8-2023

Generalizing Deep Learning Methods for Particle Tracing Using Transfer Learning

Shubham Gupta
Utah State University

GENERALIZING DEEP LEARNING METHODS FOR PARTICLE TRACING USING TRANSFER LEARNING

by

Shubham Gupta

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	John Edwards, Ph.D. Committee Member
Shuhan Yuan, Ph.D. Committee Member	D. Richard Cutler, Ph.D. Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2023

ABSTRACT

Generalizing Deep Learning Methods for Particle Tracing using Transfer Learning

by

Shubham Gupta, Master of Science

Utah State University, 2023

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

Visualization of vector fields is very important for understanding flow behaviour in different domains of science and engineering. Particle tracing is one of the most popular methods to simulate particle transport, reveal flow patterns, and understand the behavior of the flow-field around critical points. However, this technique requires expensive numerical integration in space and time. The use of AI and particularly deep learning based techniques have shown to be able to speed up this process and provide accurate results. However, such models have limited practical use, as they perform well only on the specific use case they are trained for. In this work we present a methodology to generalize the use of deep learning for particle tracing using transfer learning. We demonstrate the performance of our approach through a series of experimental studies that address the most common simulation design scenarios: varying time span, Reynolds number, and problem geometry. Our results show that transfer learning can be effectively used to generalize and accelerate the training and practical adoption of deep learning models for visualization of unsteady flows.

(40 pages)

PUBLIC ABSTRACT

Generalizing Deep Learning Methods for Particle Tracing using Transfer Learning

Shubham Gupta

Particle tracing is a very important method for scientific visualization of vector fields, but it is computationally expensive. Deep learning can be used to speed up particle tracing, but existing deep learning models are domain-specific. In this work, we present a methodology to generalize the use of deep learning for particle tracing using transfer learning. We demonstrate the performance of our approach through a series of experimental studies that address the most common simulation design scenarios: varying time span, Reynolds number, and problem geometry. The results show that our methodology can be effectively used to generalize and accelerate the training and practical use of deep learning models for visualization of unsteady flows.

sreyan sva-dharmo vigunah
para-dharmat sv-anusthitat
sva-dharme nidhanam sreyah
para-dharmo bhayavahah
- Krishna, Bhagavad Gita

”It’s better to commit mistakes on the path that one’s soul is meant to walk on, than to live a perfect life on a path that is not meant for one’s soul.”

ACKNOWLEDGMENTS

First and foremost, I am deeply grateful to my major professor, Dr. Steve Petruzza, for his unwavering support and mentorship. Dr. Petruzza's expertise, insightful feedback, and dedication to academic excellence have been crucial in shaping the direction of this research. I am grateful for the time he invested in guiding me through the complexities of the thesis and for instilling in me a passion for continuous learning. I would also like to express my appreciation to the committee members who provided valuable insights and constructive feedback during the thesis defense.

I want to express my heartfelt gratitude to my dear mother, whose unconditional love, belief, and encouragement have been my guiding light. She has been my constant pillar of strength, always pushing me to pursue my dreams and excel in everything I do. I owe my success to her unwavering support and sacrifices. To my sister, thank you for being my confidante and cheering me on during the most challenging times. My heartfelt gratitude extends to my father, my uncle for making me tough, and Bua for taking care of me, your love and blessings have been an essential part of my life's journey. I am grateful for their continuous support.

I am indebted to my friends, who have been an integral part of this academic endeavor. Anjali, your patience and unwavering belief in me have been a source of strength. Your presence made this journey more enjoyable. Kartik, thank you for always being there whenever I needed a helping hand. Arun, I appreciate your friendship. Raushan, your guidance and mentorship throughout this thesis have been invaluable. Most importantly, Sarvesh and Anupreet for being my backbone in life. Avnish for pushing me to achieve greater things.

To everyone who has contributed to my growth and success, thank you from the bottom of my heart. Each of you has played a significant role in shaping me into the person I am today. I am truly blessed to have such wonderful people in my life.

Shubham Gupta

CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

Flow visualization is an essential tool for understanding the complex behavior of fluid dynamics in a wide range of scientific and engineering applications, from improving the performance of aircraft and automotive designs to developing new medical treatments.

Practical use of AI and particularly deep learning in scientific visualization of fluid-flow is difficult due to the lack of available data that can be used for training. This is exacerbated by the fact that each new simulation is different, based on the computational parameter, boundary conditions or the geometry. Fluid flow engineers and scientists design their simulations to quantify the effect of different variables pertinent to the phenomena under investigation. These variable include change in the simulation parameters, geometry and turbulence conditions. Often a phenomena is simulated across different governing parameters, like the viscosity of the fluid and flow speed, which contribute to the change in the non-dimensional simulation parameter Reynolds number (*Re*). *Re* quantifies how turbulent the flow would be in the simulated domain, thus it is one of the most common parameter that is varied while conducting different simulations for the same phenomena [18]. In the same vein, often simulations are conducted by making small changes in the geometry, in order to quantify the effect of geometrical changes [18] and to generate optimal designs [17]. These changes in flow parameters and geometry often result in substantial difference in the flow-field across the simulations. Thus, an AI based model for flow visualization that is trained using data from a specific simulation case, may not work for another case. It is this question of model *generalization*, we will be exploring in this paper.

One of the methods widely used for visualizing the flow is the computation of pathlines, trajectories of massless particles advected by the flow, which are generated by using Lagrangian particle tracing [7]. In this context, existing deep learning models require long training time and can be only used effectively on a particular simulation and a particular time span that the model was trained on. In Figure 1.1, we present three cases in which a Neural Network (NN) based Lagrangian tracing

model [12] trained on different time-span (Figure 1.1a), different Reynolds number (Figure 1.1b), and different geometry (Figure 1.1c) was used to predict a new simulation. The results show that there is not enough knowledge in the model that can be effectively used across the various flow simulations, posing serious limitations to models generalization. In this work we tackle those challenges providing a methodology to generalize deep learning based particle tracing across different simulation settings.

Insights from our experimental studies demonstrates that: (i) the use of multiple neural network models can improve the accuracy of predicting particle trajectories over long time spans; (ii) models trained with high turbulent flows can be effectively reused (with short incremental training) to accurately predict particle trajectories on flows with different flow structures, induced by change in Reynolds number (Re). Similar behaviours are also observed when tracing particles over different time spans and when small changes in the geometry of the simulation occur. Those findings can dramatically reduce the training time required to produce a new model for each different simulation scenario.

In summary, the contributions of this paper are:

- A methodology to use deep learning models to predict particle trajectories over long time series;
- A methodology to generalize deep learning models for particle tracing using transfer learning;
- An experimental study to test the proposed methodologies in three common simulation scenarios: long and varying time spans, varying Reynolds number, and varying geometry.

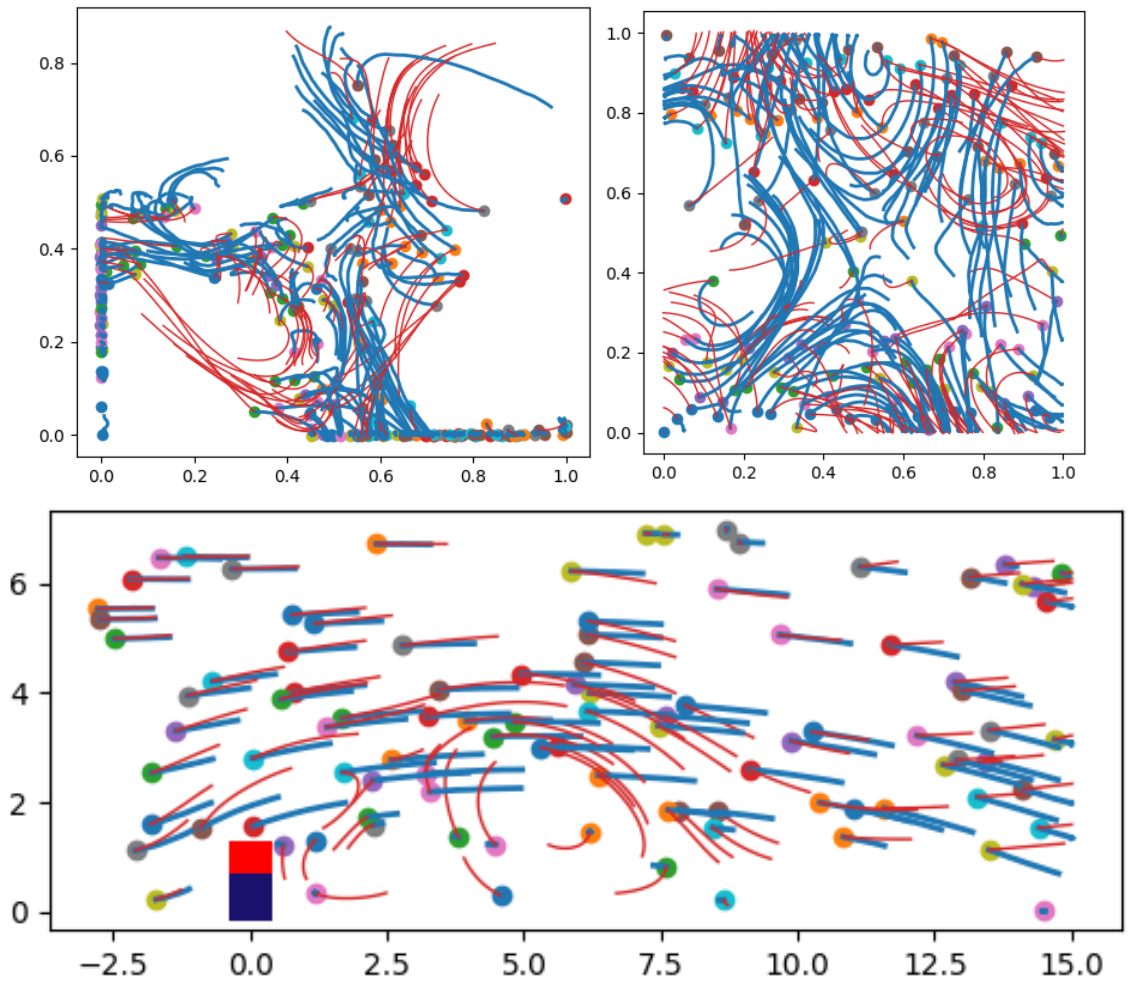

Fig. 1.1: Particle trajectories computed by a neural network model [12] on a different simulation scenario. Red lines are ground truth, blue lines are predictions. Filled circles are seeding locations. On the top left (a), the model was trained on a simulation time span different from what was used for testing (Training for time 0-100 vs testing for time 101-200). On the top right (b), the model was trained on a simulation with a different *Re* number (training at 48.5 vs testing at 147.0). On the bottom (c), the model was trained on a simulation of a flow behind a rectangular obstacle of equal height and width (blue box) and used to predict Lagrangian tracing trajectories for a higher obstacle of height 1.5x the width (red box). The results clearly show that neural network models are not directly portable to different scenarios from those they were trained on.

CHAPTER 2

Background

Deep learning techniques have been increasingly explored for scientific visualization [16, 25]. More specifically, in the area of flow visualization, deep learning methods have been employed to improve data access patterns of parallel particle tracing [14], produce higher-resolution versions of vector fields [10], identify vortices [6], and select streamlines for effective flow visualization [11]. The Lagrangian-based representation of time-varying vector fields [1, 2, 23], which has demonstrated improved accuracy-storage propositions and reduce errors from temporal subsampling [4], has garnered increased attention for use in deep learning approaches. Previous work by Han et al. [12] provided a model for predicting particle trajectories in Lagrangian-based flow fields using a multi-layer perceptron model (MLP). Sahoo et al. [22] employed an implicit neural network to learn the integration scheme of vector fields using Lagrangian-based flow maps. Such models have demonstrated high accuracy but still require lengthy training times and extensive training data. Ensemble simulations present even greater challenges for these models. In this context, deep surrogate models have been employed to improve flow map interpolations [15] and enable parameter space exploration for in situ visualization [13].

This challenge of *Model Generalization* is also identified by Wang et al. as the main issue in their recent survey [25]. Wang and Han noted the limitations imposed by the current practice of "one training for one dataset" and the lack of research into how the Neural Network modeling framework could be adapted for *transfer* learning across variables and ensembles. Transfer learning [28] has proven to be a promising technique for generalizing knowledge across domains [21]. For example, the use of convolutional neural networks (CNN) from a pre-trained fine-tuned model for lymph node detection [24]. Other studies have successfully used transfer learning for speech and language processing [26], identification of plants [9], hyper-spectral superresolution [27], remote sensing [3]. However, no existing studies investigate how to apply transfer learning to ensemble simulations for particle tracing. In this work, we apply the transfer learning methodology to generalize particle

tracing using neural networks across different flow simulations, aiming to overcome current model development limitations for scientific visualization (as those reported in Figure 1.1).

CHAPTER 3

Modeling framework

Our modeling framework aims to use transfer learning as a tool to facilitate the reuse of models across ensemble simulations. In particular, we focus on the most common variants in flow ensemble simulations: variation over time spans, turbulence conditions, and changes in geometry. In Figure 3.1, we report a diagram of a traditional workflow (on top), which trains a different model from scratch for every simulation variant, and the workflow of our generalized approach using transfer learning. This approach allows to dramatically reduce the training time of models across several kind of simulation variants.

For this study the computation of particle trajectories is considered to happen in situ where we have full spatial and temporal resolution available to compute Lagrangian flow maps. We use the $Lagrangian_{long}$ method [12] which extracts a single flow map with long particle trajectories using uniform temporal sampling and a Sobol quasirandom sequence as seeding strategy.

In Figure 3.2 we report the model architecture from Han et. al [12] that was employed in this work. This architecture implements an auto-encoder architecture consisting of three main components: a positional encoder, a timestamp encoder, and a decoder (see Figure 3.2). Each of these components utilizes linear layers with layer normalization in between, followed by ReLU activation functions.

In the model the input positions are fed into the positional encoder, while the file cycle C_{ij} is passed through the timestamp encoder. These two outputs are concatenated and then passed into the Decoder to generate the target location at a given file cycle C_j.

We are utilizing the Adam optimizer with a learning rate of 0.0001, coupled with a learning rate scheduler while keeping the batch size constant at 1000. Our chosen loss function is the L1Norm, which is also known as the Mean Absolute Error (MAE) loss. This loss function measures the absolute differences between the predicted and actual target positions, providing a reliable measure of model performance.

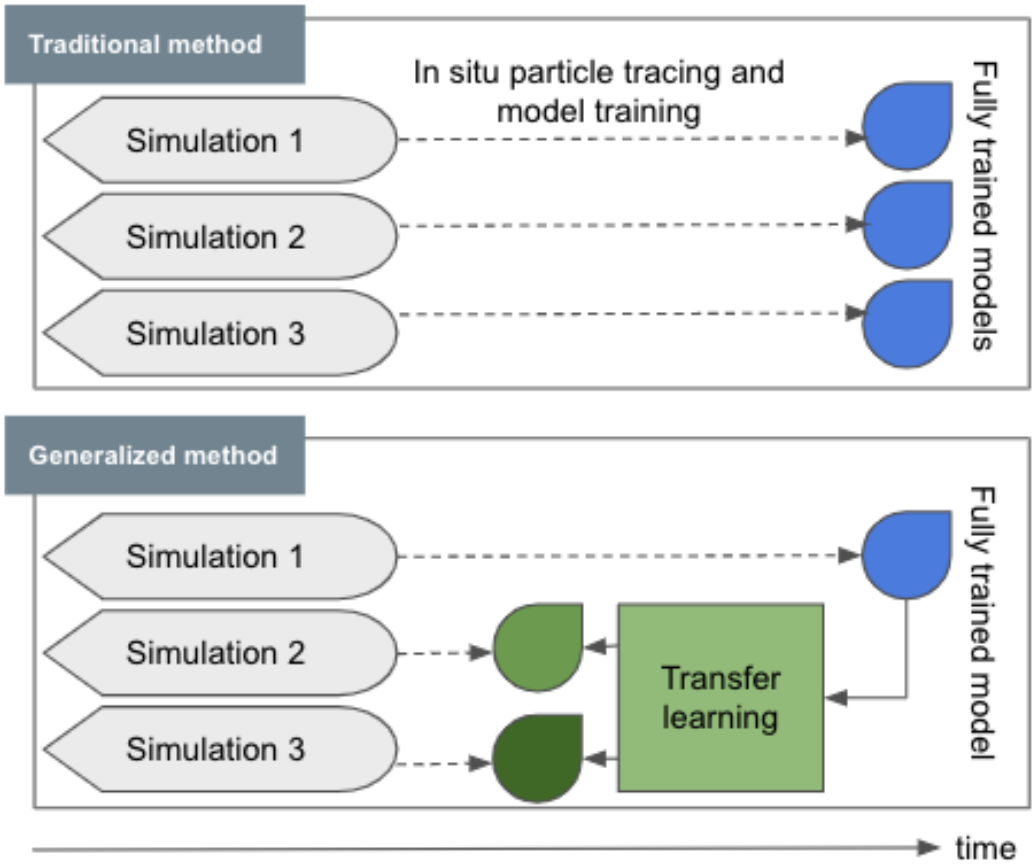

Fig. 3.1: Traditional workflow and proposed generalized workflow. Different simulations might be varying a few parameters and producing very different flows. Traditional approaches train different models for each different simulation cases from scratch. Our proposed approach used transfer learning to generate new models from existing ones dramatically reducing the training time.

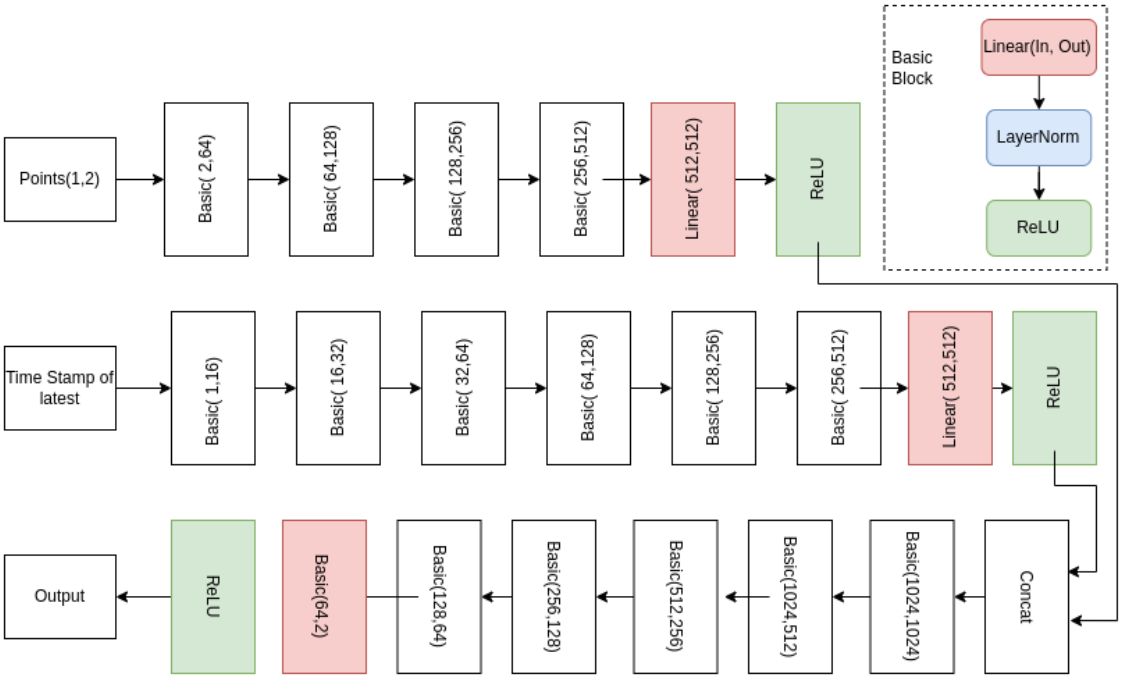

Fig. 3.2: Model architecture based on multi-layer perceptron (MLP). The model accepts a pair of particle start locations and file cycles as input and predicts the particle end location for the specified file cycle.

To train a model to predict the trajectories of particles, the data containing their trajectories and corresponding end positions is stored in an array of shape $[n+1, N, 2]$, where n represents the number of flow maps and N represents the number of seeds. Before being used for training, the data is arranged based on eqn:inputs. The training process includes start locations, denoted as $start_i$, (where i ranges from 0 to $N-1$), queried file cycles C_j (j ranges from 0 to $n-1$) to predict target end location $target_{i,j}$ where (where $i = 0, 1, ..., N-1$ and $j = 0, 1, ..., n-1$). To ensure consistency and accuracy during training, the bounds of the training flow maps, file cycles, start locations, and end locations are first normalized to the $[0, 1]$ range. The normalized bounds are saved and later used to scale the output of the model back to its original form during inference, allowing for accurate predictions on new data outside the range of the original training set.

$$
\begin{aligned}
Inputs = \{ & \{start_0, C_0, target_{0,C_0}\}, \\
& \{start_0, C_1, target_{0,C_1}\}, ..., \\
& \{start_0, C_{n-1}, target_{0,C_{n-1}}\}, ..., \\
& \{start_{N-1}, C_{n-1}, target_{N-1,C_{n-1}}, \}\}
\end{aligned}
\tag{3.1}
$$

3.1 Use of multiple models for particle tracing

Our modeling framework aims to support the prediction of particle trajectories over long and varying time series (of length n). In order to produce accurate predictions we create multiple models (M) which will learn the flow behaviours using a number of separate time spans of a given simulation. Let d be the number of training samples per model defined by Equation 3.2.

$$
d = \begin{cases} n/M - 1 & \text{if } \lambda = M - 1, \\ n/M & \text{otherwise.} \end{cases}
\tag{3.2}
$$

Where λ is the number of the model ($\lambda = 0, 1, .., M-1$) With $Inputs$ defined as following:

$$
\begin{aligned}
Inputs[\lambda] = & \left\{start_0, C_{\varepsilon+0}, target_{0,C_{\varepsilon+0}}\right\}, \\
& \left\{start_0, C_{\varepsilon+1}, target_{0,C_{\varepsilon+1}}\right\}, ..., \\
& \left\{start_0, C_{\varepsilon+d}, target_{0,C_{\varepsilon+d}}\right\}, \\
& \left\{start_{N-1}, C_{\varepsilon+d}, target_{N-1,C_{\varepsilon+d}}\right\}
\end{aligned}
\tag{3.3}
$$

Where $\varepsilon = \lambda * d$. Note that Equation 3.3 is a generalization of Equation 3.1.

In practice, when learning long particle trajectories, each separate model will have to predict the starting location of the same particle trajectory in the next model. For instance, if the first model predicts the trajectory of a particle for timesteps 0 to 100, the next model should predict the trajectory from 100 to 200 timesteps. It is essential that the 100th timestep is predicted by the first model, and this prediction serves as the input for the next model. In this setting we will still expect error to propagate over time (as we are using a $Lagrangian_{long}$ method), but since each model is trained on a shorter time span, the overall accuracy is expected to improve compared to that of a

single model learning trajectories on a long time span. In our experimental study, in the next section, we will validate this hypothesis.

3.2 Transfer learning

In the current scenario, we are utilizing model adaptation and incremental training processes to address our problem. This is because the data distribution in both the source and target domains is similar, as is the feature space, and the models being used are the same. By adapting the existing model and incrementally training it on the target flow data, we can improve its performance on the specific simulation case while leveraging the knowledge gained from the source flow model. For example, we train a model on a particular simulation time span or *Re*. This pre-trained model is then fine-tuned on a new time span or *Re* to learn the particle trajectories in the new flow field. By doing so, the training process is accelerated because the initial layers of the model are already adept at extracting relevant features from the data and making predictions. This allows us to quickly adapt the model to the new task, while leveraging the knowledge learned from previous training.

For this study we have used Pytorch [19] which allows to perform transfer learning from one model to another by: (i) loading the weights from an existing model we want to transfer knowledge from, (ii) creating a new *model* instance, (iii) and using the method $model.load\_state\_dict$ to load the initial weights (previously retrieved).

For our experimentation, we have used the same model architecture for all experiments. We acknowledge that changing parameters, number of hidden layers, loss function, etc. could optimize performance of the models in some circumstances but this is not the goal of this work. By using the same model architecture across different scenarios, we are able to better highlight the advantages that come from transfer learning (alone) rather than from fine-tuning model optimizations. We also believe that the proposed approach can be implemented using any other neural network based model which can accurately predict particle trajectories for a particular simulation scenario.

3.3 Evaluation method

Evaluating the quality of those methods require use of different metrics such as: (i) model performance in terms of loss and training time; (ii) accuracy of the flow features predicted by the

model. The models performance were analyzed using the ”Weight and Biases” software [5], which provided visualization of training and validation loss of each model. Accuracy of the flow features are evaluated computing the mean square error (MSE) of finite-time Lyapunov exponent (FTLE), which is a popular tool to visualize stretching and folding of fluid elements in a flow. Finally, we compute an average distance error to compare the point-wise Euclidean distance of predicted particle trajectories to the ground truth. The distance error will give us a precise indication of how much two particle trajectories differ. Our experimental studies in the next section will also show that training loss, quality of FTLEs and distance error are strictly related.

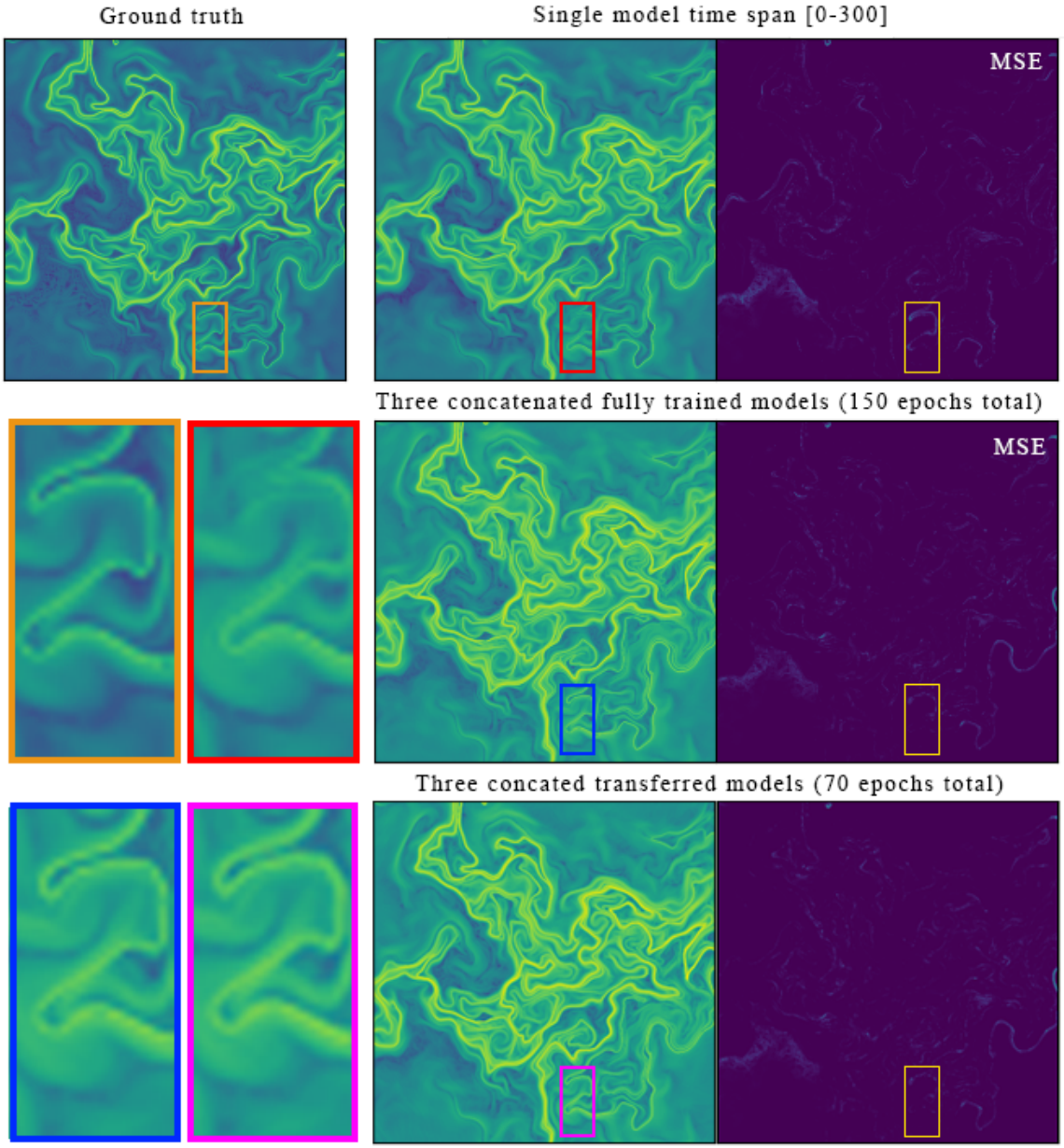

Fig. 3.3: FTLE and MSE for a single model trained on a long time span [0-300] simulation with *Re* 147.0 and three models trained on separate temporal subsets. The single model presents lower accuracy compared to using the three models. In particular in the highlight we can notice how some features are not being captured by the single model. At the bottom, the FTLE from models using transfer learning (one fully trained over 50 epochs and two trained only for 10 epochs) present very similar accuracy to the fully trained models (trained for a total of 150 epochs).

CHAPTER 4

Experimental studies

In this section we describe the experiments we performed to verify the use of transfer learning as a viable method to generalize deep learning models for particle tracing in three scenarios: long time series, varying *Re*, and varying geometry. For the experiments we have used two data sets connected to two different flow setup: (i) an ensemble of 2D flows with spatially-periodic boundaries [15] and varying Reynolds number (using Gerris flow solver); (ii) simulations of 2D flow behind a rectangular obstacle with varying height and the same Reynolds number (using Nek5000). The general layout of the the flow domain for the two cases is illustrated in Figure 1.1.

2D flows in the square box with periodic boundaries at different Reynolds numbers had been simulated using the opensource volume of fluid (VOF) based incompressible Navier-Stokes solver Gerris [20]. The original flow dataset has simulations at multiple *Re*, of which four cases were selected at increasing *Re*: 3.0, 48.5, 147.0, 2352.5. In this setup higher *Re* results in smaller flow vortices.

The flow behind the rectangular obstacle was simulated using Nek5000 [8], which is an open-source high-order spectral element method (SEM) based Incompressible Navier-Stokes Equations (INSE) solver. The domain consists of a uniform Dirichlet velocity inflow boundary, and an open Neumann outflow boundary condition. The lateral boundaries are no-slip walls. The domain has dimensions 18×7. The obstacle has a constant width of 1, and the height is varied from 1 to 1.5. The Reynolds number for both Nek5000 simulations was 200. All training experiments have used 100,000 seed points.

The modeling experiments were computed on a workstation with two NVIDIA GeForce RTX 3080 GPUs running on CUDA version 11.8, with Python version 3.9.0 and PyTorch version 1.13.1+cu117. The seed for the random number initialization was set to 999 manually to ensure reproducibility of the results.

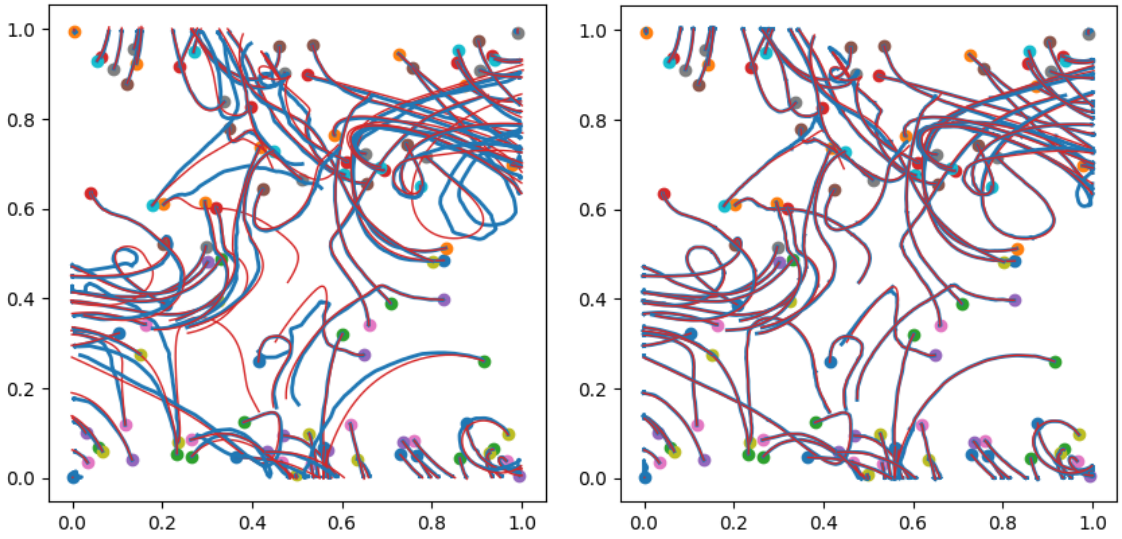

Fig. 4.1: Trajectories of particles predicted by a single model trained on a time span (0-300) and by three separate models trained on subset time spans. Red is ground truth, blue are predictions. The use of multiple models improves dramatically the accuracy for tracing particles over long time series. The mean distance error for the single models is 0.0159 while when using multiple models this error drops to 0.004.

4.1 Particle tracing over long time series

Fluid dynamics simulations are often performed across long time intervals, often resulting in unsteady flow-structures that vary in time and space. Even in one of the simplest flow setup of 2D flow in a periodic box, the vortices that are generated keep on changing spatio-temporally. Thus the first experiment conducted was about predicting particle tracing across long intervals of time, once the model has been trained using data from the start of the simulation.

As described in Figure 1.1 a single model trained on a particular time span is not able to accurately predict particle trajectories in a different time span of an unsteady flow simulation. Furthermore, we want to demonstrate that training a single model on a long time series significantly reduces the accuracy of its predictions. In Figure 4.1 we report on the left trajectories predicted (in blue) using a model trained on a long time span (0-300) of a simulation with *Re* 147.0 vs trajectories predicted using three different models (on the right) each of them trained on a smaller time span (i.e., 100 time steps each). In particular, the mean distance error for the single model is 0.0159 while when using multiple models this drops by almost one order of magnitude to 0.004. This result

clearly calls for the need of multiple models to accurately trace particles over long time spans.

In this set of experiments we split the time series 0-300 in three sections/spans and trained three models from scratch on each of them. Then we used the first model (trained on time span 0-100) to generate the other two models using transfer learning. In Figure 3.3 we report FTLE computations for the two configurations. The single model is not able to accurately capture all flow features (see highlighted area and also Figure 4.1).

Next, we consider the use of transfer learning to quickly derive two of those three models from the first one. Training loss performance reported in Figure 4.2 show how transferred models achieve a significant lower loss faster compared to fully trained models over 50 epochs. In particular, the dotted line in Figure 4.2 shows the loss of a model transferred twice, first from the first time span to the second (10 epochs incremental training). In summary, transfer learned models achieved virtually an identical distance error (over the time target span model) of 0.00435 (compared to 0.00403 of the fully trained models) while using only a total of 70 epochs (50 epochs for the first time spans and a total of 20 for transfer learning on the other two time spans, accounting for 64% less training time). To put things in perspective, if we were to extend this approach to the entire available time series (0-1000) the overall time saving using transfer learning would be of 72% (140 epochs of training instead of 500).

4.2 Particle tracing for varying Reynolds number

In this set of experiments we want to generalize models over different flow regimes, i.e., varying *Re*. High *Re* corresponds to flow regime that has smaller vortices and enhanced mixing, whereas at low *Re* the flow can be said to be more laminar. We have shown in Figure 1.1 how a model trained on a simulation with a particular *Re*is not suitable for use in a different *Re*. In Figure 4.3, we report FTLE results from training different models from scratch on the first 100 time steps of each simulation data for varying *Re*. The models can capture well the features of each flow but still require long training time for each single model (see Figure 4.4 for *Re* 147.0). Using our transfer learning approach we performed experiments to determine if and how transferring a model from a different *Re* would affect the training (and validation) performance.

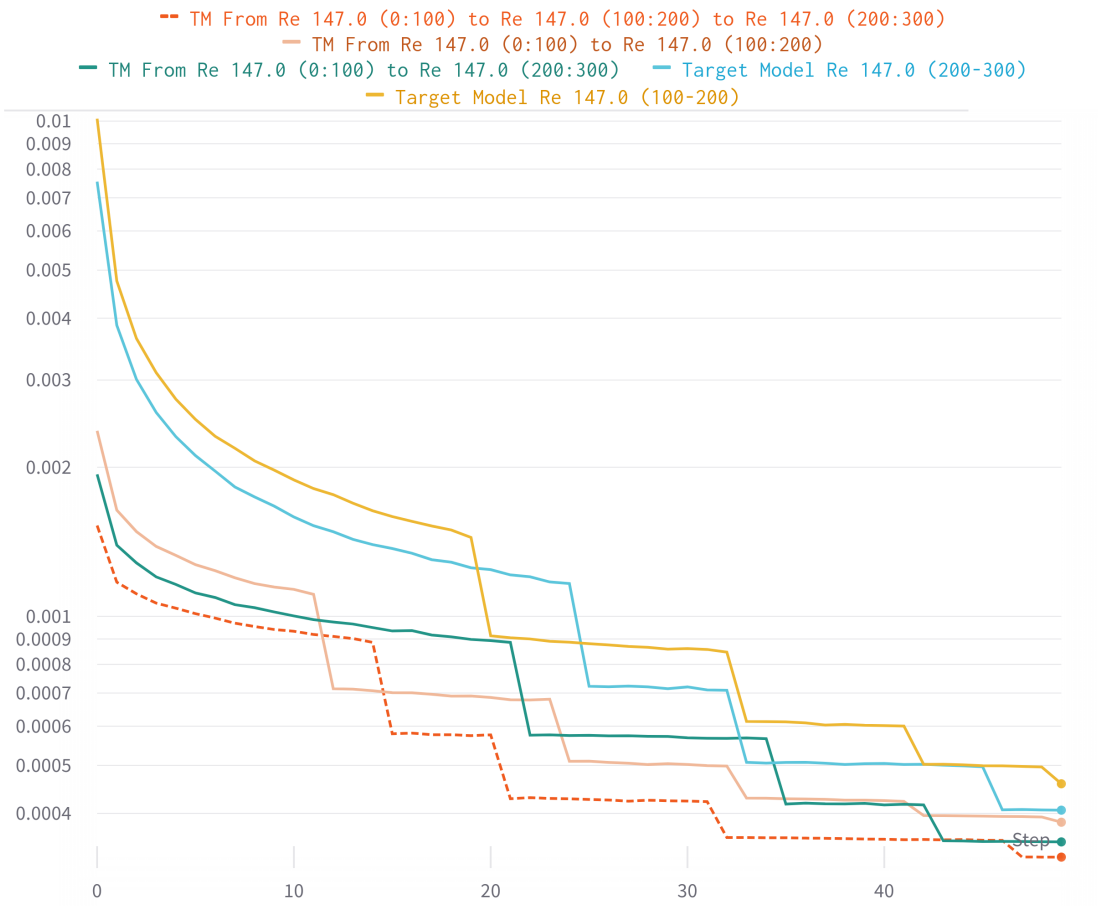

Fig. 4.2: Training loss of models over different simulation time spans. The transferred models (green and orange) present overall a significant lower loss compared to modeled trained from scratch. Both models were transferred from the first time span (0-100) to the second (100-200) and third (200-300) time spans. With the dotted line we report the loss of a model which used transfer learning twice, from both first and second time span. This presents overall the fastest convergence to the lowest loss.

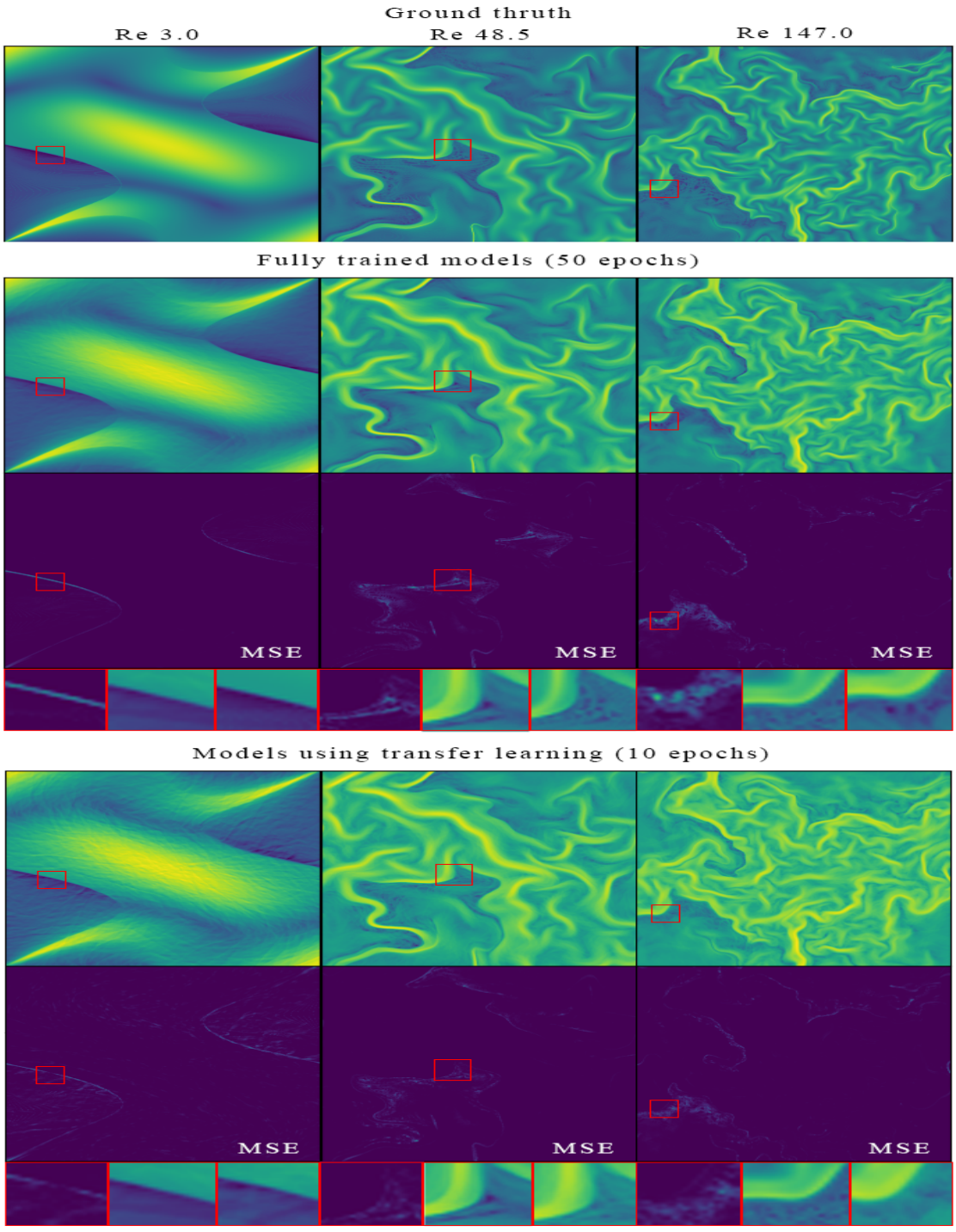

Fig. 4.3: FTLE of different models trained on simulations with different *Re* (3.0, 48.5, 147.0) and corresponding models created using transfer learning from a single model trained at *Re* 2352.5. In red are highlighted areas with higher errors. The three red boxes under each column are details of the MSE, ground through and model. Overall both models are capturing well the flow features, but the transferred models only required 10 epochs of training vs 50 used for the models trained from scratch.

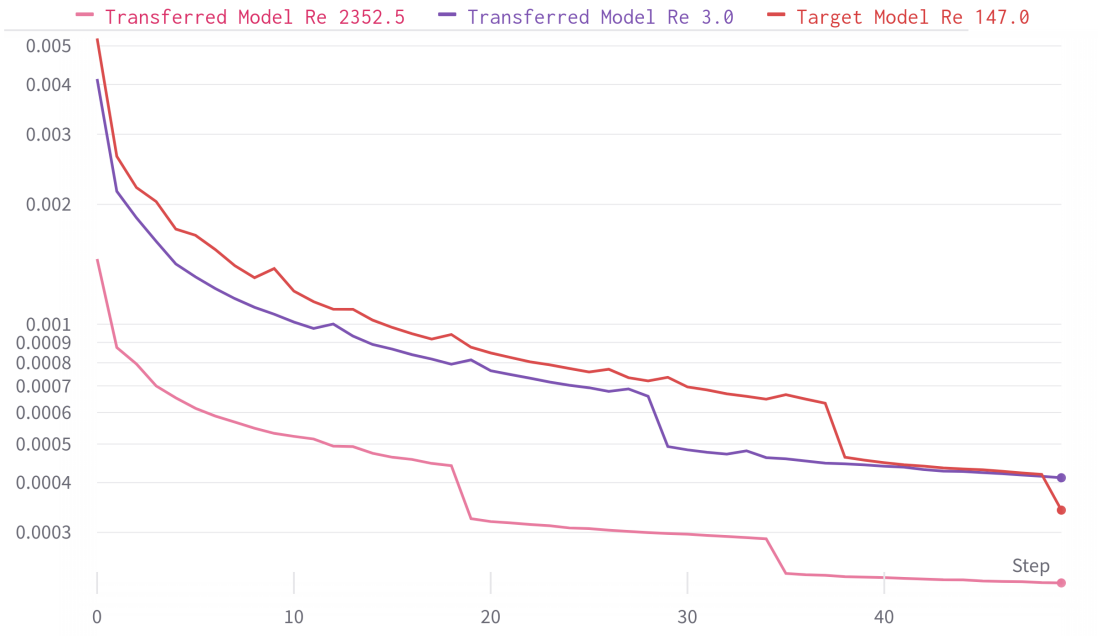

Fig. 4.4: Training loss when performing transfer learning over different turbulence conditions (*Re*). The target models is for a simulation with *Re* 147.0. In red is the loss to train a model from scratch, in purple is the loss when performing transfer learning from *Re* 3.0, in pink is the loss when starting from *Re* 2352.5. Transfer learning from a simulation with higher *Re* allows to quickly train a model that will be used to predict particle trajectories for simulations using a lower *Re*.

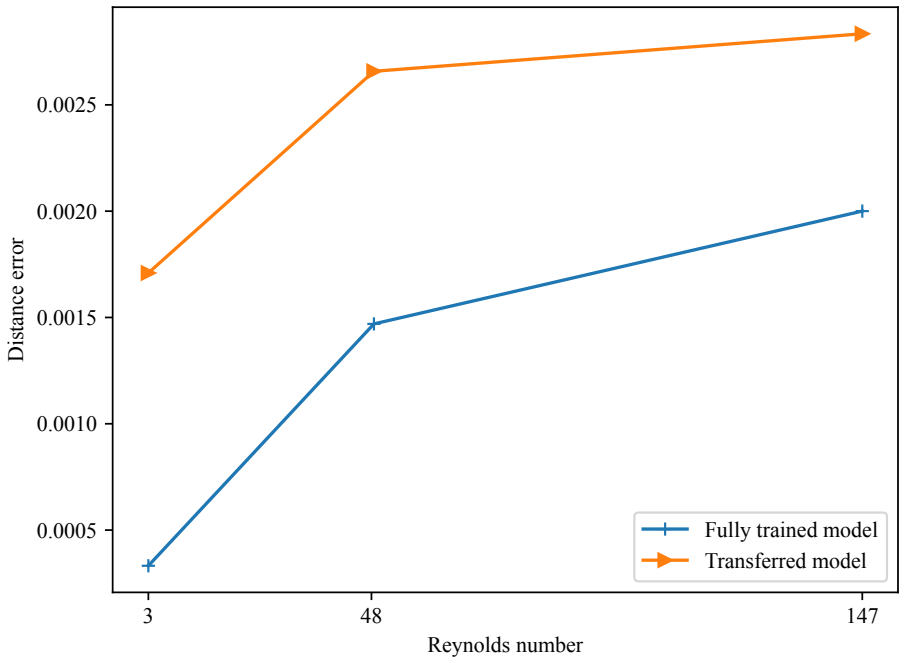

Fig. 4.5: Mean distance error for fully trained models and transferred models (from *Re* 2352.5) when generalizing over turbulence conditions. The error of the transferred model is closer to the one of the fully trained model when the *Re* of the model we transfer from is closer to the target *Re* model. The grid size of this simulation domain is 0.0019, hence the distance error of both models is good.

In Figure 4.3 FTLE computed for transferred models (trained on the flow with *Re* 2352.5) show similar results to the fully trained models, but using only 10 epochs instead of 50. Furthermore, we investigated how to best choose which model to use to transfer knowledge from and how that affects the accuracy and model performance. To do so, we selected a flow with *Re* 147.0 as target for our predictions and observed results when transferring from a higher *Re* vs a lower *Re*.

In Figure 4.4 we report the training loss when performing transfer learning from a model trained on a much higher *Re* (2352.5) and from a much lower *Re* (3.0). These results show that transfer learning from a higher *Re* can significantly speed up the training on models that will be used on a lower *Re*. For example, in only ten epochs our model learns to predict accurately particle trajectories in a simulation with *Re* 147.0 when the model is transferred from *Re* 2352.5; while it would take 30 epochs to achieve the same loss if transferring from *Re* 3.0 (purple line) and 40 epochs if starting from scratch (red line).

Furthermore, we analyzed the relative error (see Figure 4.5) of fully trained and transferred models and noticed that when generalizing over Re the difference of distance error between a fully trained model and a transferred one is smaller when the two Re is closer. This means that similar Re might need fewer fine-tuning epochs when using transfer learning to produce accurate predictions.

These observations prove that it is relatively easier to generalize from higher to lower Re , compared to transferring learning from lower to higher Re. Based on our understanding of the flow, this is not unexpected. Higher Re correspond to regimes with smaller vortices, more complex flow and pathlines having smaller radius of curvature. Thus, when a model is trained using pathline data from a more complex flow, generalization to a less complex flow regime is relatively easier. This trend is also observed in the experiments with varying geometry.

4.3 Particle tracing for varying geometry

For this experiment, we simulate different geometries because geometry has a direct effect on vortex-shedding. Larger bluff-bodies create larger low-pressure zones, which in turn form larger vortices. The geometry of these obstacles also affects the vortex-shedding frequency.

In Figure 1.1 (bottom) we see how a model trained on a simulation of a flow behind an obstacle of height 1.0 is not anymore usable in a simulation with obstacle of size 1.5 because the flow changes quite dramatically.

In this set of experiments we tried to apply our transfer learning approach to a simulation of a flow behind a varying size obstacle. In Figure 4.6 we report FTLE of models trained from scratch on two different cases with obstacle height of 1.0 and 1.5. The two flows present different features. We see very little vortex-shedding in the case of height 1, while in the case of height 1.5, we observe large eddies. When attempting to use transfer learning on this case we want to understand what would be the best model to transfer from, if the one with bigger or smaller obstacle size. We performed experiments in both directions and report the loss in Figure 4.7. The training loss shows that transferred models converge much faster to a smaller loss, in particular when using transfer learning from a taller obstacle to a shorter one the convergence is faster than vice versa. This shows

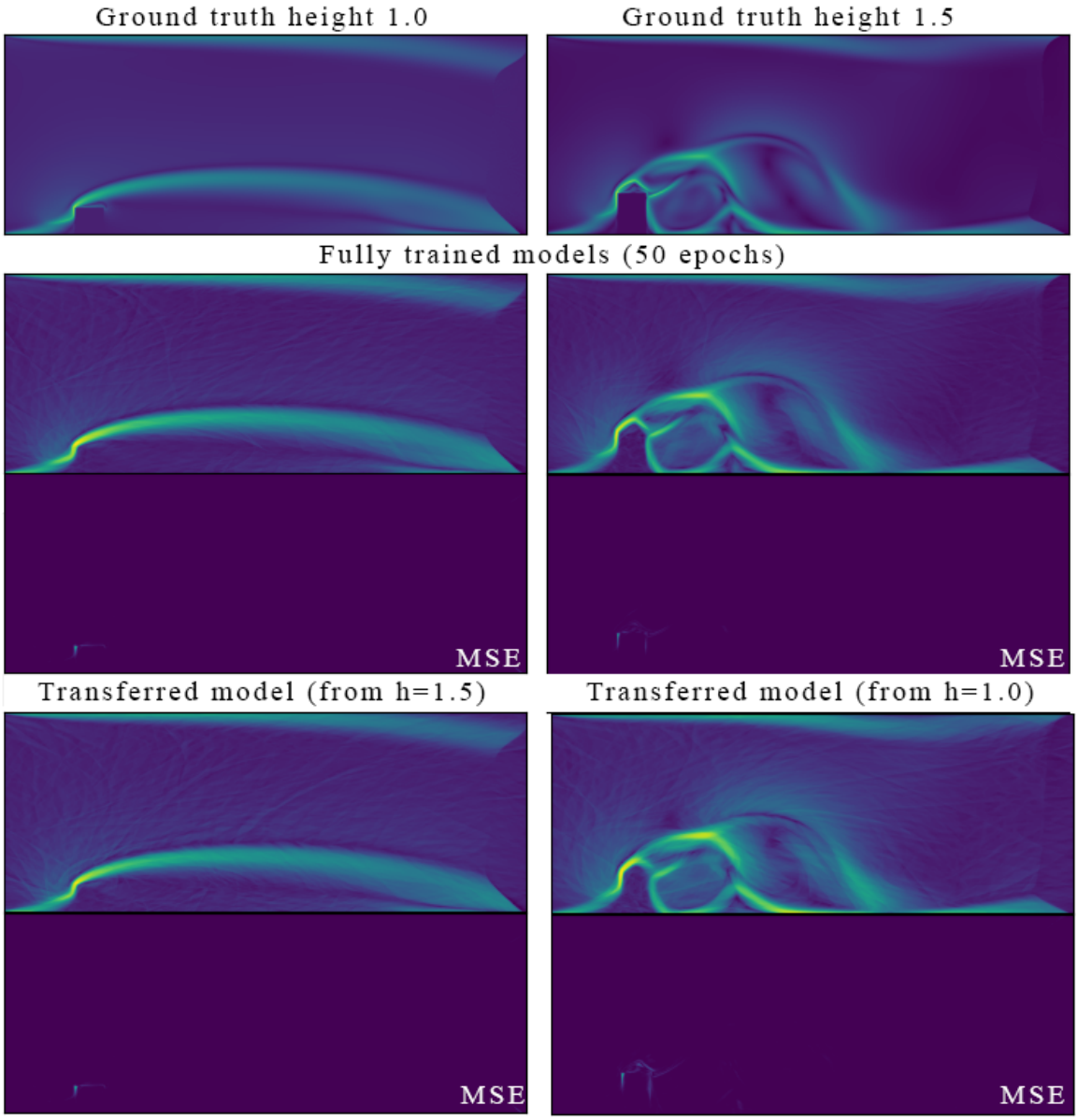

Fig. 4.6: FTLE results for fully trained and transferred model for flow behind an obstacle of varying height.

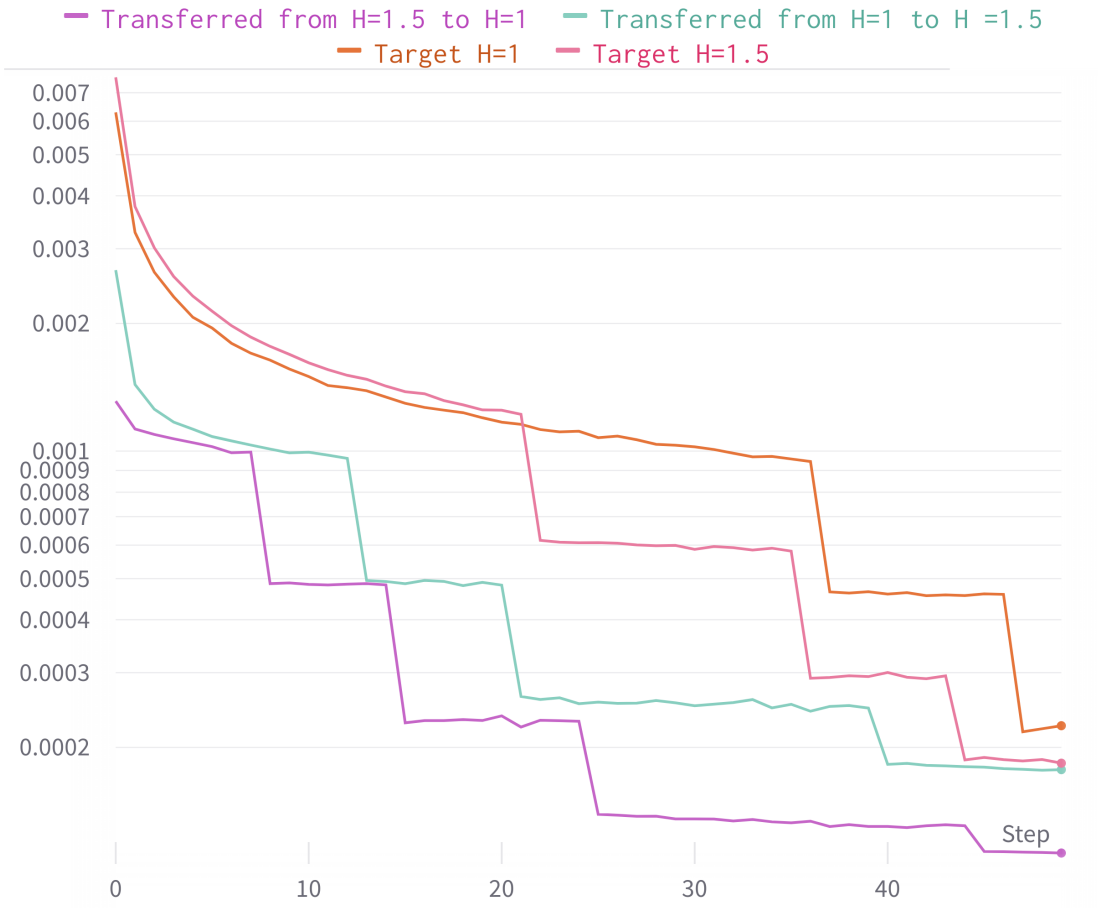

Fig. 4.7: Training loss of fully trained models and transferred models when generalizing over geometry. Transferred models converge to a smaller loss much faster, especially when transferring from a more turbulent flow (i.e., object height 1.5 to 1.0).

that if geometry changes across simulations, it might be better to train using the more complex flow and use that for transfer learning. Distance errors reported in Figure 4.8 confirm this finding as the relative error between transferred and fully trained models is smaller when transferring from the flow with the taller obstacle (see figure 4.7).

4.4 Particle tracing for varying time spans and *Re*

In our final set of experiments we attempt to perform transfer learning over two variables: time spans and *Re* using the periodic flow ensemble simulation dataset.

In particular, in this experiment we want to predict trajectories for a simulation with *Re* 147.0

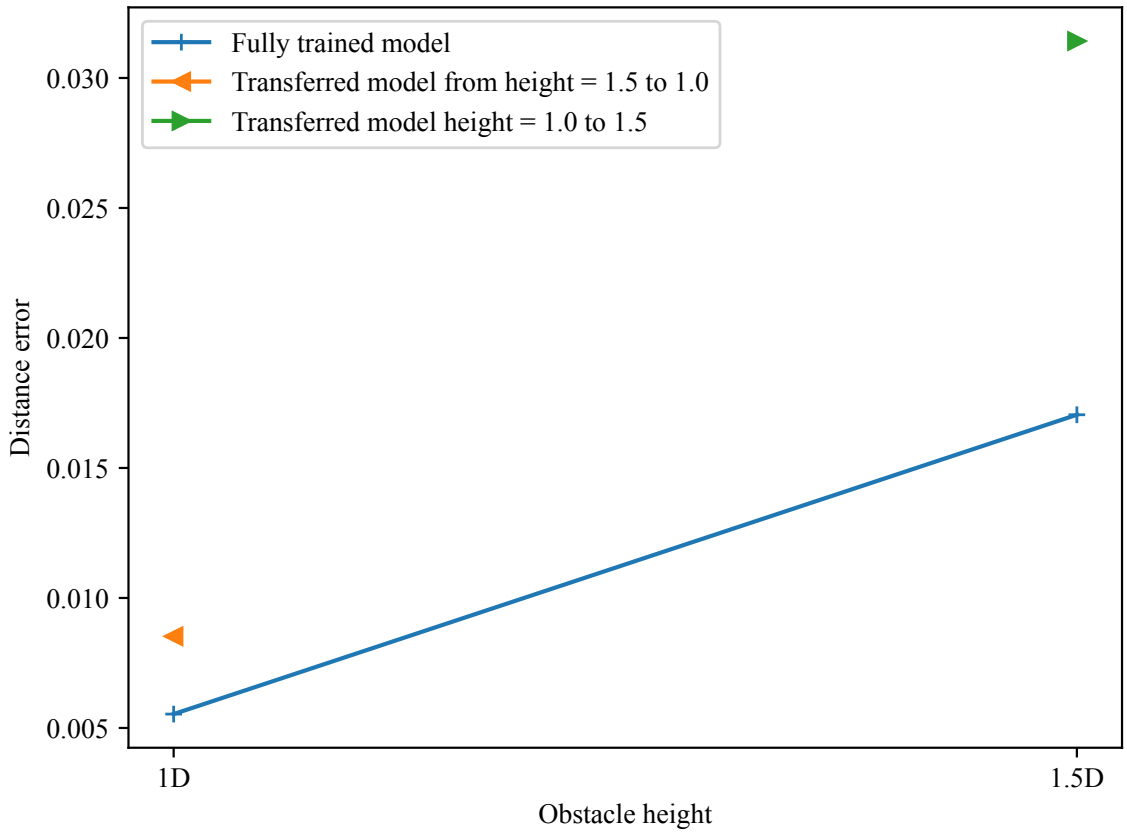

Fig. 4.8: Distance error of fully trained models for flow behind an obstacle of varying height. The fully trained models (trained on 50 epochs) incur in higher error for the simulation with step height 1.5 as the flow presents more turbulence. The transferred models (trained on 10 epochs) show a smaller relative error when transferring from more turbulent (higher step) to more laminar flows (in orange) than viceversa.

at time span (200-300) starting from a model trained with *Re* 2352.5 and time span (0-100). In Figure 4.9 we report the training loss of both fully trained and transferred model. In this case 10 epochs would be enough to reduce the training error and produce accurate results (with a distance error of 0.00126) to predict particle trajectories in the target time span and *Re* configuration. Also in this case, we can notice how when transferring from a model with a lower *Re* (loss in grey) we incur in a higher loss. Furthermore, we can observe in Figure 4.10 how the distance error of the model transferred from a higher *Re*(and different time span) follows the same trend of the training loss.

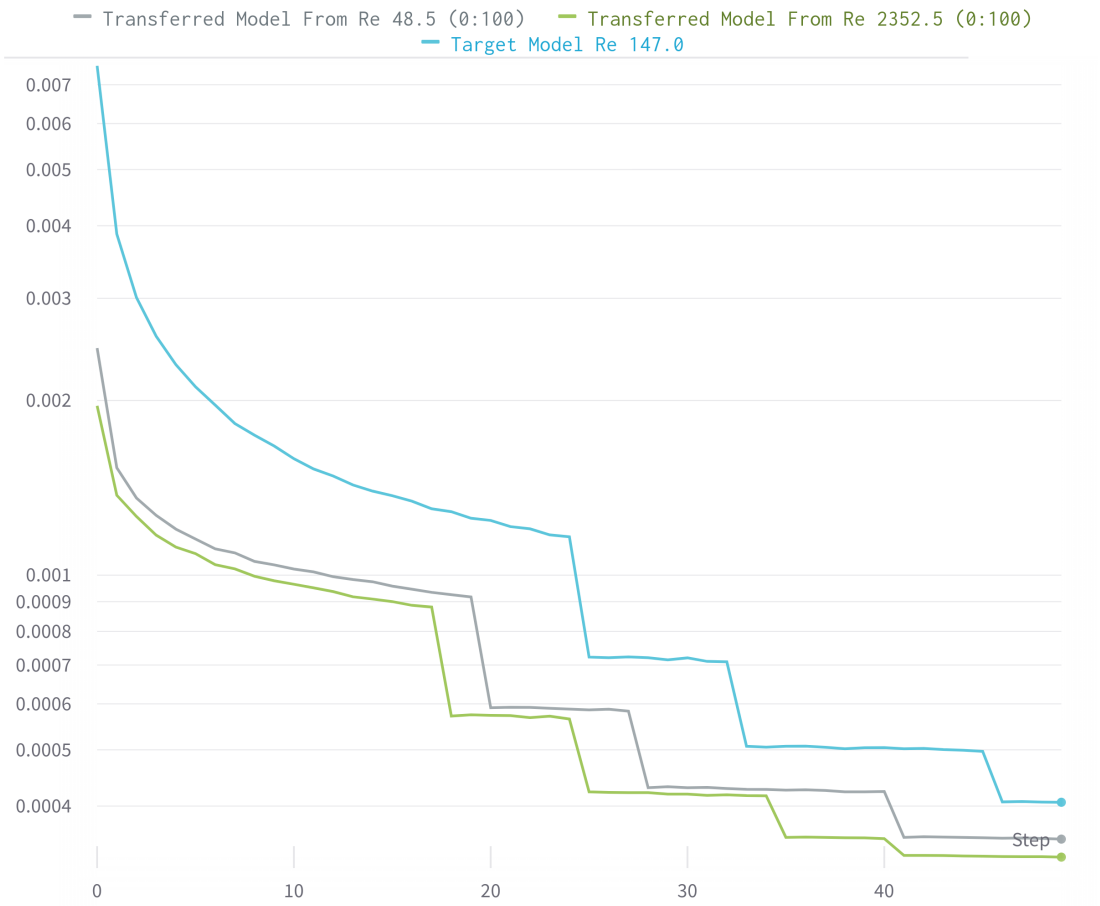

Fig. 4.9: Training loss of fully trained model and transferred models when both *Re* and time span change. The transferred model (green) from higher *Re* converges to a smaller loss faster then the fully trained model (light blue). At 10 epochs the error of the transferred model is already small enough to predict accurate particle trajectories on the new field. In grey we report loss performance for transferred model from a lower *Re* , which as we learned from previous studies incurs in a higher loss.

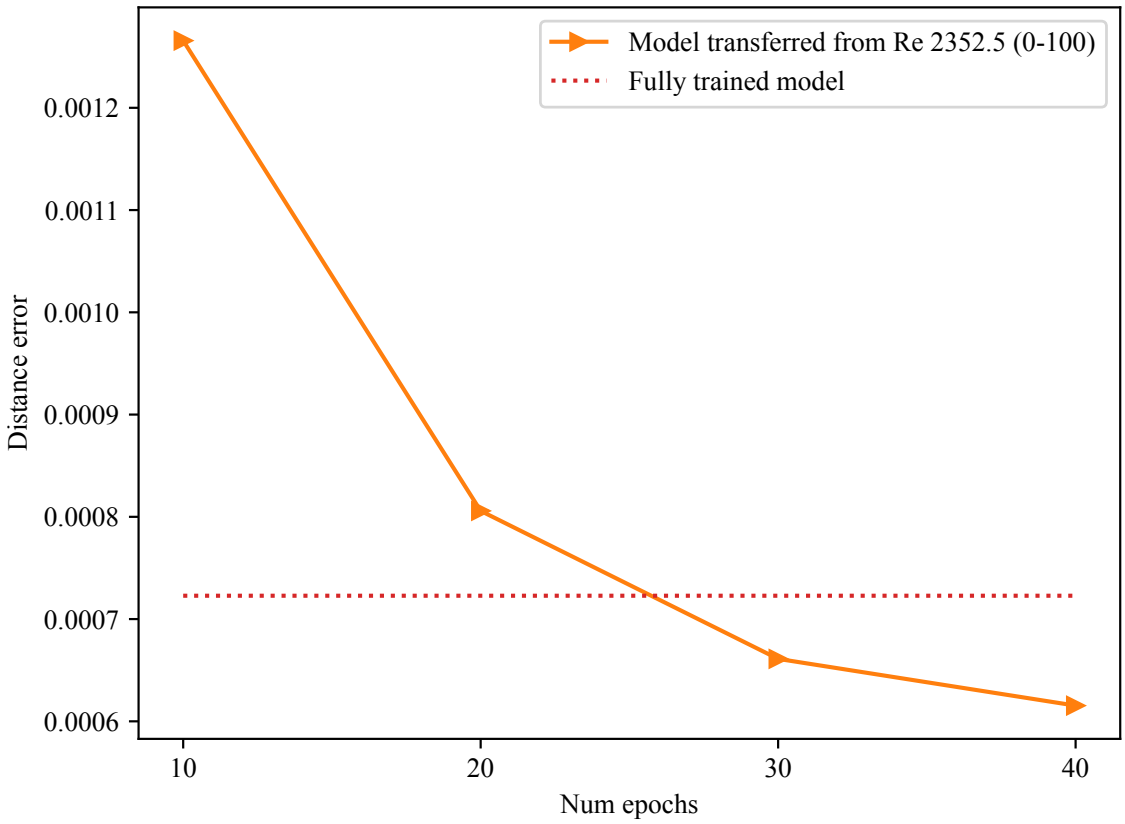

Fig. 4.10: Mean distance error of the transferred model when trained for different numbers of epochs on a target simulation with different *Re* and time span. We can observe how the distance error (computed in validation) follows the same trend that we observe in the training loss.

CHAPTER 5

Conclusions

The visualization of flows using Lagrangian particle tracing requires expensive computation and resources. Deep learning based models have improved the performance of the Lagrangian particle tracing predictors, while maintaining the accuracy of the calculations. However, existing approaches have limited applicability as one trained model can only be used for a dataset that came from a particular simulation. Furthermore, generating any of these models requires long training time and extensive training data. In this work, we presented a simple methodology based on transfer learning that allows generalization of pre-trained models that can be more efficiently applied to different ensemble simulation scenarios. In particular, we address common simulation design configurations which include the integration of particle trajectories over long time series, varying flow parameter (Reynolds number) and varying geometry. Through the numerical experiments we analyzed the benefit of transfer learning and gained insights about choosing the model for training that would facilitate a more effective transfer of knowledge across cases/simulations. Specifically, we have found that for predicting long time series the use of multiple models can predict more accurate particles trajectories that a single model. Moreover, a model trained on a more complex flows can be reused and trained much faster to predict particle trajectories on a less complex flow than viceversa.

This study demonstrates that the use of transfer learning is a viable solution for the generalization of deep learning methods for scientific visualization and can dramatically speed up the development of new models reducing training time and data.

We acknowledge that more complex ensemble simulations might not be addressed by a simple transfer learning with incremental training but instead require a more sophisticated knowledge transformation and adaptation.

REFERENCES

[1] A. Agranovsky, D. Camp, C. Garth, E. W. Bethel, K. I. Joy, and H. Childs. Improved Post Hoc Flow Analysis Via Lagrangian Representations. In *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 67–75, 2014.

[2] A. Agranovsky, H. Obermaier, C. Garth, and K. I. Joy. A Multi-Resolution Interpolation Scheme for Pathline Based Lagrangian Flow Representations. In *Visualization and Data Analysis 2015*, vol. 9397, p. 93970K, 2015.

[3] J. E. Ball, D. T. Anderson, and C. S. Chan. A comprehensive survey of deep learning in remote sensing: Theories, tools and challenges for the community. *Journal of Applied Remote Sensing*, 2017. doi: 10.1117/1.jrs.11.042609

[4] H. Bhatia, S. N. Petruzza, R. Anirudh, A. G. Gyulassy, R. M. Kirby, V. Pascucci, and P.-T. Bremer. Data-driven estimation of temporal-sampling errors in unsteady flows. In *Advances in Visual Computing: 16th International Symposium, ISVC 2021, Virtual Event, October 4-6, 2021, Proceedings, Part I*, pp. 235–248. Springer, 2021.

[5] L. Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[6] L. Deng, Y. Wang, Y. Liu, Y. Liu, F. Wang, S. Li, and J. Liu. A cnn-based vortex identification method. *Journal of Visualization*, 2019. doi: 10.1007/s12650-018-0523-1

[7] S. Dutta, M. W. V. Moer, P. Fischer, and M. H. Garcia. Visualization of the bulle-effect at river bifurcations. In *Proceedings of the practice and experience on advanced research computing*, pp. 1–4. 2018.

[8] P. Fischer, J. Lottes, and H. Tufo. Nek5000, 2007. Argonne National Lab.(ANL), Argonne, IL (United States).

[9] M. M. Ghazi, B. Yanikoglu, and E. Aptoula. Plant identification using deep neural networks via optimization of transfer learning parameters. *Neurocomputing*, 2017. doi: 10.1016/j.neucom.2017.01.018

[10] L. Guo, S. Ye, J. Han, H. Zheng, H. Gao, H. Gao, H. Gao, D. Z. Chen, H. Xiao, J.-X. Wang, J.-X. Wang, and C. Wang. Ssr-vfd: Spatial super-resolution for vector field data analysis and visualization. *null*, 2020. doi: 10.1109/pacificvis48177.2020.8737

[11] J. Han, J. Tao, and C. Wang. Flownet: A deep learning framework for clustering and selection of streamlines and stream surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 2020. doi: 10.1109/tvcg.2018.2880207

[12] M. Han, S. Sane, and C. R. Johnson. Exploratory lagrangian-based particle tracing using deep learning. *Journal of Flow Visualization and Image Processing*, 29(3), 2022.

[13] W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, M. Raj, Y. S. G. Nashed, and T. Peterka. Insitunet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 2020. doi: 10.1109/tvcg.2019.2934312

[14] F. Hong, J. Zhang, and X. Yuan. Access pattern learning with long short-term memory for parallel particle tracing. *null*, 2018. doi: 10.1109/pacificvis.2018.00018

[15] 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.

[16] C. Liu, R. Jiang, D. Wei, C. Yang, Y. Li, F. Wang, and X. Yuan. Deep Learning Approaches in Flow Visualization. *Advances in Aerodynamics*, 4(1):1–14, 2022.

[17] J. R. Martins. Aerodynamic design optimization: Challenges and perspectives. *Computers & Fluids*, 239:105391, 2022.

[18] K. Mittal, S. Dutta, and P. Fischer. Direct numerical simulation of rotating ellipsoidal particles using moving nonconforming schwarz-spectral element method. *Computers & Fluids*, 205:104556, 2020.

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[20] S. Popinet. Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries. *Journal of computational physics*, 190(2):572–600, 2003.

[21] A. Rozantsev, M. Salzmann, and P. Fua. Beyond sharing weights for deep domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. doi: 10.1109/tpami.2018.2814042

[22] S. Sahoo, Y. Lu, and M. Berger. Neural Flow Map Reconstruction. In *Computer Graphics Forum*, vol. 41, pp. 391–402. Wiley Online Library, 2022.

[23] S. Sane, C. R. Johnson, and H. Childs. Investigating In Situ Reduction via Lagrangian Representations for Cosmology and Seismology Applications. In *International Conference on Computational Science*, pp. 436–450. Springer, 2021.

[24] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. J. Mollura, and R. M. Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *arXiv: Computer Vision and Pattern Recognition*, 2016. doi: 10.1109/tmi.2016.2528162

[25] 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*, 2022.

[26] D. Wang, D. Wang, D. Wang, and T. F. Zheng. Transfer learning for speech and language processing. *arXiv: Computation and Language*, 2015. doi: 10.1109/apsipa.2015.7415532

[27] Y. Yuan, X. Zheng, and X. Lu. Hyperspectral image superresolution by transfer learning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2017. doi: 10.1109/jstars.2017.2655112

[28] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.