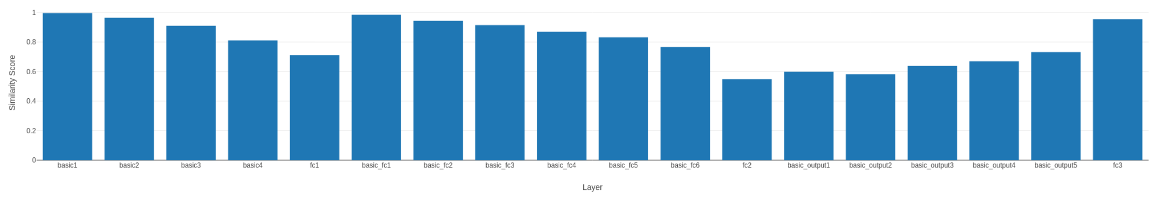
- **Ensemble Datasets**: Ensemble simulation of 8,000 unsteady 2D flows with spatially-periodic boundary conditions. The simulation was done with Gerris flow solver and is carried out on a regular grid. The 8,000 flows are ordered from laminar configurations to turbulent configurations. The last column and row are left out, since they are the same as the first column and row. A detailed description of the dataset can be found in Jakob et al [19].

  Regular grid resolution (X x Y x T): 512 $\times$512 $\times$ 1001

  $Simulationdomain: \quad [0,1] \times [0,1] \times [0,10]$



(a) Bar chart comparison of histogram intersection similarity for double gyro dataset between the default speed (a=0.2) and higher speed (a=0.4).



(b) Bar chart comparison of histogram intersection similarity for double gyro dataset at default speed (a=0.2) and one at lower speed (a=0.1).

Fig. 4.1: Comparison of histogram intersection similarity using bar charts for different Double Gyre datasets.

$$ReynoldsNumberrange: \quad [1, 4096]$$

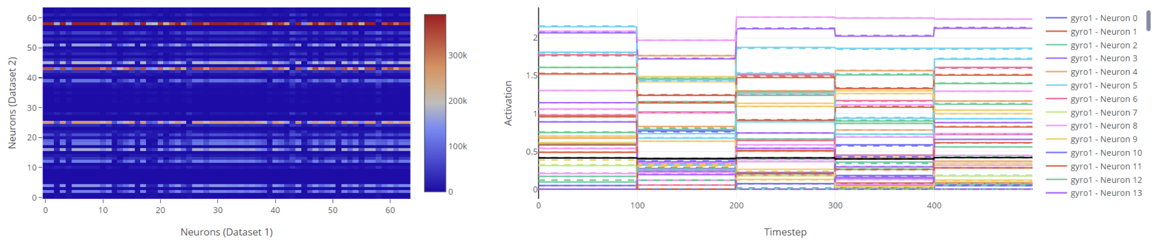
$$Kinematicviscosityrange: \quad [1 \times 10^{-5}, 1 \times 10^{-4}]$$

4.0.2 Results

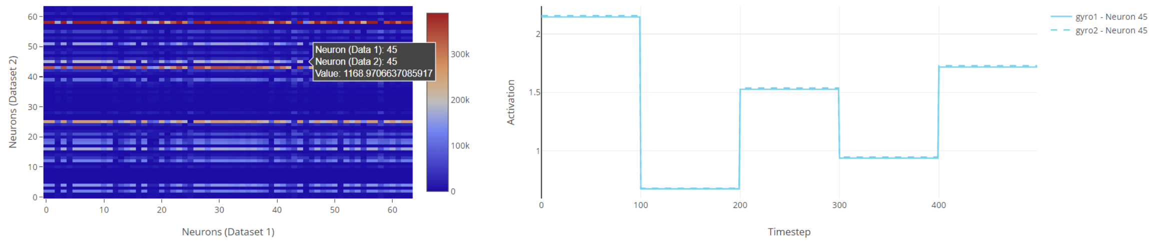
4.0.2.1 Double Gyre Simulation

This section presents the findings from a straightforward scenario utilizing the double gyre dataset. The objective is to illustrate the evolution of model predictions across various epochs. Through visualizations, we observe the model's progression in predicting particle trajectories during the training process, providing valuable insights into its learning dynamics.

Comparing All Layers and Activations: In Figure 4.1a, it can be observed that within the `basic1` layer, the similarity between the compared datasets is nearly 1. This high similarity can be attributed to employing identical seeding locations in our evaluation dataset. It is intriguing to note the model's ability to learn trajectories even when provided with



(a) Activation Similarity and Lines for Layer `basic1`



(b) Activation Similarity and Lines for Neuron 45 in Layer `basic1`

Fig. 4.2: Comparison of Activation Similarity and Neuron Lines between double gyre datasets with different vector fields at speed 0.2 and 0.4.

identical seeding locations, which highlights how the initial input layer consistently processes information.

Furthermore, the increased similarity observed in other input layers (`basic2` to `basic4`), which is greater than 0.5, can be ascribed to the lower complexity respect to the speed of the vector field within the double-gyre system . Similar results will be discussed further.

An interesting observation in this visualization arises when comparing different versions of the double gyre simulations where we vary the speed of the vector field. For example, when comparing the default double gyre simulation (0.2) with one with lower speed (0.1)4.1b, a high similarity is observed due to the closeness in the vector field of these datasets 4.3. This similarity tends to increase as the changes in the vector field increase.

To gain a more comprehensive perspective, Figure 4.2a provides a detailed view where all neuron lines in layer `basic1` are closely aligned. For a focused examination, individual neurons, such as neuron `45` in Figure 4.2b, can be considered.

Similarly, deeper insights into each layer's dynamics can be gained by visualizing them within our framework (Figure 4.4). This enables the examination of activations and weights, facilitating comparisons of similarities, activations across the entire batch, mean activation,

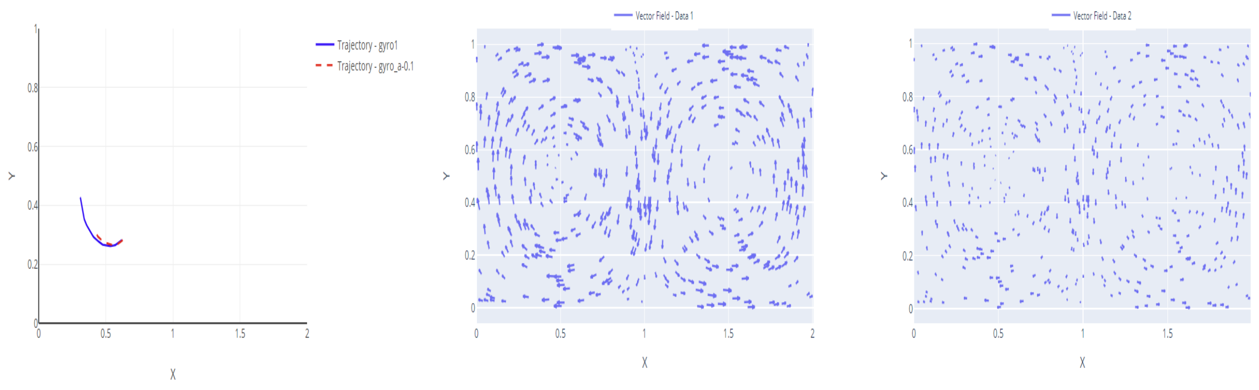


Fig. 4.3: Vector field and trajectories visualization depicting the double gyre system at speeds 0.2 and 0.1.

and individual weight plots.

A useful feature for gyro datasets in our framework is to visualize their gradients, vector field, and trajectories at different epochs throughout the model training. We can see the visualization system in Figure 4.5. Let's see how we can use all of these different components for better interpretability:

- The trajectory plots and vector field plots help us visualize where that particular particle is in the vector field and how the vector field is there. This helps us understand the motion of the particle across the whole vector field. We can see the same in Figure 4.6 where we look at different particles trajectories and the vector field that the network has learned (i.e., derived by predicted trajectories).

- It becomes even more useful when we look at the different particles' motion over the vector field as the training progresses. For example, we can see what the model learned about the vector field at that particular epoch (Figure 4.7) and how it made that prediction for a particle. The framework helps us understand the correction in vector field and trajectory prediction as well, which is very helpful.

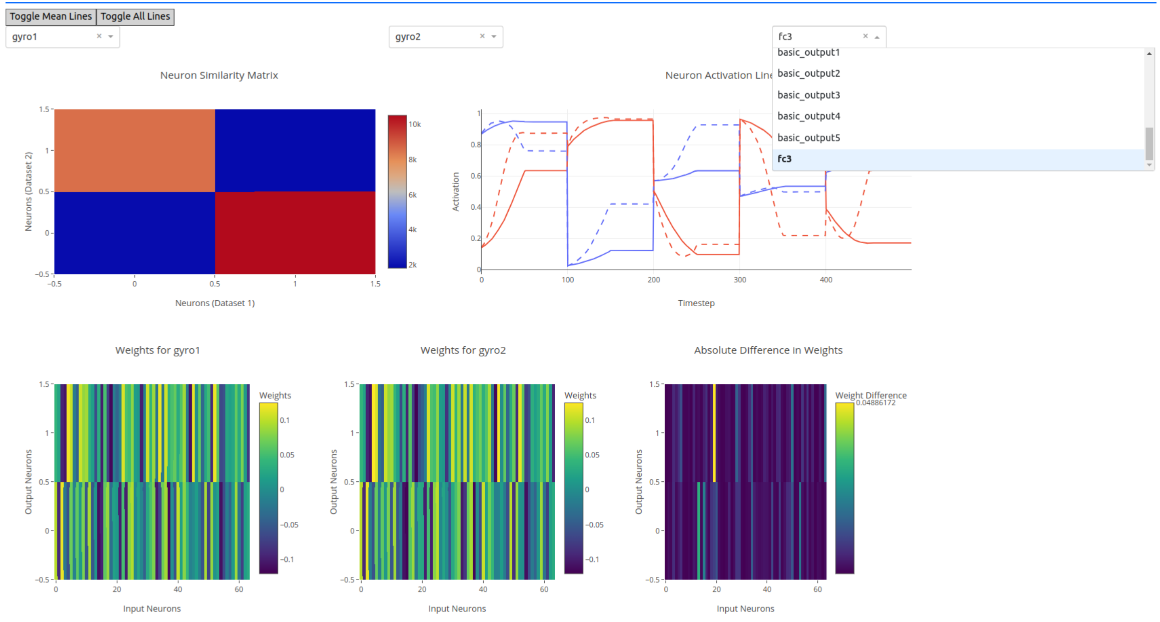


Fig. 4.4: Comparison of Double Gyre model for the layer fc3, including activations and weights

- Along with the trajectories and vector field, we can also see the gradients for inputs (coordinates x, y and timestep) changing over different epochs as seen in Figure 4.9.

  Magnitude of Gradient with Speed Increase: As the speed of the double gyre vector field increases, we observe a corresponding increase in the magnitude of gradients for all input features, including coordinates x and y as well as the timestep. This phenomenon can be attributed to the intensified dynamics within the vector field.

  For instance, as particles traverse through regions of the vector field characterized by higher speeds when comparing datasets at speeds of 0.2 and 0.4, or 0.2 and 0.1 respectively, the model may need to update its predictions more frequently to accurately capture their trajectories. This requires larger adjustments to the model parameters, leading to higher gradient values like in Figure 4.8.

  Conversely, when the vector field operates at lower speeds, particles move more slowly and predictably, resulting in smoother trajectories and smaller changes in the model's

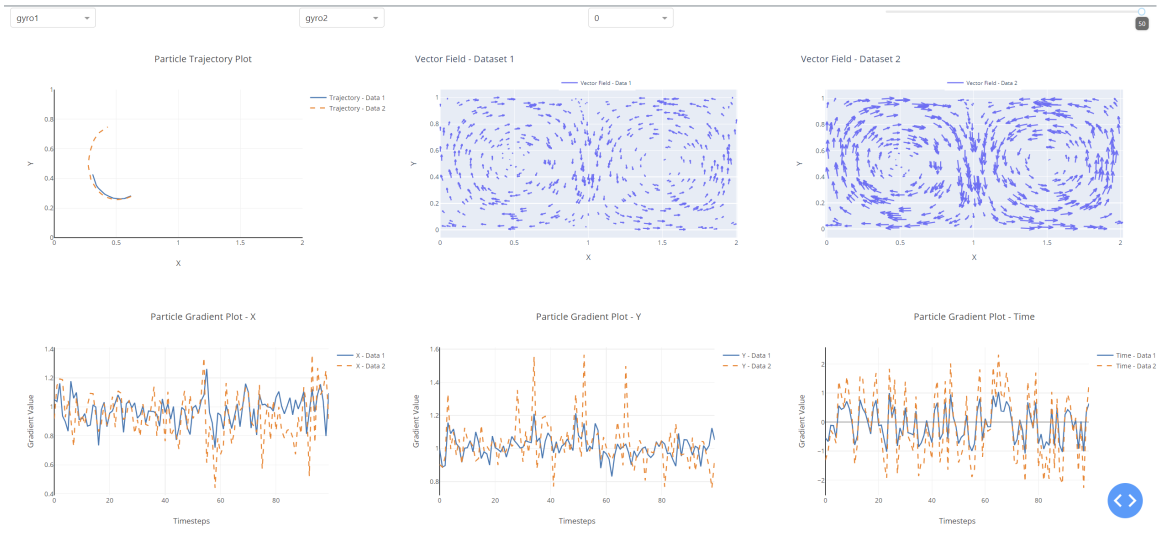
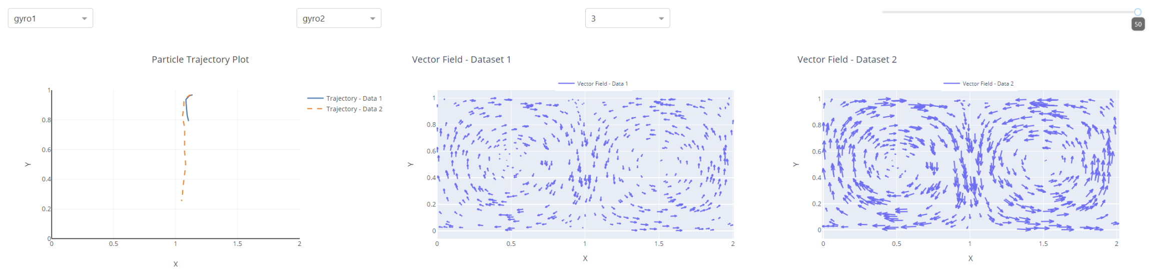


Fig. 4.5: An overview of our gradient, vector field, and trajectory plots for Double Gyre

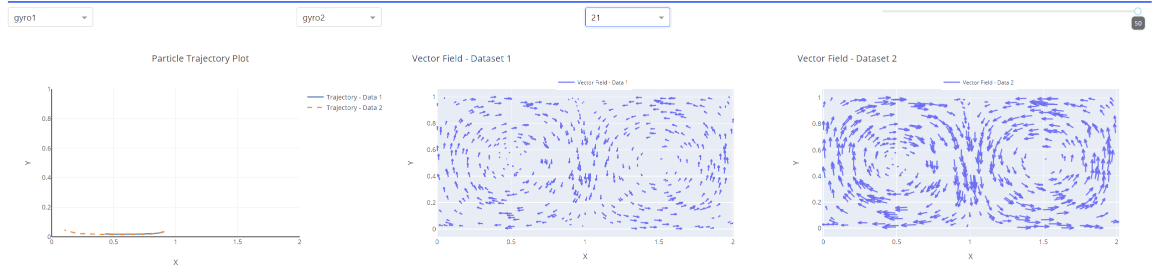
predictions over time. Consequently, the magnitude of gradients observed during training tends to be lower compared to scenarios with higher vector field speeds.

- The most important feature of this page is that when we click at a point in the gradient plots, we can see the position of that particle at that particular timestep highlighted. This helps us understand if anything in the vector field or particle motion caused this change. We can also use this feature to see the exact position of the particle in the vector field when seeing sudden highs and lows in the gradient plots like in Figure 4.10 we see a sudden peak in the gradients of input "y" at timestep 34, by clicking at that point we see that the particle two was having a change of trajectory in y direction and that's the reason the gradient adjusted its value to incorporate that change. This provides a very comprehensive understanding of the inner working of our neural network model.

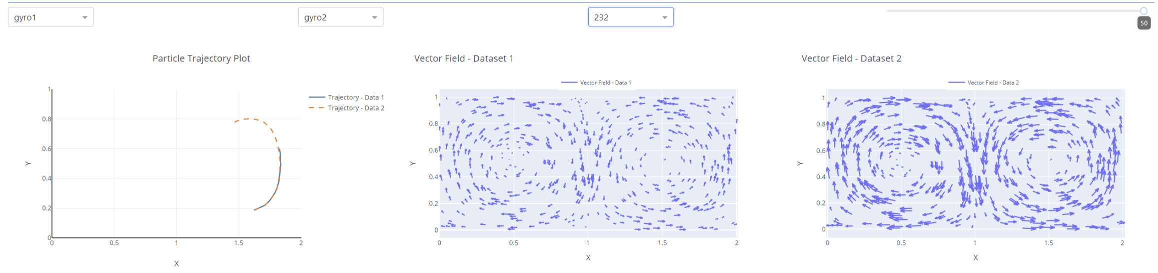
Functionality of Picking Peaks in Gradient and Correlating with Changes in Direction: In Figure 4.9, we visualize the gradients for inputs (coordinates x and y, and timestep) changing over different epochs. For example, if we observe a sudden peak in the gradient at a particular timestep like at 22 as seen in Figure 3.6b when we are



(a) Trajectory and vector field for Double Gyre Datasets and particle 3.



(b) Trajectory and vector field for Double Gyre Datasets and particle 21.

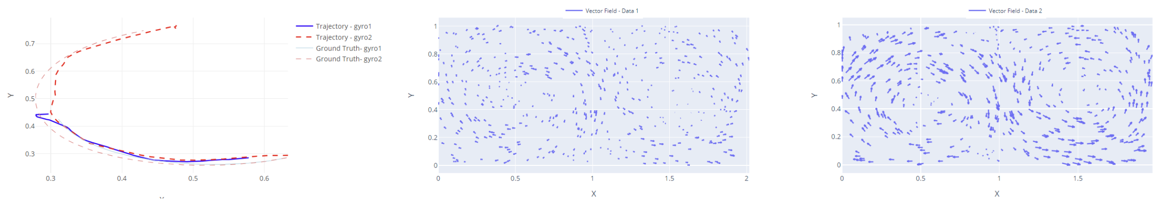


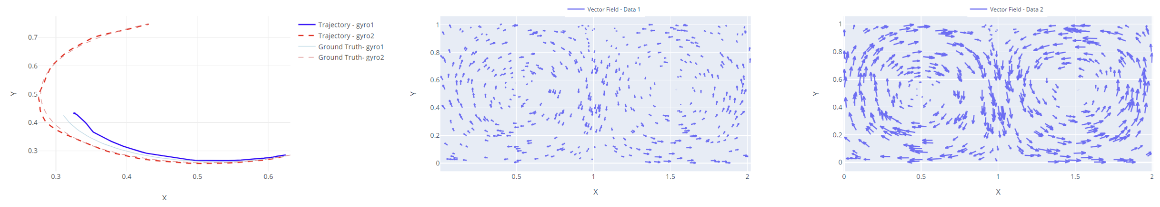
(c) Trajectory and vector field for Double Gyre Datasets and particle 232.

Fig. 4.6: Visualization of Different Particles in the vector field for Double Gyre Datasets.

comparing double gyre with speed 0.1 and 0.4, we can use the interactive feature to identify the corresponding position of particles in the vector field at that moment. By analyzing the direction of particle movement at this point, we can infer the underlying cause of the gradient peak.

The gradient peak coincides with a significant change in particle direction, such as a sudden shift from x to y rotation within the vector field, we can conclude that the model has detected this change and adjusted its predictions accordingly. This insight helps us understand how the model adapts to variations in the vector field dynamics and improves the interpretability of its predictions.


(a) Epoch 0.


(b) Epoch 10.

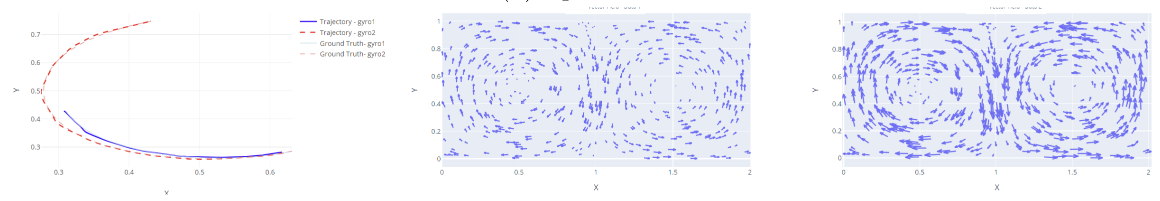

(c) Epoch 25.

Fig. 4.7: Trajectory of Particle 3 with changing vector field for both Double Gyre datasets over changing epochs.

4.0.3 Comparison of Ensemble Datasets

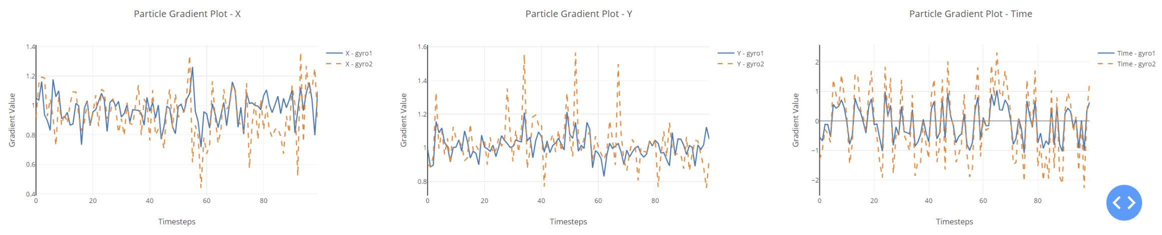
4.0.3.1 Comparing all layers

When comparing all the layers using Histogram Intersection Similarity of ensemble with different Reynolds numbers and Transfer Learning Dataset, we observe the following similarity values as shown in Figure 4.11:

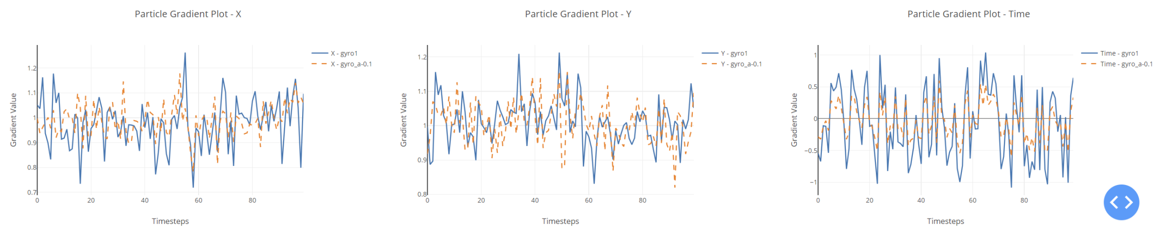
Let's analyze these similarity values and their implications:

1. **Decrease in Similarity in Early Layers (basic1-fc1):**

    - The similarity values notably decrease in the input layers as we compare Reynolds numbers (RE) 200 with models of increased Reynolds Number. While there is still relatively good similarity when compared to RE=3200 in layer `basic1` (see Figure 4.11a), as we progress to `basic4`, we observe a drop in similarity scores. This decrease is attributed to these layers learning more complex features. The



(a) Comparison of gradients: Velocity 0.2 vs Velocity 0.4.



(b) Comparison of gradients: Velocity 0.2 vs Velocity 0.1.

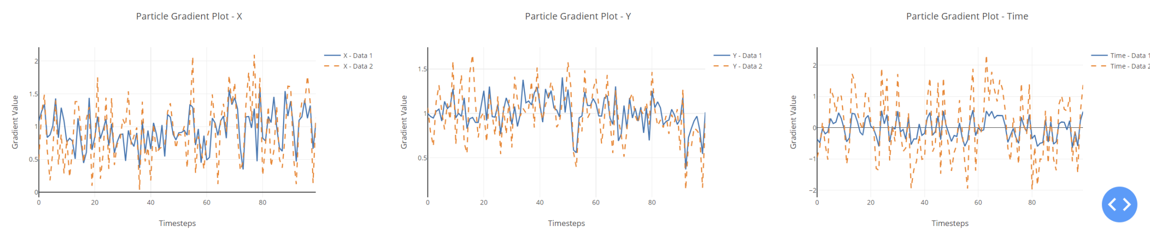
Fig. 4.8: Comparison of gradients in double gyre datasets at varying speeds. The gradient values increase as the speed of the vector field increases, as illustrated by the comparison between different velocity settings.

`start` input undergoes a series of transformations through basic fully connected layers (`basic1` to `fc1`), extracting relevant features related to the particle's position in the fluid simulation.

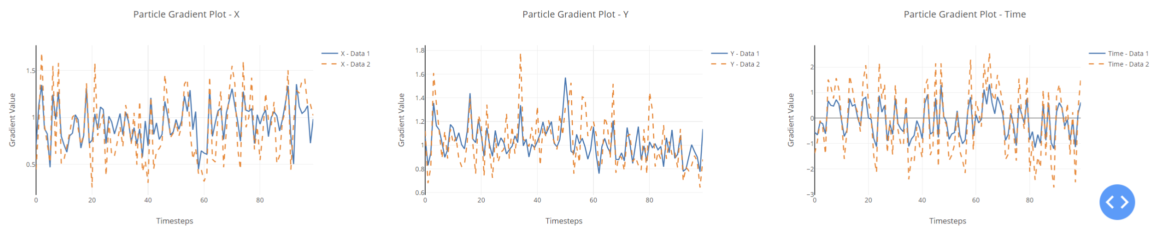
- This decrease in similarity indicates that these layers are more sensitive to changes in Reynolds numbers or the fluid vector field, as depicted in Figure 4.11d.
- We observe a general low similarity score in layer `fc1` as spatial information is concatenated here from the previous layers, and any change in the fluid flow resulting from the change in Reynolds number leads to reduced similarity. Similar trends are visible in Figures 4.11a, 4.11b, 4.11c, and 4.11d.

2. **Consistent Similarity in Intermediate Layers (basic_fc1-fc2):**

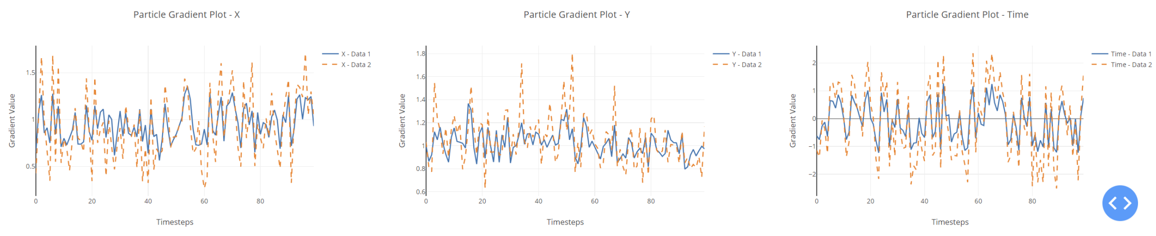
- The consistently high similarity values (over 0.5) across these layers (**basic_fc1-basic_fc6**) 4.11 suggest that they encode temporal dynamics and complex relationships that are less influenced by variations in Reynolds numbers.



(a) Epoch 0.



(b) Epoch 10.



(c) Epoch 25.

Fig. 4.9: Gradients of Particle 3 changing for input coordinate X, Y, and time for both Double Gyre datasets over changing epochs.

- They are likely learning representations that are generalizable across different flow conditions, contributing to the model's robustness.
- One interesting thing to note is the drop in similarity at the `fc2` layer. The activations from both the basic and fully connected layers are concatenated to capture joint representations of spatial and temporal features here.
- `fc2` is responsible for transforming the abstract representations from the previous layers into a format that is suitable for predicting particle trajectories.
- The variations in similarity at this layer indicate differences in the learned representations between datasets, possibly due to the sensitivity of these representations to changes in flow conditions.

3. **Variability in Output Layers (basic_output1-fc3):**

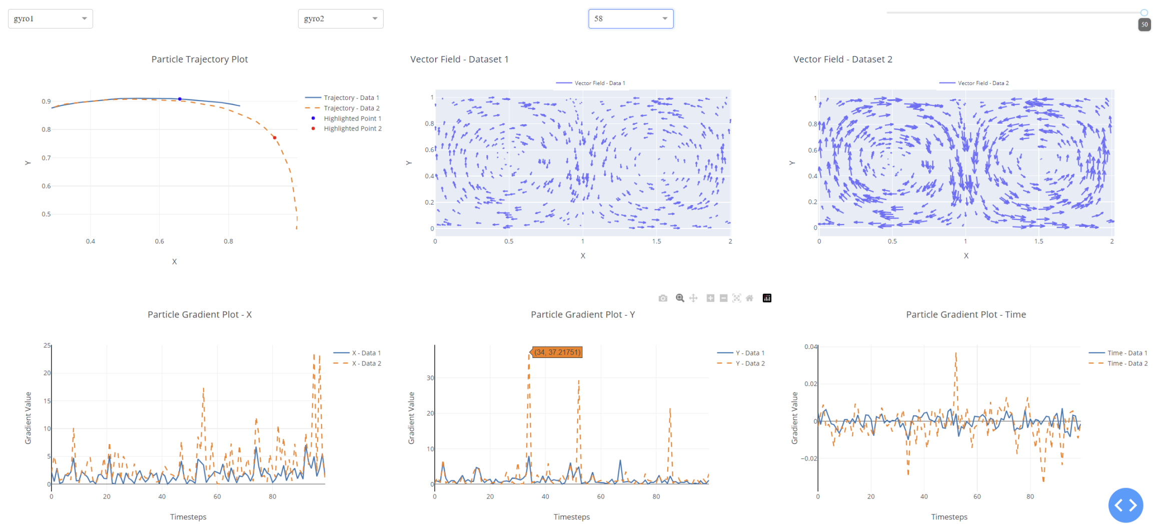
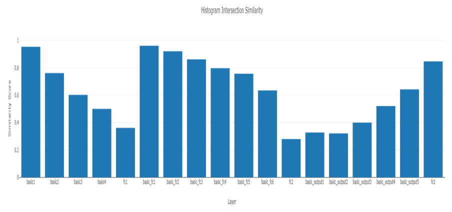


Fig. 4.10: Highlighting timestep 34 using "Particle Gradient Plot- Y" in the trajectories plot of Double Gyre to Understand the change in trajectories and gradients

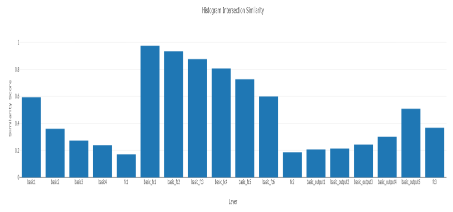
- The output layers decode the learned representations to predict particle trajectories. This architecture is designed to effectively model the complex relationships between particle behaviors and fluid flow conditions in simulation datasets based on the learned representations from earlier layers.
- Contrary to the trend observed in the early layers, we notice an increase in similarity values across the output layers.
- The final fully connected layer (`fc3`) produces the predicted end location of particles. The differences in similarity at this layer suggest variations. When encoding information in the earlier layers of the model from spatial data, if there isn't a big change in the fluid simulation of two datasets, we can get a higher similarity 4.11a. But as it gets more complex, we can see the same change reflected in the `fc3` as well because it provides us with the predictions learned from that `start` point 4.11d.

4.0.3.2 Comparison of Activations across Individual Layers and Neurons

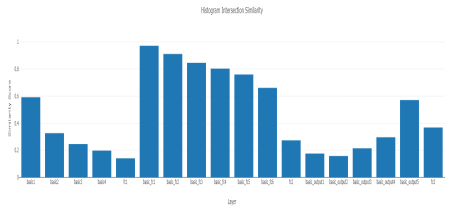
To validate our analysis regarding the comparison of all layers using histogram intersection (see Section 4.0.3.1), we will delve into the activation of individual layers and



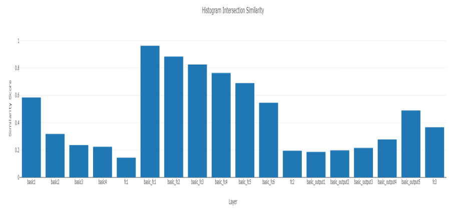
(a) Histogram Intersection Similarity of ensemble datasets with RE=200 and RE=3200.



(b) Histogram Intersection Similarity of ensemble datasets with RE=200 and RE=4400.



(c) Histogram Intersection Similarity of ensemble datasets with RE=200 and RE=4400 Transfer by 200.



(d) Histogram Intersection Similarity of ensemble datasets with RE=200 and RE=4400 Transfer by 7400.

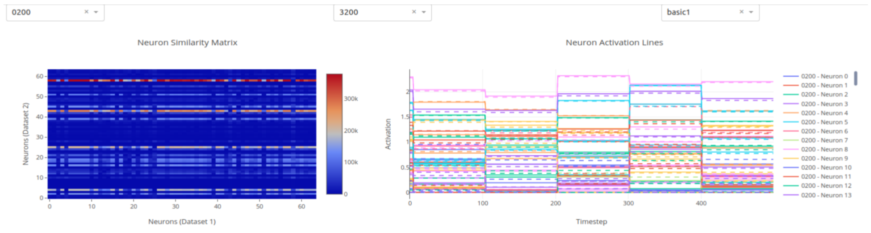
Fig. 4.11: Particle trajectory visualization

their neurons. This deeper investigation will provide additional insights and confirm the information gleaned from the histogram similarity analysis.

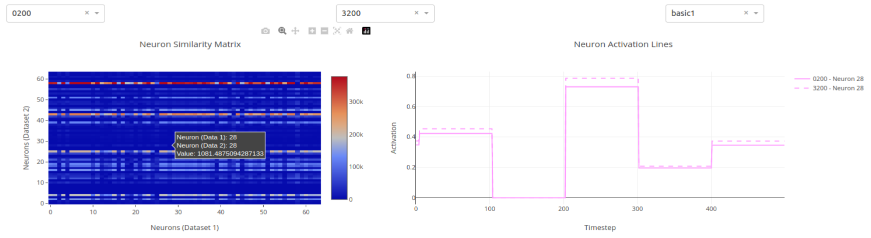
Examining the activation patterns of individual layers and neurons allows us to understand how information is processed and transformed throughout the network. By analyzing the activations, we can observe how different layers respond to variations in input data, particularly changes in Reynolds numbers or fluid flow conditions. This granular analysis will help us corroborate our findings from the histogram intersection similarity analysis and provide a more nuanced understanding of the model's behavior.

Let's start by investigating why we observed a decrease in similarity in the input layers (see Section 1). Additionally, we will explore why there was a peculiar higher similarity in layer `basic1` when comparing Reynolds numbers of 200 and 3200.

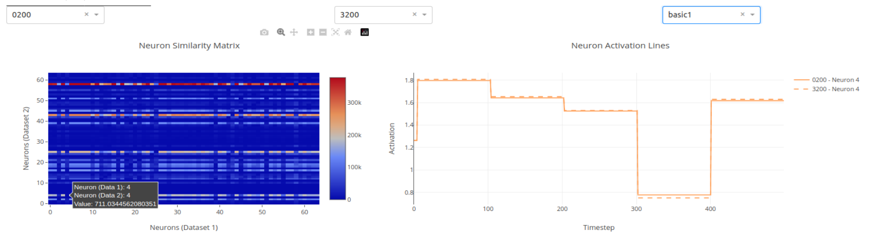
- Figure 4.12 depicts the activation similarity for the layer `basic1` for all the neurons in Figure 4.12a. We can clearly see that the activation values for neurons across both datasets are very similar, which is the reason we observed an increased histogram



(a) Activation similarity and lines for layer `basic1`.

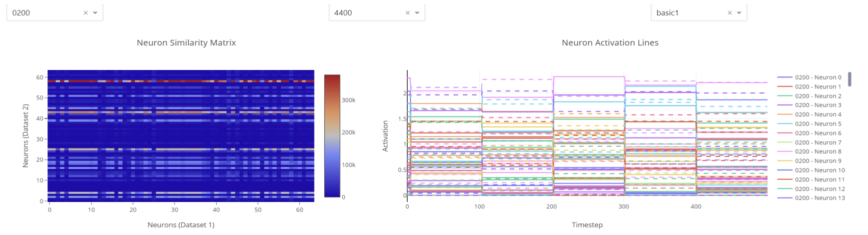


(b) Activation similarity and lines for neuron 28 in layer `basic1`.

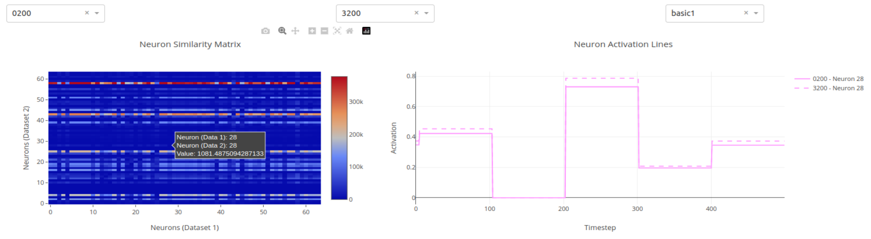


(c) Activation similarity and lines for neuron 4 in layer `basic1`.

Fig. 4.12: Activation similarity and Neuron Lines comparison between RE=200 and RE=3200

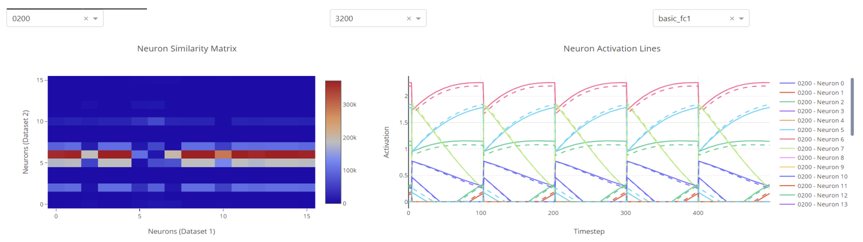


(a) Activation similarity and lines for layer `basic1`.

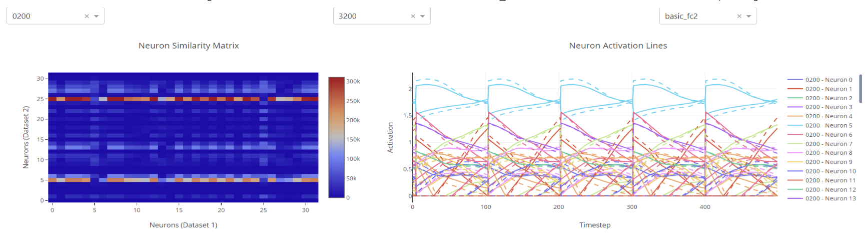


(b) Activation similarity and lines for neuron 4 in layer `basic1`.

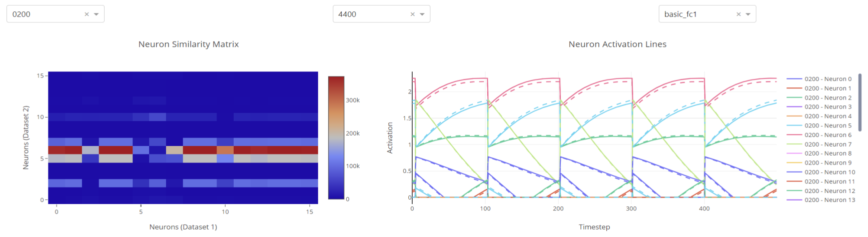
Fig. 4.13: Activation similarity and Neuron Lines comparison between RE=200 and RE=4400



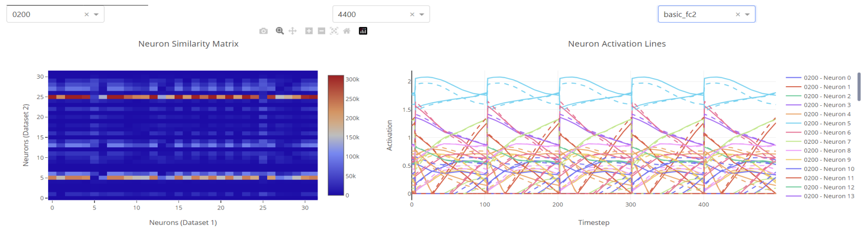
(a) Activation similarity and lines for when compared to RE=3200, layer `basic_fc1`.



(b) Activation similarity and lines for when compared to RE=3200, layer `basic_fc2`.

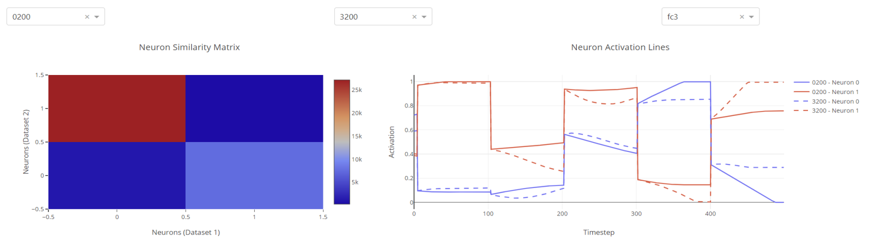


(c) Activation similarity and lines for when compared to RE=4400, layer `basic_fc1`.

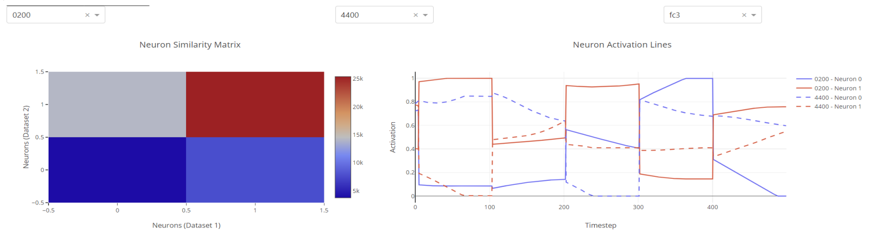


(d) Activation similarity and lines for when compared to RE=4400, layer `basic_fc2`.

Fig. 4.14: Activation similarity and Neuron Lines comparison between RE=200 and other datasets



(a) Activation similarity and lines when compared to RE=3200, for layer `fc3`.



(b) Activation similarity and lines when compared to RE=4400, for layer `fc3`.

Fig. 4.15: Activation similarity and Neuron Lines comparison between RE=200 other datasets

similarity when comparing the entire models in Section 1.

- For better interpretability, we also visualized the activation of neuron 4 in Figure 4.12c and neuron 28 in Figure 4.12b as examples to closely examine and understand how these activation values relate and why we observed a high histogram similarity for particularly these two datasets. The variations in activation values of both of these neurons are minimal, validating our results in Section 1.

- In Figure 4.13, we illustrate how the difference in Reynolds number decreases the similarity between given layers. We can see the same happening for all the neurons in Figure 4.13a, where the activation values are different for both datasets. Similarly, Figure 4.13b demonstrates that when examining a single neuron, the activation values differ significantly, resulting in lower similarity for the histogram as well in Section 1.

- As described in section 2 we can see that in 4.14 for different input layers `basic_fc1` and `basic_fc2` for two different datasets we see how the activations values are very similar even when changing the layers and datasets. This shows the consistency in results with histogram similarity and helps us understand the model working and which layers or inputs hold more importance.

- For the final section 3 we saw the changes in the `fc3` layer. To analyse the same we can look at the Figure 4.15 which tells us the change in the activations values across datasets. Also, this is the layer which provides us the final output the reason we have only two neurons, so both the disparity in activation values and the lower number of neurons in the layer results in a lower histogram similarity as well for this layer

This comprehensive examination of individual layer activations and neurons will complement our previous analysis and provide additional evidence to support our conclusions regarding the comparison of ensemble datasets. It will strengthen the validity of our findings and enhance the overall robustness of our investigation into the behavior of MLP networks in fluid simulation ensembles.

Overall, the experimental study demonstrates the efficacy of the visualization framework in enhancing the interpretability of MLP networks for particle tracing in fluid simulation ensembles. The results provide valuable insights into model behavior and performance across various scenarios, facilitating informed decision-making in scientific applications.

CHAPTER 5
Conclusion

In this thesis, we developed a comprehensive framework for visualizing and interpreting a neural network model for particle trajectory prediction in fluid dynamics simulation data. Our framework encompasses a range of visualization techniques tailored specifically for analyzing neural network behavior in fluid dynamics applications. These techniques include visualizations of model activations, trajectories, gradients, and other relevant metrics, providing a multifaceted approach to understanding the inner workings of the model.

By leveraging visualization techniques and in-depth analysis of model activations, trajectories, and gradients, we gained valuable insights into the learning dynamics and predictive capabilities of these models. Our investigation focused on understanding how neural networks process information related to particle trajectories in fluid flow simulations and how they generalize across different flow conditions.

First, we explored the performance of our model on a simple scenario using the double gyre dataset. Through visualizations and quantitative analysis, we observed the model's ability to learn particle trajectories and its sensitivity to variations in vector fields. Despite the simplicity of the scenario, our framework provided some insights into the model's learning process and highlighted how gradients change according to particle trajectories and their speed.

Furthermore, we conducted a comprehensive comparison of ensemble datasets with varying Reynolds numbers. Through histogram intersection similarity analysis and examination of individual layer activations, we uncovered intriguing trends in how the model processes information at different stages of the neural network. We observed a decrease in similarity in the early layers, indicating sensitivity to changes in flow conditions, while intermediate layers showed consistent performance across different datasets. The variability in output layers highlighted the model's ability to capture complex relationships between particle behaviors and fluid flow conditions.

Our findings contribute to the understanding of neural network models' performance in fluid dynamics simulations and provide valuable insights for their application in real-world scenarios. By combining advanced visualization techniques with quantitative analysis, we were able to gain a deeper understanding of the inner workings of these models and their response to varying input conditions.

Moving forward, our framework can be extended to analyze more complex fluid dynamics simulations and explore the applicability of neural network models in predicting particle trajectories in diverse conditions. Additionally, integrating techniques from explainable AI and uncertainty quantification can further enhance the interpretability and reliability of our models, making them valuable tools for understanding and predicting fluid flow phenomena. Overall, our research lays the foundation for future advancements in the interpretability of neural networks for scientific visualization, with implications for a wide range of applications.equation

REFERENCES

[1] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2014.

[2] M. Han, S. Sane, and C. R. Johnson, "Exploratory lagrangian-based particle tracing using deep learning," *Journal of Flow Visualization and Image Processing*, vol. 29, no. 3, pp. 73–96, 2022.

[3] C. Wang and J. Han, "Dl4scivis: A state-of-the-art survey on deep learning for scientific visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 8, pp. 3714–3733, 2023.

[4] C. Molnar, G. Casalicchio, and B. Bischl, *Interpretable Machine Learning – A Brief History, State-of-the-Art and Challenges.* Springer International Publishing, 2020, p. 417–431. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-65965-3_28

[5] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, D. Pedreschi, and F. Giannotti, "A survey of methods for explaining black box models," 2018.

[6] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," 2013.

[7] M. Zeiler, G. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *2011 International Conference on Computer Vision, ICCV 2011*, ser. Proceedings of the IEEE International Conference on Computer Vision, 2011, pp. 2018–2025, 2011 IEEE International Conference on Computer Vision, ICCV 2011 ; Conference date: 06-11-2011 Through 13-11-2011.

[8] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, p. 336–359, Oct. 2019. [Online]. Available: http://dx.doi.org/10.1007/s11263-019-01228-7

[9] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization," *Distill*, 2017, https://distill.pub/2017/feature-visualization.

[10] X. Chen, Q. Guan, X. Liang, L.-T. Lo, S. Su, T. Estrada, and J. P. Ahrens, "Tensorview: visualizing the training of convolutional neural network using paraview," *Proceedings of the 1st Workshop on Distributed Infrastructures for Deep Learning*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:20084419

[11] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016.

[12] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines vinod nair," vol. 27, 06 2010, pp. 807–814.

[13] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[15] A. Bicciato, L. Cosmo, G. Minello, L. Rossi, and A. Torsello, "Gnn-lofi: A novel graph neural network through localized feature-based histogram intersection," *Pattern Recognition*, vol. 148, p. 110210, Apr. 2024. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2023.110210

[16] L. Wang, L. Hu, J. Gu, Y. Wu, Z. Hu, K. He, and J. Hopcroft, "Towards understanding learning representations: To what extent do different neural networks learn the same representation," 2018.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html

[18] S. C. Shadden, F. Lekien, and J. E. Marsden, "Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows," *Physica D: Nonlinear Phenomena*, vol. 212, no. 3, pp. 271–304, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167278905004446

[19] J. Jakob, M. Gross, and T. Günther, "A fluid flow data set for machine learning and its application to neural flow map interpolation," *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization 2020)*, 2021.