ACSE-9: Applied Computational Science Project

Shallow water flow field inference using convolutional neural networks with particle images

Mingrui Zhang

Supervisor: Matthew Piggott, James Percival

Submitted for the degree of
MSc Applied Computational Science and Engineering

Imperial College London
mingrui.zhang18@imperial.ac.uk
Github:@erizmr

August 2019

Abstract

A network for end-to-end fluid flow inference called Particle-Net is presented in this project. The Particle-Net is able to infer the flow velocity field for shallow water problems from images of advected particles. Particle-Net outperforms the state-of-the-art particle image velocimetry (PIV) software PIVlab on synthetic data sets, both in terms of accuracy and efficiency, especially on extremely low-quality particle images. The root means square error and relative error of Particle-Net inference result are 60% lower than that of PIVlab. In addition, even though Particle-Net was trained using only synthetic data, it still shows a promising ability to infer flow fields from real lab data even with sparse particles, although it does not outperform PIVlab on all lab data. Particle-Net processes images 25 times faster than PIVlab and is able to process nearly 72 pairs per second, which is quite promising for real-time fluid flow inference.

Acknowledgements

Above all, I would like to express appreciation to my supervisors: Professor Matthew Piggott and Dr James Percival. Their patient guidance and insightful instructions help me overcome difficulties and find the right direction throughout this project. I would also like to express my gratitude to Mr Lucas Mackie and Mr Raul Adriaensen. The high-quality particle images they provided supports my work and give me the chance to further verify my model using real lab data. Thanks to Dr Lukas Mosser and Mr Jianhong Wang for discussing with me about machine learning problems when I got stuck. Finally, I would like to thank my parents, for their unlimited support for me all the time.

Contents

1 Introduction

Shallow water problems are quite common in the real world. Many fluid flows in the oceans, coastal regions, estuaries, rivers and channels can be defined as shallow water flows. The general characteristic of these flows is that the vertical dimension is much smaller than the typical horizontal scale. In this case, the vertical dimension can be removed by averaging over the depth. Tidal currents represent a typical shallow water flow. Tides are also able to produce promising renewable energy due to high energy density close to shore and periodic occurrence. Tidal energy has the potential to become a valuable part of the energy supply in the drive towards a carbon-free society [1]. Tides and tidal energy thus represent a motivating physical case which requires the development of new modelling, analysis and prediction techniques.

Tidal turbine farms can be used to collect tidal energy. One critical challenge for constructing the farm is how to choose an optimal farm location and size. These usually decide the availability and capacity of tidal energy. To deal with it, coastal modelling can be used to help analyze and predict the potential energy available. Traditional coastal modelling relies on computational fluid dynamic (CFD), which solves the Navier-Stokes (NS) equations. However, due to high computational costs, CFD-based approaches are not sufficiently efficient when dealing with large scale problems [2].

One possible approach to improve efficiency is through flow field inference. Given collected oceanic data (such as satellite observations, drone images, limited sensor measurements, etc.), an inference model could predict the related physics properties (velocity field, vortex strength, tidal resource, etc.) of the target area. These results could then be assimilated into numerical models and help conduct rapid simulations. These data usually describes the complex ocean flow phenomena. To better understand and analyze these data, it requires flow field quantitative measurement methods.

1.1 PIV method

In experimental fluid mechanics field, image processing techniques are used for flow field identification and prediction such as the measurement of flow velocities. Particle Image Velocimetry (PIV) is one of the most popular measurement technique. The method is used to obtain instantaneous velocity measurements and related properties in fluids.

When conducting the PIV technique, the target fluid is seeded with sufficiently small tracer particles. These particles are assumed to faithfully follow the flow dynamics. With illumination (often using lasers), the particles in the fluid are visible. By comparing the flow images between timestamps, the flow motion information can be calculated using image processing algorithms [3].

PIV method is widely used in lab-scale experiments. However, when considering large real-world problem such as ocean modelling, there are some difficulties. Above all, the real oceanic data is much more sparse than lab-scale data - there are not always dense tracers in the flow field like such as a high density of particles. It is hard for PIV technique to search for correlation between particles in a sparsely seeded image pair. Also, real oceanic data is usually of large scale, an extreme high computational cost can therefore be required for PIV technique to process these data. Therefore, it is appropriate to take data-driven or machine learning methods into consideration to perform these analysis and inference tasks.

1.2 Machine Learning Method

Machine learning methods, especially deep learning, have made great progress in applications to many real-world problems in recent years along with huge improvements in computational ability. Deep learning techniques show an obvious advantage when tackling problems which are complex and hard to model. These problems include image processing, computer vision, natural language processing, etc [4, 5]. Some highlights of deep learning techniques are introduced below.

A Convolutional Neural Network (CNN) is a kind of deep neural network, which is most commonly applied to computer vision. The CNN extracts features from objects using convolutional kernels in multiple hidden layers simulating the visual recognition in biological processes [4]. A recurrent neural network (RNN) is a kind of neural network where connections between nodes form a directed graph along a temporal sequence. This allows it to process time-series data. RNNs can use their internal memory to process sequences of inputs [6].

Deep learning technique can be regarded as a surrogate modelling method for inverse problems. The key idea of the common deep learning technique is using neural networks as substitutes for the physical model. The inspiration of neural networks comes from imitating the intelligent thinking of the brain [7]. There are several layers for each neural network. Each layer has a series of simple processing units, called neurons. The output value of each neuron is obtained by applying a transfer function on the weighted sum of its inputs. The neurons weights are optimized(or trained) using an algorithm such as stochastic gradient descent, by minimizing a cost function on a training data set [8]. A well-trained neural network is expected to extract features and information from the underlying physics process. Then the direct mapping relationship between observed data and solutions are established. When given new data, the network is able to predict the solutions according to the learned features.

Flow field inference is a typical inverse problem, which aims to find the relationship between observations and interesting physical properties. It is appropriate and timely to apply deep learning techniques to this problem [9]. Some related work has been reported recently, a literature review is provided in the next section.

2 Background Research

Here the background research is divided into two parts. The first is an overview of machine learning methods applied in the fluid mechanics area. The second part is about flow inference or estimation problems, which is our main focus in this project.

2.1 Machine learning in fluid mechanics

Machine learning techniques have been widely used in the field of fluid dynamics in recent years. The main applications include flow field inference, dimension-reduction, turbulence modelling and flow control.

Many works have been reported on flow field inference using machine learning methods. Rabault et al. [3] proposed a proof-of-concept for applying artificial neural network performing particle image velocimetry and compare the results with traditional PIV methods. Erichson et al. [10] implement flow field reconstruction using limit velocity sensor measurements with a shallow fully-connected neural network. Deep learning techniques are also applied to unsteady flow field predictions. Miyanawala et al. [11] proposed a neural network based method to predict drag coefficients of unsteady wake flow given the geometry and velocity field as inputs. In unsteady fluid field visualization area, a Convolutional Neural network (CNN) based feature extraction frame was implemented by Kim et al. [12], which achieves higher accuracy compared to linear optimization methods. In addition to spatial flow field predictions, a temporal-spatial unsteady flow prediction method was reported by Lee et al. [13]. A GAN (Generative Adversarial Networks) combined with CNN was applied to process flow time series data and make predictions. As for ocean current scenarios, Bolton et al. [14] implemented a CNN algorithm to predict unresolved turbulent processes and subsurface flow fields using oceanographic observations. Franz et al. [15] proposed ocean eddy identification and tracking method through the use of C-LSTM (convolution long-short-term-memory) and CNN.

Dimension-reduction is another important application. Wang et al. [16] presented a reduced-order fluid dynamics systems using deep learning. The method is a combination of traditional POD (proper orthogonal decomposition) [17] and neural networks, which showed a better performance compared to POD only method. Xiao et al. [18] made a step further, applying POD-NN to generate a ROM (Reduced Order Model) for turbulent flows in the urban environment. Dimension-reduction modelling also applied to Reynolds stress models, Zhang et al. designed a neural network to predict the Reynolds stress for channel flow at different Reynolds numbers [19]. The results showed a higher efficiency compared to DNS (direct numerical simulation) and LES (large eddy simulation) methods.

When making the trade-off between interpretability and tractability or efficiency, the data-driven method usually sacrifices the former when the network gets deeper [20]. However, turbulence (and related complex system) modelling highly requires physical interpretability. Common “black box” deep learning methods are not suitable here. Therefore, Physics-Informed neural network have been proposed [21, 22, 23], which combine some physics rules as prior knowledge. Ling et al. [24] applied a neural network to a RANS (Reynolds averaged Navier Stokes turbulence models), learning the Reynolds stress anisotropy tensor from high-fidelity simulation data. Erihson et al. [25] implemented a physics-informed auto-encoder for Lyapnov-stable fluid flow prediction, using the stability of equilibrium as prior knowledge.

As for flow control, Rabault et al. [26] presented the application of an Artificial Neural Network trained through a Deep Reinforcement Learning agent to perform active flow control. Morton et al. [27] also proposed an unsteady flow active control method using deep learning. A stable dynamical model is trained according to labels from Koopman theory. The model is able to predict the time evolution of the cylinder system over extended time horizons.

2.2 Flow estimation and inference

Optical flow is defined as the distribution of apparent velocities corresponding to the movement of brightness patterns in an image [28]. There are many similarities between optical flow and fluid flow inference problems. The target of both problems is estimating an object's motion. The difference is, in optical flow, the target object is usually a rigid body, but in fluid flow inference, the targets are fluid parcels. The PIV method acts like a bridge between these two – the fluid is represented by many small rigid bodies, say particles, which could be regarded as an extreme dense inference problem in optical flow. There is various research works on the topic of optical flow estimation in the computer vision field. For traditional methods, in [29], Horn and Schunck proposed variational methods, which have been widely applied in optical flow estimation. Further, Brox et al. integrate rich descriptors into the variational formulation in [30]. For machine learning method, the FlowNet [31] proposed by Fischer et al. achieves promising results for optical flow estimation using a convolutional neural network. Their successor works FlowNet2 [32] further improve accuracy. These important previous works inspire our work developing fluid flow inference methods.

In this project, we focus on the fluid flow field inference problem in the shallow water scenario. The related methods mentioned above for fluid flows mainly compare the machine learning method with Computational Fluid Dynamics (CFD) simulation results. In our case, in addition to simulation data, a shallow water lab experiment's data is used as a benchmark to our methods in this project. The next section describes the software developed in this project, explains how the code is verified against existing state-of-art PIV software and describes an application to real-world lab data. Finally, some conclusions of this work are drawn.

3 Software description

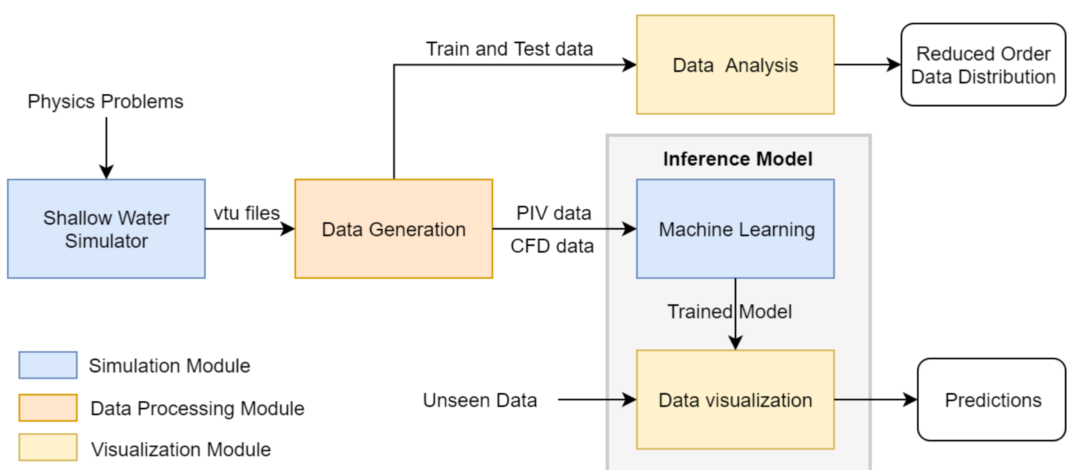


Figure 1: Software Architecture.

The software developed in this project infers the flow field using PIV data pairs. As shown in the graph above, the software consists of several modules, which are divided into three types: simulation module, the data processing module and visualization module. For simulation modules, a shallow water simulator is developed based on Firedrake and Thetis, which solve shallow water equations using Computational Fluid Dynamics techniques and outputs flow field results such as velocity, pressure, elevation, etc. These results are stored in .vtu files. The machine learning module provides functions for data pre-processing, neural network construction, model training and validation, etc. The trained model will be sent to data visualization predictions made using the unseen data as model inputs.

There are two data processing modules, the data generation module reads and processes .vtu files. The module first inserts particles into the flow field then calculate the positions of particles using a Lagrangian approach according to the velocity information in the .vtu files. These particles position and trajectory information are stored in .vtk files. Further, the .vtk files are processed and transformed into PIV image pairs data. The pair here means two consecutive frames for each time step. These pairs are input data for the network. But for training purpose, these data have to be given labels, which means the ground truth. In this project, the underlying CFD results are considered to be the ground truth. Therefore, besides generating the PIV pairs, the corresponding velocity information is extracted from .vtu files and processed as the labels.

The data analysis module is used to analyze the training and test data. By using PCA (Principal Component Analysis) method, the high-dimension training data is reduced to two dimensions. Then the distribution of both training data and test data can be calculated. This can be used to compute KL-divergence between the two distributions and to check whether the training and test data come from the same distribution.

3.1 Software Development Life Cycle

3.1.1 Ecosystem

The shallow water simulator in this software is developed based on Thetis, which is an extension of the Firedrake ecosystem. Firedrake is an automated system for the solution of partial differential equations using the finite element method (FEM). Firedrake uses sophisticated code generation to provide mathematicians, scientists, and engineers with a very high productivity way to create sophisticated high-performance simulations [33].

As for the machine learning part, the codes are developed based on PyTorch, which is an auto-grad machine learning framework. The software is developed under the Agile approach. An Agile software development approach emphasizes adaptive planning and evolutionary development. This research project is an exploration of machine learning techniques applied to flow field inference using particle images. Thus, more cutting-edge techniques would be assimilated into the software. It requires a rapid and flexible response to change, which is what the Agile approach encourages.

3.1.2 Functionality

Shallow Water Simulator Shallow water equations (SWE) describe a physical problem where the horizontal length scale is much greater than the vertical length scale. The simulator in this software solves the SWE using the Firedrake finite element framework and the unstructured grid coastal ocean model Thetis. In this project, a flow past cylinder test problem is taken into consideration as a typical example. The inputs are physical parameters such as initial velocity, the geometry of the cylinder, water viscosity, drag coefficient of the basin, etc. The outputs are horizontal depth-averaged velocity in two dimensions and water elevations at each time step.

Data Generation The detail of the data generation component of this work is described in figure 3. There are two parts to the data generated in this module. The first part is the particle. Particle Module [34] is a Python based module for driving Lagrangian particles via vtk formatted data with collisions. It is used here to generate particle data in this project. The outputs of the Particle Module are .vtk files which only contain the positions of each particle at every time step. Therefore, here comes two problems. The first is that the .vtk files only provide particles (illuminated pixels) in a particle image but not the empty positions. In addition, the particle size in the real world is not an ideal point. It should be decided how many pixels a particle occupies. In order to solve the two problems above and complete the images, a definition and an algorithm is proposed here.

In reality, PIV data are grayscale images whose gray values represent the illumination of particles. Here, the gray value of each pixel is defined as a two-dimension Gaussian distribution

$$I(x,y) = I_0 \exp\left[\frac{(x-x_0)^2+(y-y_0)^2}{-0.125{d_p}^2}\right], \tag{1}$$

where I_0 is the max gray 255 (or 1.0 in a normalized range), d_p is the diameter of the particle, x and y are the positions of the current pixel, and x_0 and y_0 are that of the current particle being considering. Due to the heavy computational cost of traversing every pixel when considering each particle, a domain is defined as a kernel (shown in figure2 below). The illumination function only takes the pixels inside the kernel into consideration.

The algorithm below shows the specific process of generating PIV images from position information in the vtk files:

Algorithm 1 Generate PIV images

function PIVGENERATOR($particle\_array, img\_size, ranges, dp, kernel\_size, noise$)
    $M =$ MATRIX(img_size) ▷ Initialize the Image matrix
    $particle_array =$TRANSFORMATION($particle\_array, img\_size, ranges$) ▷ Convert unit to pixels
    $kernel =$KERNELGENERATOR($kernel\_size$)
    for x, y **in** $particle\_array$ **do**
        $idx, idy =$BOUNDCHECK(x, y)
        for $index$ **in** $kernel$ **do**
            if BOUNDCHECK($idx + index, idy + index, kernel = True$) **then**
                $M_{idx,idy}$ += GAUSS2DILLUMINATION($index, dp$)
            else
                continue
            end if
        end for
        if $Noise$ **is** $True$ **then** ▷ Noise filtering if needed
            $upper_bd = factor_u *$MAX($M_{i,j}$)
            $lower_bd = factor_l *$MAX($M_{i,j}$)
            FILTERING($M, upper\_bd, lower\_bd$)
        end if
    end for
    return M
end function

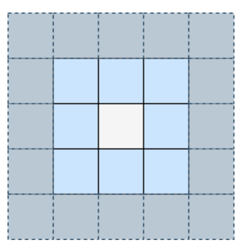


Figure 2: kernel size with one pixel and two pixels.

The other part of the data is a label or ground truth. The unlderying CFD solver results are stored in vtu files. In these files, the physical property values are linked to mesh grid points discretely. With the *scipy.interpolate.griddata* function, the empty entries could be filled quickly using cubic interpolation. Finally, every two consecutive PIV image pairs with a label formed one sample. All samples consist of the whole dataset.

Machine Learning In this module, the codes for the whole workflow of deep learning techniques are developed. The dataset is loaded and divided into training, validation and test sets. Then data pre-processing technique such as normalizing is conducted. The network model is constructed and trained using training data. The error function is also defined in this module. Finally, the trained model is evaluated by the validation set and saved in pt file.

Data Visualization In the data visualization module, the trained network model is loaded. The model makes predictions using test data then visualized by plot functions.

Data Analysis This module provides tools to analyze data properties in different dimensions. A Principle Components Analysis method is implemented in this module, the input data can then be reduced to low dimensions and then interpolated using statistical distributions. Also, a function calculating Kullback–Leibler divergence between distributions is also developed.

3.1.3 Novelty and creativity

The novel contributions of this work are highlighted here.

Develop and Implement the Particle-Net Develop, implement and train the new network Particle-Net using PyTorch. The code developed includes the network structure construction, error definition, data pre-processing, training and validation tools, etc. Training of the network use a different strategy and combination of the dataset, and conduct parameters tuning. Development of the visualization tool for all the results.

Implement the shallow water simulation using Thetis Set up the shallow water simulation based on Thetis. Design the series of simulation experiments including the lab experiment setting, and perform parameter selections. Run the simulations and collect CFD data in vtu files.

Flow field particle data generation Generate all particle images with the CFD data using ParticleModule [34]. Writing scripts and set up environment for ParticleModule on High-Performance Cluster of Imperial College London (cx1.hpc.ic.uk).

Develop synthetic and lab data processing/analysis tool There are several procedures (shown in Figure 3) in the data generation process. And there are four different data sources, each has its own format (vtu, vtk, txt, images). Therefore, a multi-format support data processing tool is developed. The specific functions including, generating PIV image pairs and labeling from vtk and vtu files. Generate the labeled training data set. Process lab data and convert them to suitable resolution for the neural network. Process PIVlab output data, and make it comparable with neural network inference. Also, a dimension reduction analysis tool for training, test and lab data is developed.

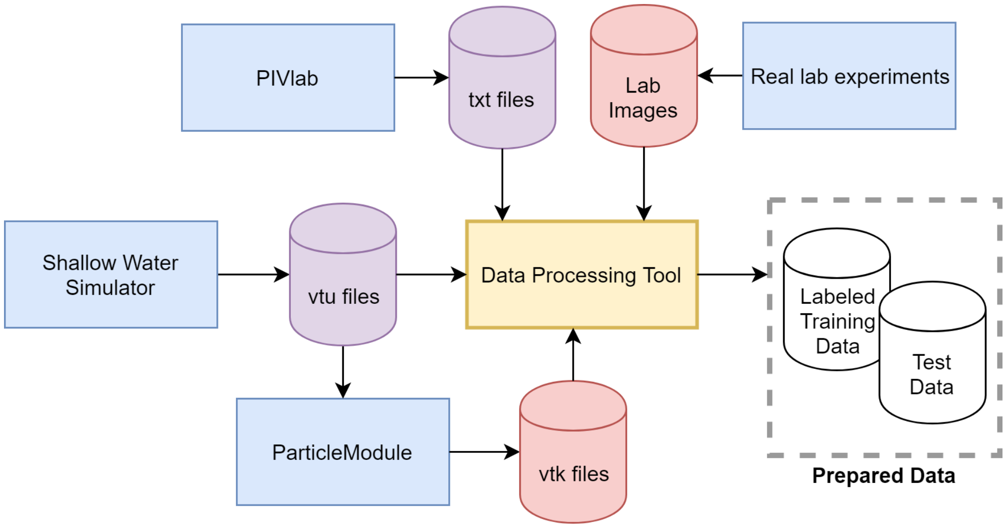


Figure 3: Details of the data generation module. The shallow water simulator, PIVlab, data processing tool are mainly executed on a personal computer. The main platform for ParticleModule is HPC.

3.2 Code metadata

Platform

Python3 is the main developing language of the software. In addition, the libraries and framework below are included. For the developing platform, the software is developed under Mac OS Mojave version 10.14. The data generation module is run and tested on an Imperial College London High-Performance Cluster (cx1.hpc.ic.uk). The platform for machine learning is Google Colab, it provides a NVIDIA Tesla K80 12GB GPU. The user can use one GPU up to 12 hours per session for free.

Libraries and framework

The following are utilised in this work:

1. Firedrake (2019)

   Firedrake is an automated system for the solution of partial differential equations using the finite element method (FEM). Firedrake uses sophisticated code generation to provide mathematicians, scientists, and engineers with a very high productivity way to create sophisticated high-performance simulations [33]. In this project, Firedrake is the underlying framework for Thetis.

2. Thetis (2019)

   Thetis is an unstructured mesh based coastal ocean model built using the Firedrake finite element framework. Currently, Thetis consists of 2D depth-averaged and full 3D baroclinic models [35].

   The shallow water flows in this project are described by Shallow Water Equations (SWE). Thetis can be used to model shallow water flows and solve SWEs providing CFD result outputs.

3. Pytorch (1.1.0)

   PyTorch is a Python package that provides two high-level features: Tensor computation (like NumPy) with strong GPU acceleration; Deep neural networks built on a tape-based auto-grad system [36].

   It is a widely used Python-based machine learning framework. The machine learning aspects of this work are based on PyTorch.

4. ParticleModule (2019)

   A python module for driving Lagrangian particles via vtk formatted velocity data [34].

   The particle module is used to generate PIV-like data in this project.

5. SciPy (v1.3.0)

   SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering [37].

6. NumPy (v1.16.4)

   NumPy is the fundamental package for scientific computing with Python. It contains among other things: a powerful N-dimensional array object; sophisticated (broadcasting) functions; tools for integrating C/C++ and Fortran code; useful linear algebra, Fourier transform, and random number capabilities [38].

7. PIVlab (2.02)

   PIVlab is a time-resolved particle image velocimetry (PIV) piece of software that is updated regularly with software fixes and new features. It does not only calculate the velocity distribution within particle image pairs but can also be used to derive, display and export multiple parameters of the flow pattern. A user-friendly graphical user interface (GUI) makes PIV analyses and data post-processing fast and efficient. PIV method would be used as a comparison to our data-driven method.

Link to codes

https://github.com/msc-acse/acse-9-independent-research-project-erizmr

4 Implementation and Code

4.1 Data generation

Appropriate datasets represent one of the most essential factors for training a network with decent performance. Dirty and messy data would prevent the optimization algorithm from finding optimal points for object function. On the hand, a dataset with too many similar samples will cause overfitting problems, which means the network just memorizes the training data and lacks generalization ability. To avoid the two extreme conditions above, a robust data generation strategy is proposed here.

4.1.1 CFD data

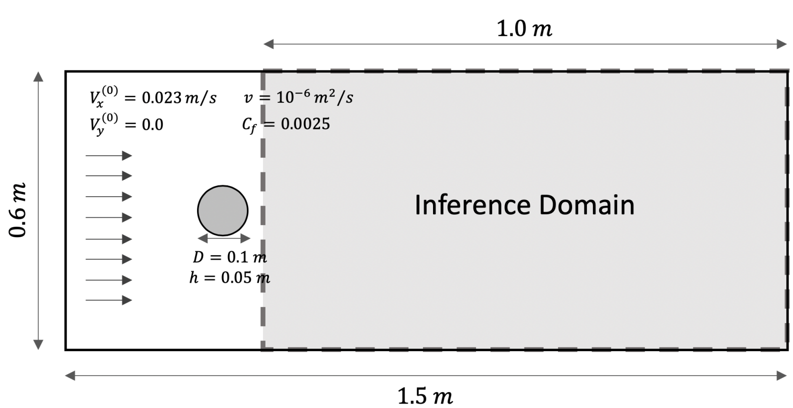
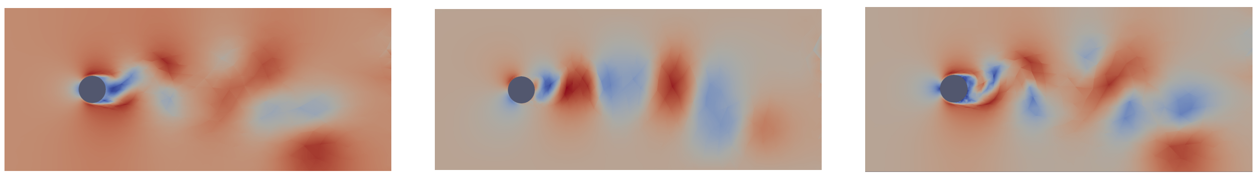


Figure 4: Schematic of experimental set-up.

In the flow past a cylinder test problem, the Reynolds number is the most significant parameter for resulting flow field patterns. In low Reynolds number conditions the flow is laminar, which is not particularly interesting for inference. With a higher Reynolds number, a vortex shedding phenomenon occurs and a so-called von-Karman vortex forms in the downstream of the cylinder. This type of phenomenon is quite common in real-world and industrial applications, which is the main focus of this project.

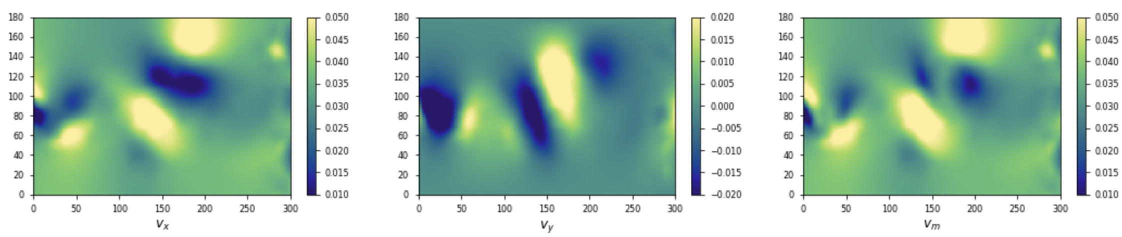
There are two phases for the flow past a cylinder: the developing phase and the stable phase. In the former process, The flow starts to pass and interact with the cylinder and vortices begin to be generated in the wake. Afterward, the wake becomes stable and the vortex street is periodically generated. Due to the periodicity, there will likely be near duplicate snapshots when the flow comes to be stable. These would violate the diversity of the dataset and cause overfitting problem. Therefore, in order to avoid this problem, the simulation is run for only 150 time steps. The stable phase starts at about the 130th step.

The experiment setting is shown in figure 4. Due to our main focus is the wake of the cylinder, to reduce the computation cost, the inference domain is selected as the downstream area of the whole domain, which is the shaded part in figure 4.

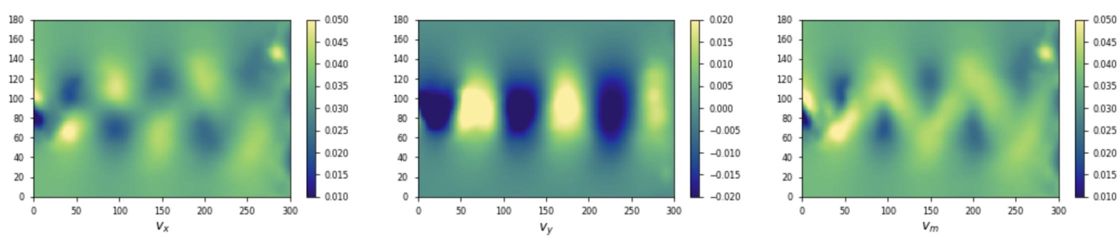


(a) Velocity-x (b) Velocity-y (c) Velocity-Magnitude

Figure 5: Full velocity field components and magnitude, visualised using Paraview.



(a) Developing Phase



(b) Periodic Phase

Figure 6: Samples of CFD result inference domain (velocity in the x and y directions velocity magnitude respectively).

4.1.2 Particle data

For a particle image, there are two key parameters controlling the image quality: particle density and particle diameter. In an image with proper particle density, particles could represent the streamline well. Too large or too small a density means that images lose effective information, so is the particle diameter.

It has been mentioned before, the PIV data is generated by two steps: 1. modelled by Lagrange particles using Particle Module; 2. converted to PIV images by Algorithm 1. The main parameters in the first step is to insert rate. The insert rate decides the number of particles inserted to the domain per time step, which finally control the particle density. In the second step, the key parameter particle diameter is one of the input arguments, which could be specified.

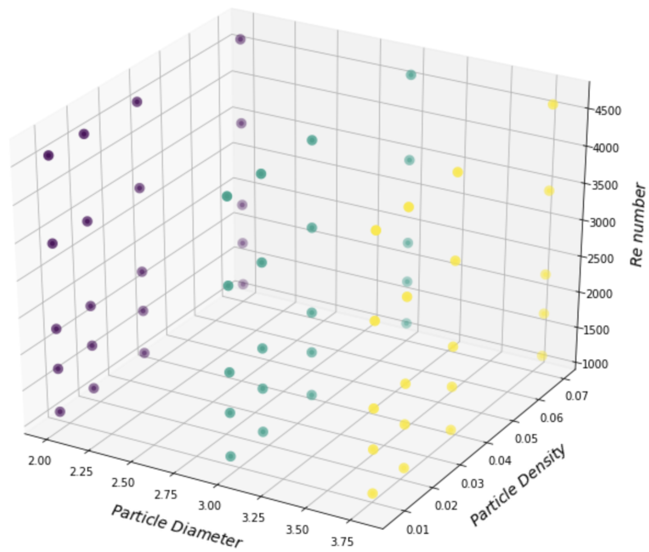


Figure 7: Data Generation Strategy. Each dot in the plot represent a unit set comes with different parameters generated from a simulation.

To construct a robust dataset which could well simulate the real particle image. Three parameters mentioned above are selected: Reynolds number (Re), particle density and particle diameter. The table 1 shows the value range of each parameter and figure 7 shows how they distributed in parameter space. Each dot in the figure represents a whole simulation. Therefore, 60 whole shallow water flow past cylinder simulations are run.

Parameter	**Value Range**
Reynolds number	[1150, 1750, 2300, 3450, 4600]
Particle density	[0.007, 0.018, 0.035, 0.07]
Particle diameter	[2.00, 3.00, 3.80]

Table 1: Value range of each parameter.

4.2 Data Property Analysis

There is a basic assumption for training a decent neural network in deep learning. It requires that the training data and test data comes from the similar probability distribution, or more mathematically the Kullback–Leibler divergence between them is small.

Dimension reduction and data distribution The raw image has a resolution 180×300, with quite high dimension features. It is difficult to compare two distribution in such a high dimension space. One feasible method is to use dimension reduction for the data and then make the comparison. Principle Components Decomposition (PCA) is a method for dimension reduction. It is an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on [39]. Figure 8 shows the results of reducing training, test and lab data to 2 dimension. It could be observed that the distribution of training and test data are quite close. There are some noises in the lab data, which indicates the components with extremely large values. Except for these, the lab data also shows a similar distribution to training data.

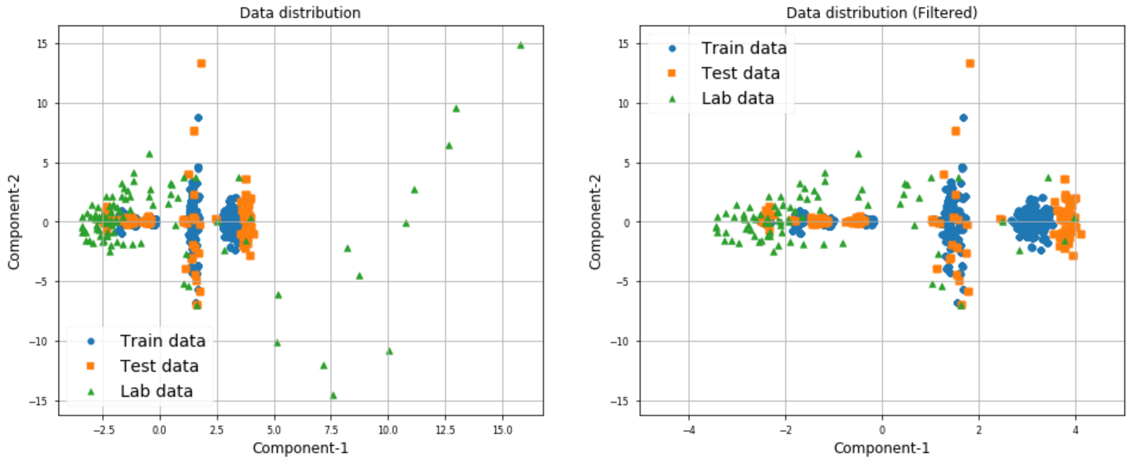


Figure 8: Data distribution after reducing to 2 dimensions. The left one is the raw distribution. The right one shows the distribution of low pass filtered data.

4.3 Neural Network Architecture

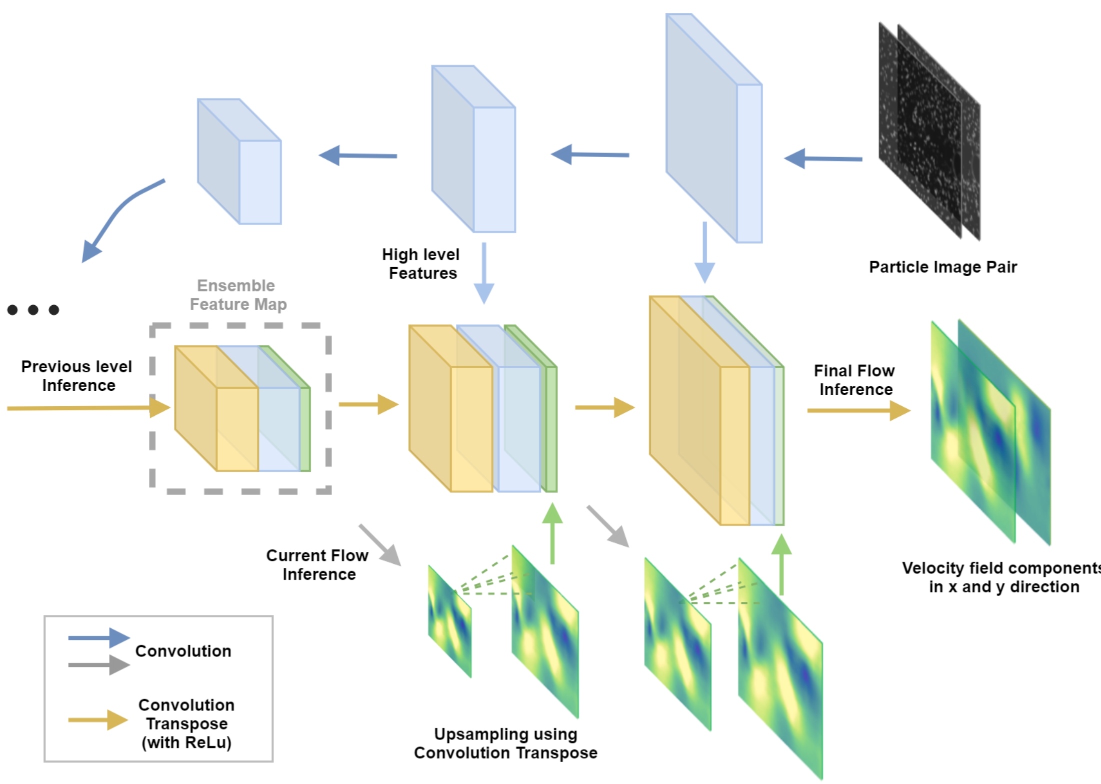


Figure 9: Details of the extraction and inference net architecture. The figure only shows the last three levels of inference net, and the first three levels of extraction net due to room limitation, the full network details could be found in the appendix. The arrows with dark blue, grey represent convolution manipulation and the ones with yellow colour indicates convolution transpose with activation function ReLU.

4.3.1 Particle-Net

In this project, a fluid flow field inference convolutional neural network is developed. The network is able to extract the features from particle images then to predict the flow field. Thus the network is named Particle-

Net. The Particle-Net has two main sub-nets: the extraction net and the inference net. The extraction net consists of a stack of convolution layers for pyramidal feature extraction. The detail of pyramidal feature extraction will be introduced in detail below. The inference net is used to do the flow inference using an ensemble feature. The ensemble feature structure is inspired by Fischer et al.'s work [31] on optical flow.

Pyramidal feature extraction The general feature extraction using CNN is a process that the high dimension features are extracted by convolution and pooling layers and finally stored in a low-level feature map. By using pyramidal feature extraction, the CNN extracts the multi-scale features and store them in a pyramidal feature map. The layers from bottom to the top match the features from highest level to lowest level. The high level means less amount but high resolution features in contrast to the low level. These features will be fed to the inference net as a part of the ensemble feature map in the same level for flow inference.

Inference Net In the inference net, the workflow is shown in figure 9. The different layer of inference net is called inference level here. The figure only shows the last three levels due to space limitations. The arrows with dark blue, grey represent convolution manipulation and the ones with yellow colour are convolution transpose with ReLU manipulations. The ensemble feature map is consists of three components, which are yellow, blue and green blocks in the figure. The yellow one represents the previous level inference result. The result is calculated using convolution transpose with ReLu and ensemble feature map in the previous level. The blue one is the same level feature map fed from the extraction net as mentioned above. The same level here means the fed feature map has the same resolution. As for the green part, this component is firstly calculated by the previous level ensemble feature map using convolution. Then the meta result is upsampling using a convolution transpose without nonlinear activation function, which is different from the yellow part. As the figure shown, the resolution of the inference flow field result will be two times larger than the previous level. The number of convolution levels in the extraction net is the same as the inference levels. Finally, the output inference result has the same resolution as the input image.

4.3.2 Reduced-layer Particle-Net

The deep neural network is notorious for being greedy on computation resources. It usually requires hours for training sometimes even for predicting. To speed up the network, a reduced-layer Particle-Net is proposed.

For image restoration, in addition to the convolution transpose, it is also deserved to try the traditional interpolation methods when the low-level feature map is accurate enough. In the final inference level, the resolution of the image is only two times smaller than the input one, which means it's possible to use some cost cheaper method than convolution transpose, such as interpolation. Here the bi-linear interpolation is taken into consideration. To reduce layer, the last one or two inference levels are substituted by bi-linear interpolations. We named the net with one layer reduced Particle-Net-RL1, and the net with two Particle-Net-RL2.

4.4 Training and validation

4.4.1 Dataset division

Dataset	Re	Density (ppp)	Diameter(pixels) 2.00	3.00	3.80
Training/Validation	1150 2300 4600	0.070	50/10	50/10	50/10
		0.035	50/10	50/10	50/10
		0.018	50/10	50/10	50/10
		0.007	50/10	50/10	50/10
Test	1725	0.084	60	60	60
		0.070	60	60	60
		0.056	60	60	60
		0.035	60	60	60
	3450	0.070	60	60	60
		0.035	60	60	60
		0.018	60	60	60
		0.007	60	60	60

Table 2: Dataset division. The number in each entry of the table represent the samples(image pairs) amount of a unit set. The property of the set is described by three parameters: Re number, particle density, particle diameter, which listed in the head and sidebar. There are 1800 samples in the training set, 360 samples in the validation set and 1440 samples for testing.

For training and validation experiments, the dataset is divided into three parts, training, validation and test set. Due to the Re number is the key factor for flow velocity field distribution, so the unseen data could be defined as the flow field which has a different Re number. The training set consists of many unit sets. The property of each unit set is described by three parameters: Re number, particle density, particle diameter. The value range for them are: Re number $1150, 2300, 4600$; particle density $0.007, 0.018, 0.035, 0.070$; particle diameter $2.00, 3.00, 3.80$. The combination of each value in these three parameters is the number of the unit set. So there are $3 \times 3 \times 4$ unit set with 50 samples in it, 1800 training samples in total. The ratio of training/validation is 5/1, so there are 360 samples for validation. The rest data with Re number 3450 and 1725 is left as the test set, 1440 samples in total.

For dividing training and validation data, there are two widely used sampling methods. The first one is random sampling in the data sequence, the second is chopping the sequence in some point then divide the data into two parts for training and validation. The random-sampling method is suitable for time-independent data, which could prevent the model overfit to some specific part of the data. But in our case, the data is a time series. If the training data is random sampling among the whole sequence, the model would have the chance to memorize almost all phases of the flow process, which might cause the overfitting problem. On the other hand, the second method made the latter part of the sequence unseen data for the model, which makes validation error more similar to test error. Therefore, the second dividing method was applied in this case.

4.4.2 Training strategy

The training parameters are chosen below. There are two widely used optimizers, stochastic gradient descent (SGD) and Adam. The latter has a faster convergence speed than SGD in this case. Thus, the network is trained using Adam optimizer with weight decay (10^{-3}). The network is trained for 12K iterations with a training batch size 32, using a learning rate 10^{-3}. All the training experiments are conducted on Google Colab with its NVIDIA Tesla K80 12GB GPU.

4.4.3 Error definition

Here, two types of error are defined for validation and test, which describe the absolute error and the relative error between predictions and labels respectively. (u^t, v^t) are ground truth velocity, and (u^p, v^p) are velocity components predictions by the neural network. In addition, in order to improve the ability to measure the generalization error, the training error metric is defined as a composite error compatible with the inference net, which is a weighted RMSE summation of each inference level. The w_i are the corresponding weights.

Root mean square error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(u_i^p - u_i^t)^2 - (v_i^p - v_i^t)^2} \tag{2}$$

Relative error

$$RE = \frac{RMSE}{\max(u_i^t, v_i^t)_{i=1..N}} \tag{3}$$

Composite error metric

$$CE = \sum_{i=1}^{N} w_i (RMSE)_i \tag{4}$$

5 Results and Analysis

5.1 Evaluation of synthetic data

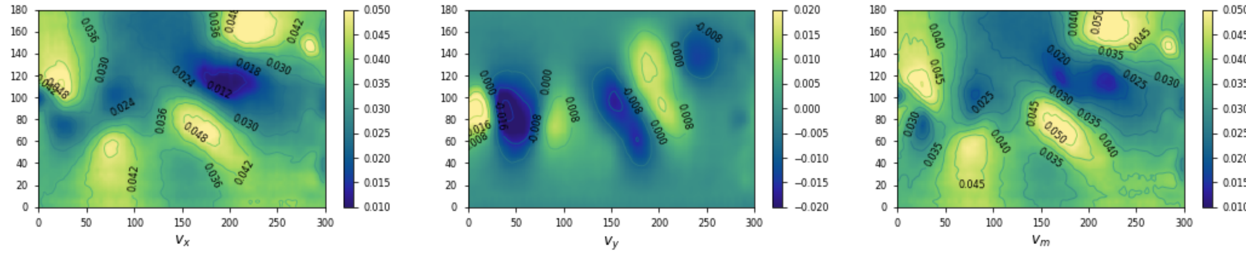
A series of experiments utilising synthetic data are conducted with different particle densities and particle diameters. The results are shown in Table 3. Among the parameters, the choice of particle density is quite an essential one. The particle images with high quality usually have a particle density range from 0.05–0.10. To show the robust ability of the neural network, the inference flow velocity fields of a test case with high-quality particle images (has a density at around 0.07 particle per pixel) and a case with extremely low-quality particle images (has a density at around 0.007 ppp) are shown below. Also, the comparison between full and reduced-layer model, Particle-Net and PIVlab software will also be discussed.

5.1.1 Comparison between full and reduced-layer models

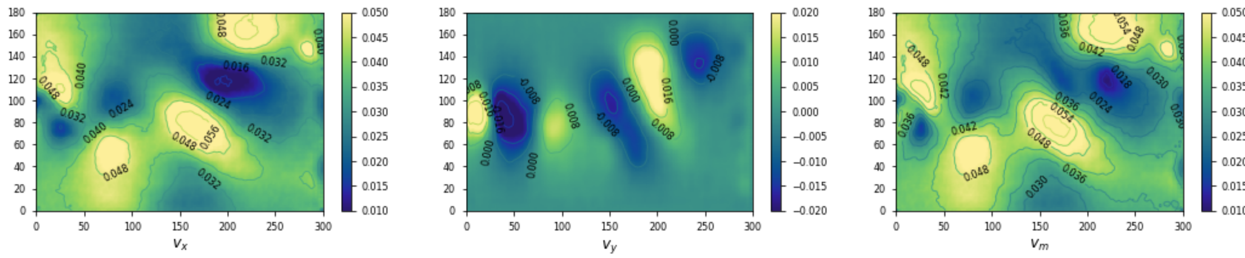
In order to explore the possibility of reducing the volume of the network as well as improving efficiency, two reduced-layer models are proposed in the above section. Here, the comparison experiments between different models are conducted and the root means square error and relative error of all experiments are listed in table 3. As mentioned before, the results of high and low-quality particle images are discussed below.

High-quality particle images(particle density 0.07 ppp)

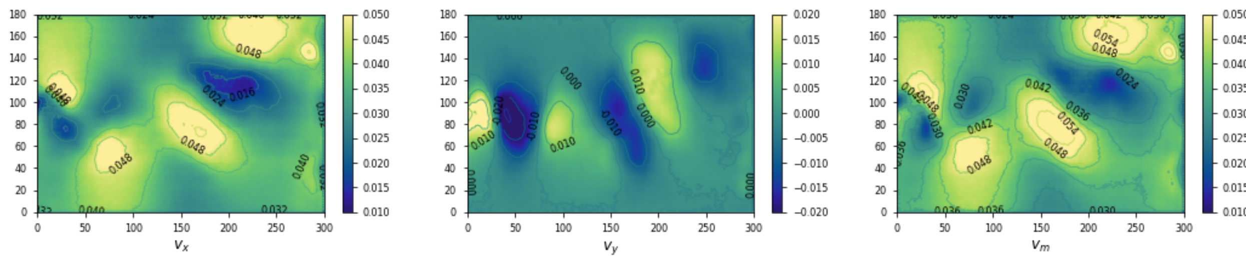
It could be observed that all three networks make predictions with similar flow velocity distribution to the ground truth. The main difference is the value in some regions. The Particle-Net performs the best. Reduced layer models lose more accuracy on velocity magnitude especially in the left-up region in the field.



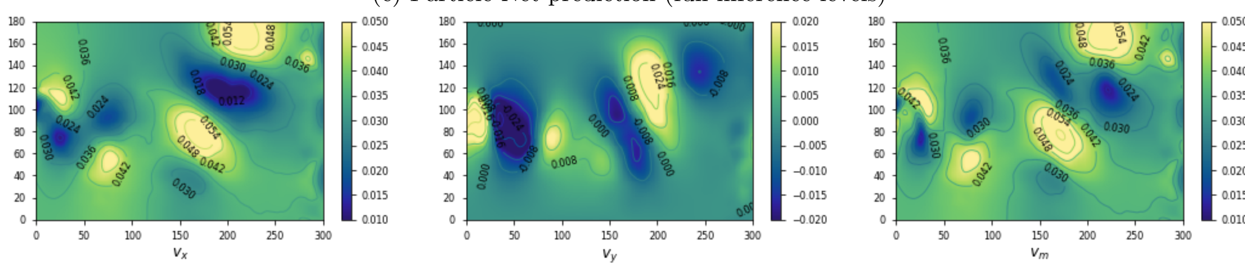
(a) Particle-Net-RL2 prediction (two inference levels reduced)



(b) Particle-Net-RL1 prediction (one inference levels reduced)



(c) Particle-Net prediction (full inference levels)

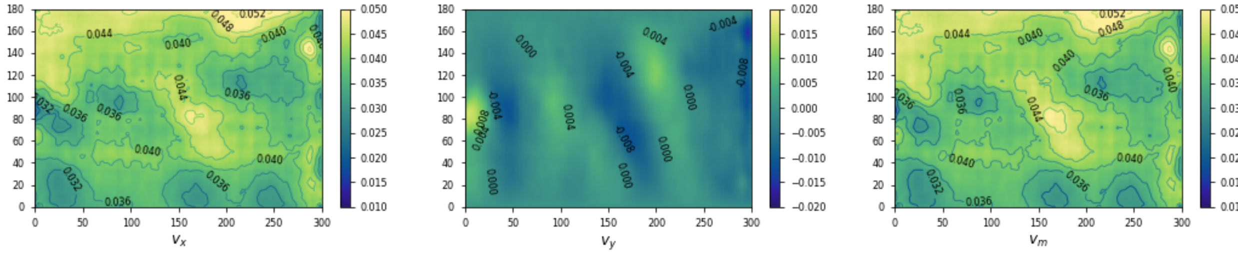


(d) Ground Truth

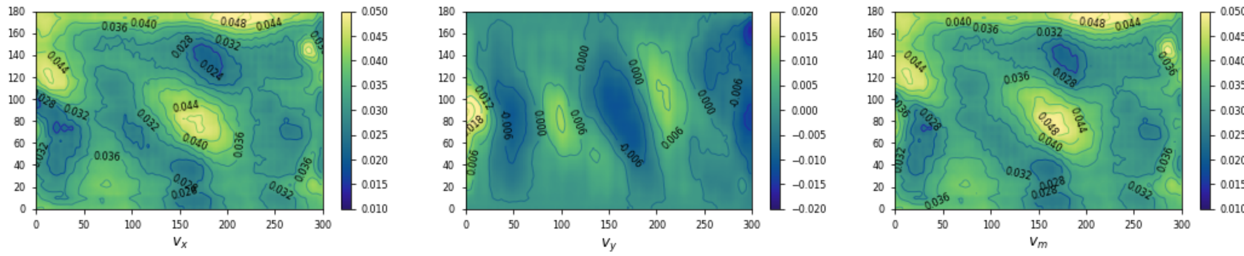
Figure 10: Ground truth versus predictions from full and reduced models (Re 3450, Particle density 0.07 ppp). The columns from left to right represent the velocity in the x and y directions and the magnitude.

Low-quality particle images (particle density 0.007 ppp)

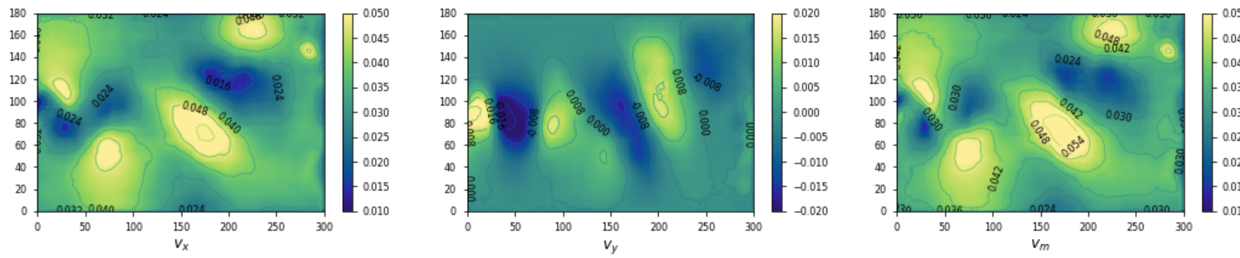
Things are quite different for extreme low particle density case, reduced-layer models perform badly. Not only the velocity magnitude, the reduced-layer models even lose most of the flow structures. On the contrast, the full model still gives a decent prediction.



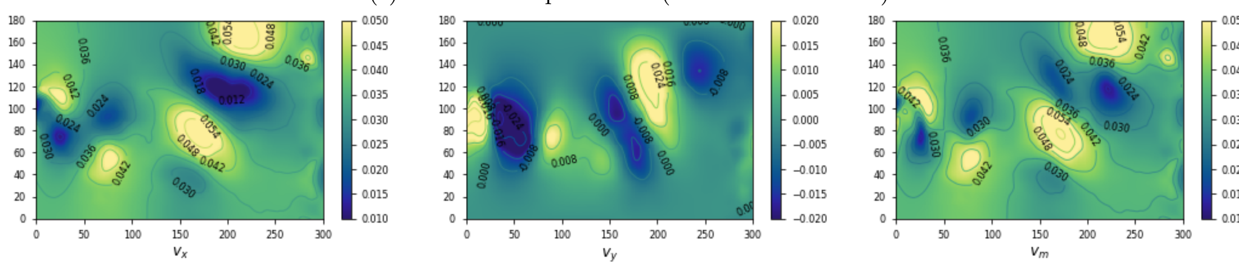
(a) Particle-Net-RL2 prediction (two inference levels reduced)



(b) Particle-Net-RL1 prediction (one inference levels reduced)



(c) Particle-Net prediction (Full inference levels)



(d) Ground Truth

Figure 11: Ground truth versus predictions from full and reduced models (Re 3450, Particle density 0.007 ppp). The columns from left to right represent the velocity in the x and y directions and the magnitude.

In addition to the two cases discussed above, it could be observed that the Particle-Net perform well in both low and high particle density as well as Re number from Table 3. The root means square error and relative error change a little when the density changes. As for the Particle-Net-RL1 and Particle-Net-RL2, they perform almost as great as the Particle-Net in lower Re number 1725. Also, their RMSE and RE are a little bit higher than Particle-Net when processing high-quality particle images (with proper particle density such as 0.07). But when the particle density is extremely low, they perform badly and even get a relative error over 10%. Therefore, the reduced-layer models are suitable for cases with high-quality particle images or handling easier problems (such as the problem with low Re number).

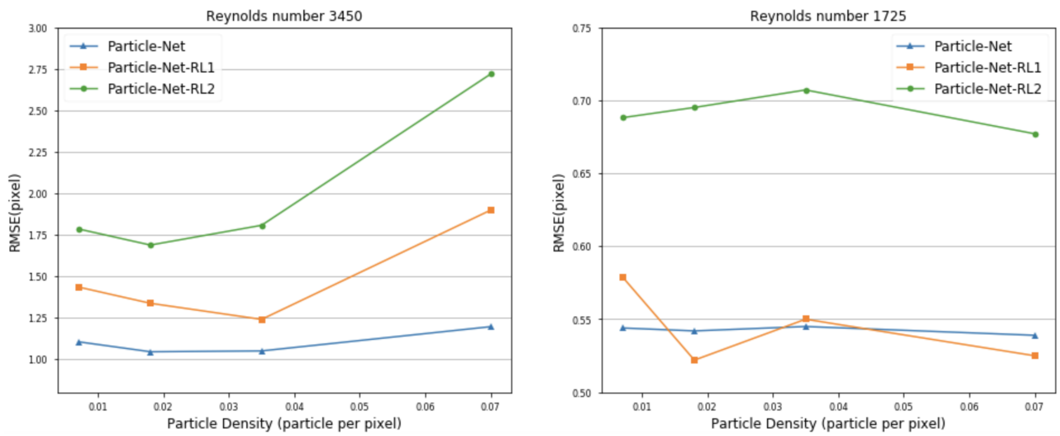


Figure 12: RMSE of full model and reduced-layer model in different particle density. A visualization of some results in Table 3.

Re	Diameter (pixels)	Density (ppp)	Particle-Net		Particle-Net-RL1		Particle-Net-RL2	
			RMSE	RE	RMSE	RE	RMSE	RE
3450	3.00	0.070	1.104	6.67%	1.434	8.67%	1.784	10.77%
		0.035	1.044	6.30%	1.336	8.06%	1.687	10.18%
		0.018	1.049	6.33%	1.239	7.48%	1.806	10.90%
		0.007	1.195	7.21%	1.897	11.55%	2.717	16.41%
1725	3.00	0.084	0.544	4.30%	0.579	4.55%	0.688	5.43%
		0.070	0.542	4.27%	0.522	4.10%	0.695	5.47%
		0.056	0.545	4.31%	0.550	4.31%	0.707	5.58%
		0.035	0.539	4.27%	0.525	4.13%	0.677	5.35%

Table 3: RMSE and relative error in pixel unit with different particle density in the test set.

Re	Diameter (pixels)	Density (ppp)	Particle-Net		Particle-Net-RL1		Particle-Net-RL2	
			RMSE	RE	RMSE	RE	RMSE	RE
3450	0.07	2.00	1.103	6.65%	1.434	8.67%	1.783	10.76%
		3.00	1.104	6.67%	1.434	8.67%	1.784	10.77%
		3.80	1.104	6.66%	1.433	8.66%	1.784	10.76%
1725	0.07	2.00	0.541	4.27%	0.522	4.10%	0.694	5.47%
		3.00	0.542	4.27%	0.522	4.10%	0.695	5.47%
		3.80	0.542	4.27%	0.523	4.10%	0.695	5.47%

Table 4: RMSE in pixel unit with the different particle diameter in the test set

5.1.2 Comparison to PIVlab

PIVlab is a time-resolved particle image velocimetry (PIV) software that is updated regularly with software fixes and new features [40]. The PIVlab version used in this project is 2.02, implemented in MATLAB 2019a. Comparisons are conducted both in terms of accuracy and efficiency. The window deformation iterative multi-grid method (WIDIM) is a state-of-art cross-correlation method used to conduct PIV. The built-in WIDIM in PIVlab is applied in the comparison experiments with four-pass configurations. The four interrogation areas are 32, 24, 12, 6 respectively.

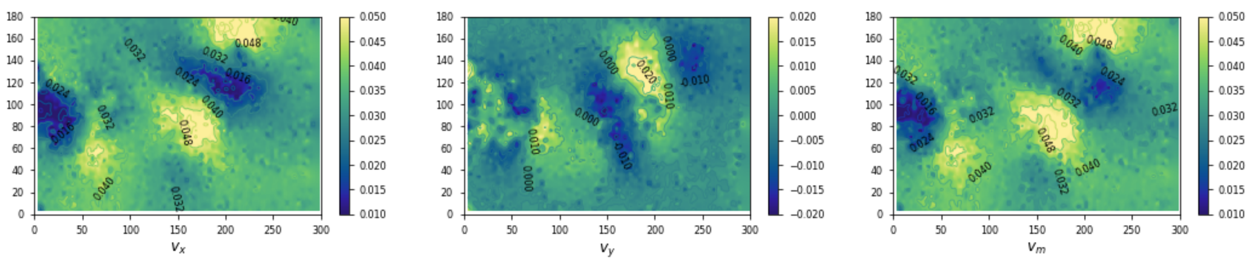
Accuracy

The comparison experiments are similar to the section above, there are two kinds of test cases, one with particle images in high quality (particle density 0.07), the other with low quality (particle density 0.007).

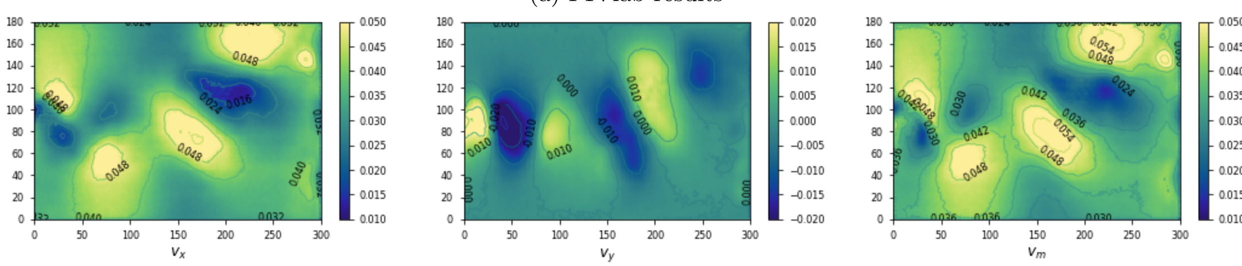
Figure 13 shows the comparison results between PIVlab and network in a high-quality case. It could be observed that the results from PIVlab have lots of noises. Also the PIV method loses more small scale flow features. For the second test case with low quality, the results are shown in figure 14. It could be observed that the PIV technique is not good at dealing with sparse particles images, the results are much worse than the previous case as there are even no clear flow structures. On the contrary, the neural network shows a more robust property. For quantitative comparison, the results are listed in the 5. The relative error of PIVlab is larger than 13%, almost two times that of Particle-Net.

Re	Diameter (pixels)	Density (ppp)	Particle-Net		PIVlab	
			RMSE	RE	RMSE	RE
3450	3.00	0.070	1.104	6.67%	2.200	13.25%
		0.007	1.195	7.21%	3.119	18.18%
1725	3.00	0.070	0.542	4.27%	-	-
		0.007	1.195	7.21%	-	-

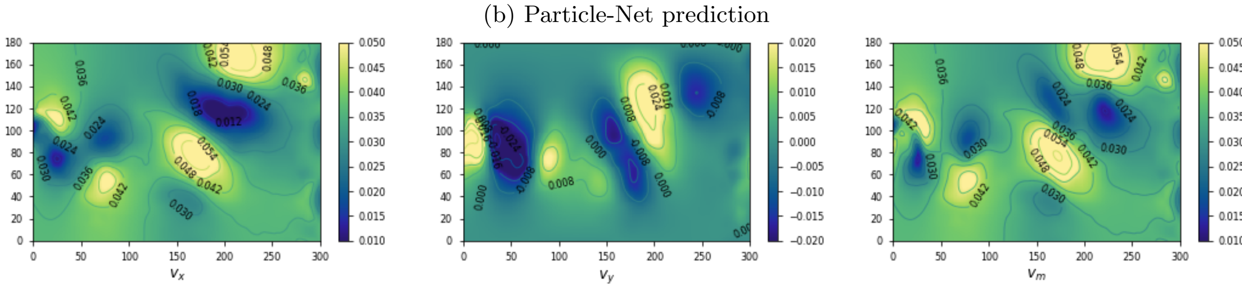
Table 5: RMSE, the relative error of PIVlab and Particle-Net



(a) PIVlab results

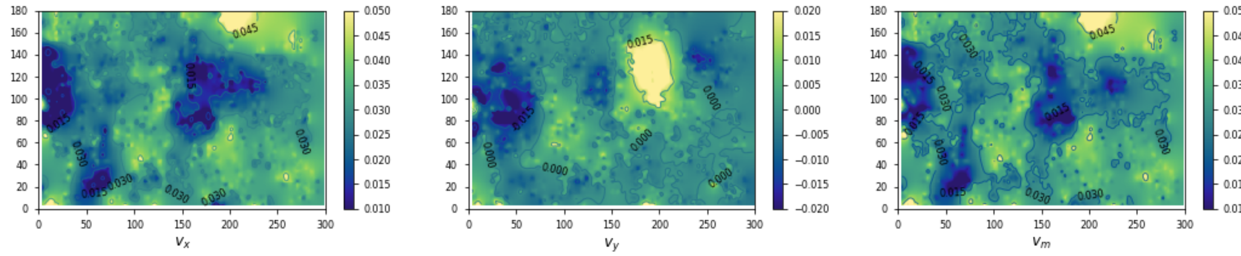


(b) Particle-Net prediction

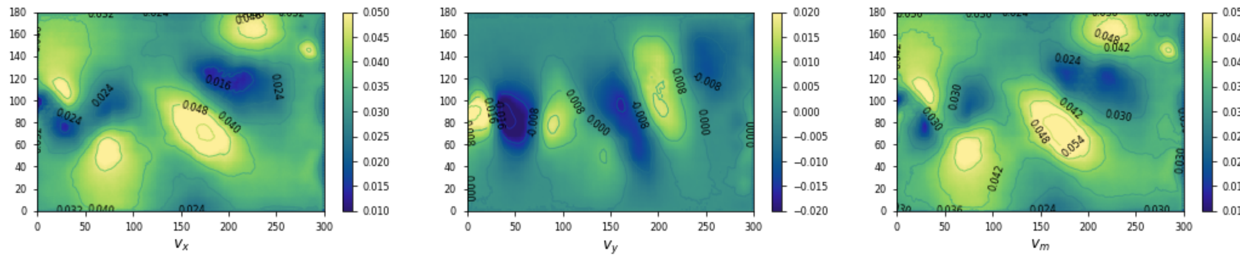


(c) Ground Truth

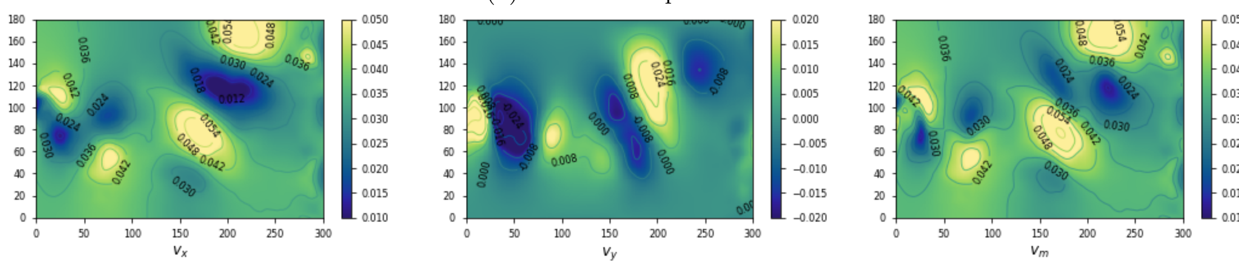
Figure 13: PIVlab results versus neural network predictions(Re 3450, Particle density 0.07 ppp).



(a) PIVlab results



(b) Particle-Net prediction



(c) Ground Truth

Figure 14: PIVlab results versus neural network predictions (Re 3450, Particle density 0.007 ppp).

Efficiency

Efficiency is another important measure for performance comparison. As shown in Table 6, PIVlab require about 101 seconds to 120 pairs with size (300,180). On the other hand, Particle-Net shows a huge advantage in efficiency. It only needs 4.24 seconds to make predictions for the same number of pairs. In addition, the time for loading the model is the majority time cost, which takes 60% of the total time. The model should only be loaded once, then it can be used to make predictions multiple times. With a loaded model, the processing speed of Particle-Net is about 72 frame/s with a resolution 300×180. Generally, the minimum frame number for an animation is 25 fps, which is much lower than the processing speed of Particle-Net. So it is quite promising to do real-time flow field inference using Particle-Net.

	PIVlab WIDIM	Particle-Net	Particle-Net-1	Particle-Net-2
Processor	CPU	GPU	GPU	GPU
Job size	$120 \times 180 \times 300$	$120 \times 180 \times 300$	$120 \times 180 \times 300$	$120 \times 180 \times 300$
Loading Time(s)	-	2.570	2.633	2.620
Calculating Time(s)	101	1.668	1.652	1.653
Total time(s)	101	4.238	4.285	4.245
FPS(frame/s)	0.84	71.94	72.64	72.59

Table 6: Efficiency comparison between Particle-Net and PIVlab.

5.2 Evaluation of lab data

Figure 15: Lab setting(image credit Mr Lucas Mackie and Mr Raul Adriaensen).

In addition to the synthetic data, the trained model is also tested on lab data. The lab data is obtained for a series of experiments which have a similar set-up (shown in figure 15) with the simulation. The particle tracers are small squares of paper around 2mm in width, and the lighting has also been set up so the tracers could be clearly observed in the images. For the camera, a Hero7 camera with 4k resolution was set up at an angle and a 2.7k resolution Hero5 camera was set up vertically. The camera takes photos with an interval 1/60 second. After getting the photo, some processing techniques are required to fix the distortion due to the photograph angles. Finally, the undistorted photos are cropped and sampled to size (180, 300) and used as the inputs for Particle-Net and PIVlab.

The results of Particle-Net and PIVlab are similar (shown in Figure 17), both methods are able to describe the flow past a cylinder from right to left, especially in the wake of the cylinder (dark blue area in the images). The mean velocity profile from cylinder to the downstream and its standard deviation (shown in figure 18) of both methods show a close agreement.

Due to the low resolution of the real lab data, some parts of the image are empty with particles (shown in Figure 16), both the Particle-Net and WIDIM method did perform interpolation in these areas. The interpolation results of the Particle-Net are a little bit higher than WIDIM but with lower noise. The experiment primarily proves that the Particle-Net trained by synthetic data has some generalization ability, which could also adapt to unseen real lab images.

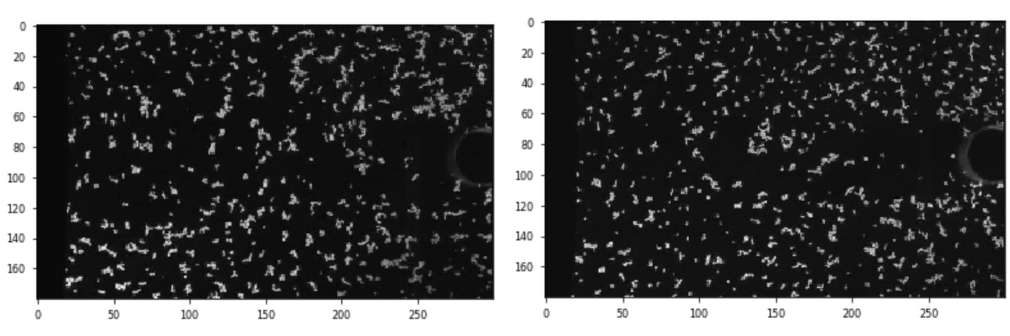
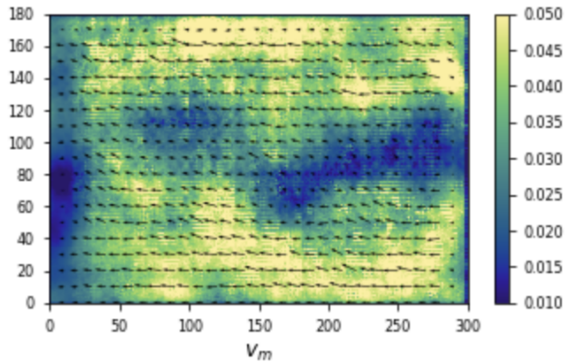
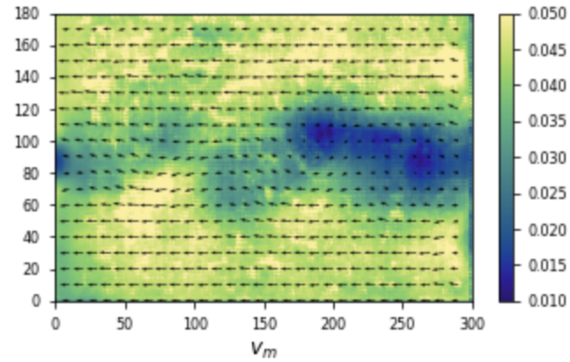


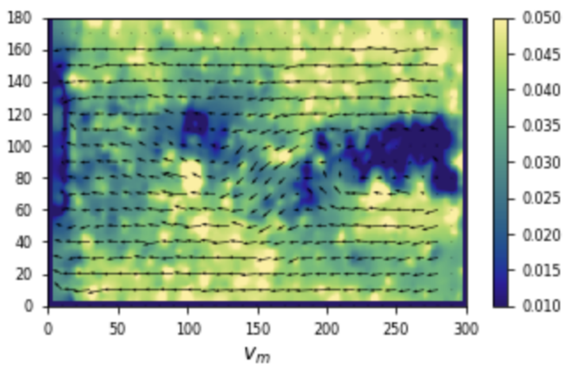
Figure 16: Samples of real lab undistorted particle image.



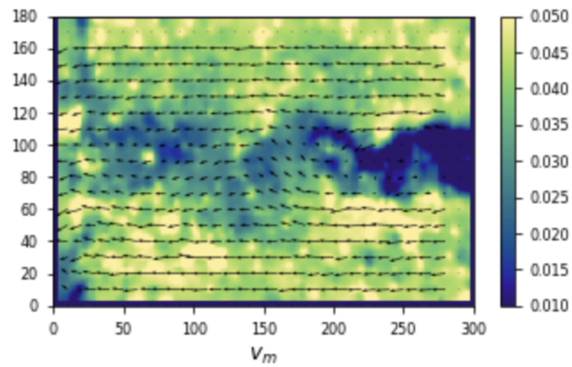
(a) Prediction of Particle-Net



(b) Prediction of Particle-Net



(c) PIVlab result



(d) PIVlab result

Figure 17: PIVlab results versus Particle-Net predictions. The left and right image represents the frames in different time levels. The fluid flows from right to left, which is opposite to training data and the Particle-Net can still make the right predictions.

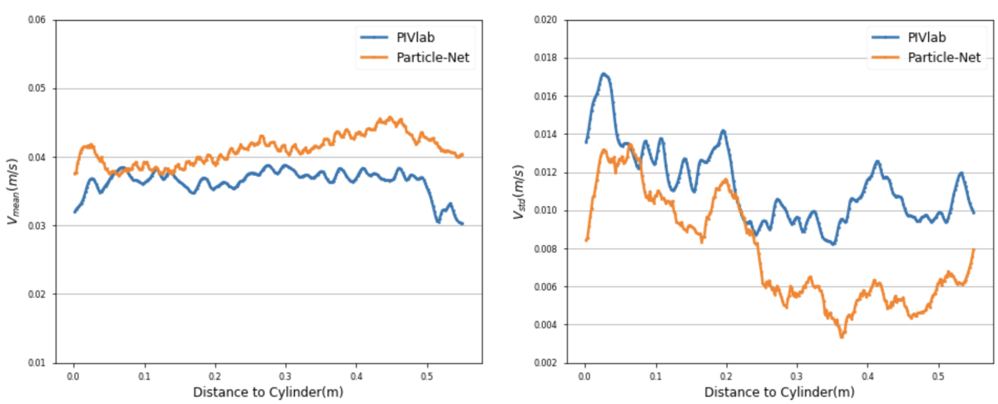


Figure 18: Mean and std velocity in different distances to the cylinder.

6 Conclusion and Discussion

6.1 Conclusion

A network for end-to-end fluid flow inference called Particle-Net has been developed in this project. Particle-Net is able to infer the flow velocity field for shallow water problems from particle images. Particle-Net is shown to outperform state-of-art particle image velocimetry software PIVlab (WIDIM method) for synthetically generated data, both in terms of accuracy and efficiency, especially when considering extremely low-quality particle images. Particle-Net was trained only using synthetic data, but still shows promising capability to analyse real lab data even with sparse particles, although it does not outperform the WIDIM on all lab data.

Particle-Net is also able to process images about 25 times faster than PIVlab (WIDIM method) and is able to process nearly 72 pairs of images per second, which shows promise for real-time fluid flow inference.

6.2 Future work

Currently Particle-Net is only trained using a relatively small volume (1800 pairs) of synthetic data due to the Colab GPU memory limitation. It is believed that a larger volume and a more diverse dataset would improve the inference ability of the Particle-Net on real-world particle images. On the other hand, Particle-Net current has 10 layers in the extraction sub-net and 6 inference levels in the inference sub-net, which represents quite a deep network. It will be promising to make it more compact with fewer layers but the same accuracy. There is only one net for two images in the pair for the feature extraction process. It is deserved to explore using a separate extraction net for each image. It is also worth combining with traditional robust approaches such as the variational method to enhance the extracted features. A further developed Particle-Net is expected to be applicable to many fluid flow related scenarios such as coastal flow inference from unmanned aerial vehicle photos, real-time coastal hazards monitoring, etc.

References

[1] R. du Feu, S. Funke, S. Kramer, J. Hill, and M. Piggott, "The trade-off between tidal-turbine array yield and environmental impact: a maximum entropy modelling approach," *Renewable Energy*, pp. 390–403, 2019.

[2] W. Pan, S. C. Kramer, and M. D. Piggott, "Multi-layer non-hydrostatic free surface modelling using the discontinuous galerkin method," *Ocean Modelling*, vol. 134, pp. 68–83, 2019.

[3] J. Rabault, J. Kolaas, and A. Jensen, "Performing particle image velocimetry using artificial neural networks: a proof-of-concept," *Measurement Science and Technology*, vol. 28, p. 125301, 2017.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[5] A. Graves, A. rahman Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," *IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649, 2013.

[6] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Google*, 2014.

[7] Y.LeCun, Y.Bengio, and G.Hinton, "Deep learning," *Nature*, vol. 521, p. 436–444, 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14539

[8] I.Goodfellow, Y.Bengio, and A.Courville, *Deep learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[9] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," *arXiv:physics*, 2019.

[10] N. Erichson, L. Mathelin, Z. Yao, S. Brunton, M. Mahoney, and J. Nathan Kutz, "Shallow learning for fluid flow reconstruction with limited sensors and limited data," *arXiv preprint*, vol. arXiv:1902.07358, 2019.

[11] T. P. Miyanawalaa and R. K. Jaimana, "An efficient deep learning technique for the navier-stokes equations: Application to unsteady wake flow dynamics," *arXiv preprint*, vol. arXiv:1710.09099, 2018.

[12] B. Kim and T. Günther, "Robust reference frame extraction from unsteady 2d vector fields with convolutional neural networks," *Eurographics Conference on Visualization (EuroVis) 2019*, vol. 38, p. Number 3, 2019.

[13] S. Lee and D. You, "Data-driven prediction of unsteady flow over a circular cylinder using deep learning," *arXiv preprint*, vol. arXiv:1804.06076, 2018.

[14] T. Bolton and L. Zanna, "Applications of deep learning to ocean data inference and subgrid parameterization," *Journal of Advances in Modeling Earth Systems*, vol. 11, pp. 376–399, 2019.

[15] K. Franz, R. Roscher, A. Milioto, S. Wenzel, and J. Kusche, "Ocean eddy identification and tracking using neural networks," *Poster at IEEE International Geoscience and Remote Sensing Symposium*, 2018.

[16] Z.Wang, D. Xiao, F. Fang, R. Govindan, C. C. Pain, and Y. Guo, "Model identification of reduced order fluid dynamics systems using deep learning," *Int. J. Numer. Methods Fluids*, vol. 86(4), p. 255–268, 2017.

[17] R. Gurka, A. Liberzon, and G. Hetsroni, "Pod of vorticity fields: A method for spatial characterization of coherent structures," *International Journal of Heat and Fluid Flow*, vol. 27, p. 416–423, 2006.

[18] D. Xiao, C. Heaney, L. Mottet, F. Fang, W. Lin, I. Navon, Y. Guo, O. Matar, A. Robins, and C. Pain, "A reduced order model for turbulent flows in the urban environment using machine learning," *Build. Environ.*, vol. 148, p. 323–337, 2019.

[19] Z. Zhang, X. dong Song, S. ran Ye, Y. wei Wang, C. guang Huang, and Y. ran An andYao-song Chen, "Application of deep learning method to reynolds stress models of channel flow based on reduced-order modeling of dns data," *Journal of Hydrodynamics*, vol. 31(1), pp. 58–75, 2019.

[20] R. King, O. Hennigh, A. Mohan, and M. Chertkov, "From deep to physics-informed learning of turbulence: Diagnostics," *arXiv preprint*, vol. arXiv:1810.07785, 2018.

[21] E. J. Parish and K. Duraisamy, "A paradigm for data-driven predictive model- ing using field inversion and machine learning," *Journal of Computational Physics*, vol. 305, pp. 758–774, 2016.

[22] J. N. Kutz, "Deep learning in fluid dynamics," *Journal of Fluid Mechanics*, vol. 814, pp. 1–4, 2017.

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

[24] J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," *Journal of Fluid Mechanics*, vol. 807, pp. 155–166, 2016.

[25] N. B. Erichson, M. Muehlebach, and M. W. Mahoney, "Physics-informed autoencoders for lyapunov-stable fluid flow prediction," *arXiv preprint*, vol. arXiv:905.10866, 2019.

[26] R. Jean, K. Miroslav, and J. Atle, "Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control," *arXiv preprint*, vol. arXiv:1808.07664, 2018.

[27] J. Morton, A. Jameson, M. J. Kochenderfer, and F. Witherden, "Deep dynamical modeling and control of unsteady fluid flows," *Advances in Neural Information Processing Systems*, p. 9258–9268, 2018.

[28] D. H. Warren and E. R. Strelow, *Electronic Spatial Sensing for the Blind: Contributions from Perception.* Springer, 1985.

[29] B. Horn and B. Schunck, "Determining optical flow," *Arifical Intelligence*, vol. 17, p. 185–203, 1981.

[30] T. Brox and J. Mailk, "Large displacement optical flow: Descriptor matching in variational motion estimation," *PAMI*, vol. 33, pp. 500–513, 2011.

[31] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional net- works," *ICCV*, p. 2758–2766, 2015.

[32] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet2.0: Evolution of optical flow estimation with deep networks," *CVPR*, p. 2462–2470, 2017.

[33] Firedrake-Community. (2019) Firedrake. [Online]. Available: https://www.firedrakeproject.org/

[34] J. Percival. (2019) Particle module. [Online]. Available: https://github.com/jrper/ParticleModule

[35] Thetis-Community. (2019) Thetis. [Online]. Available: https://thetisproject.org/

[36] PyTorch-Community. (2019) Pytorch. [Online]. Available: https://pytorch.org

[37] SciPy-Community. (2019) Scipy. [Online]. Available: https://www.scipy.org

[38] Numpy-Community. (2019) Numpy. [Online]. Available: https://www.numpy.org

[39] J. I.T., *Principal Component Analysis 2nd ed.* Springer, 2002.

[40] W. Thielicke and E. J. Stamhuis, "PIVlab:towards user-friendly, affordable and accurate digital particle image velocimetry in MATLAB," *Journal of Open Research Software*, vol. 2, oct 2014. [Online]. Available: https://doi.org/10.5334%2Fjors.bl