Predicting Motion of Engine-Ingested Particles Using Deep Neural Networks

Travis L. Bowman

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Mechanical Engineering

John A. Palmore Jr., Chair
Kevin T. Lowe
Danesh K. Tafti

July 7, 2022
Blacksburg, Virginia

Keywords: machine learning, artificial neural networks, multiphase flow, particle-laden flow

Predicting Motion of Engine-Ingested Particles Using Deep Neural Networks

Travis L. Bowman

The ultimate goal of this work is to facilitate the design of gas turbine engine particle separators by reducing the computational expense to accurately simulate the fluid flow and particle motion inside the separator. It has been well-documented that particle ingestion yields many detrimental impacts for gas turbine engines. The consequences of ice particle ingestion can range from surface-wear abrasion to engine power loss. It is known that sufficiently small particles, characterized by small particle response times (τ_p), closely follow the fluid trajectory whereas large particles deviate from the streamlines. Rather than manually deriving how the particle acceleration varies from the fluid acceleration, this work chooses to implicitly derive this relationship using machine learning (ML). Inertial particle separators are devices designed to remove particles from the engine intake flow, which contributes to both elongating the lifespan and promoting safer operation of aviation gas turbine engines. Complex flows, such as flow through a particle separator, naturally have rotation and strain present throughout the flow field. This study attempts to understand if the motion of particles within rotational and strained canonical flows can be accurately predicted using supervised ML. This report suggests that preprocessing the ML training data to the fluid streamline coordinates can improve model training. ML models were developed for predicting particle acceleration in laminar, fully rotational/irrotational flows and combined laminar flows with rotation and strain. Lastly, the ML model is applied to particle data extracted from a Computational Fluid Dynamics (CFD) study of particle-laden flow around a louver-geometry. However, the model trained with particle data from combined canonical flows fails to accurately predict particle accelerations in the CFD flow field.

Predicting Motion of Engine-Ingested Particles Using Deep Neural Networks

Travis L. Bowman

General Audience Abstract

Aviation gas turbine engine particle ingestion is known to reduce engine lifespans and even pose a threat to safe operation in the worst case. Particles being ingested into an engine can be modeled using multiphase flow techniques. Devices called inertial particle separators are designed to remove particles from the flow into the engine. One challenge with designing such a separator is figuring out how to efficiently expel the small particles from the flow while not unnecessarily increasing pressure loss with excessive twists and turns in the geometry. Designers usually have to develop such geometries using multiphase flow computational fluid dynamics (CFD) that solve the fluid and particle dynamics. The abundance of data associated with CFD, and especially multiphase flows make it an ideal application to study with machine learning (ML). Because such multiphase simulations are very computationally expensive, it is desirable to develop "cheaper" methods. This is the long term goal of this work; we want to create ML surrogates that decrease the computational cost of simulating the particle and fluid flow in particle separator geometries such that designs can be iterated more quickly. In this work we introduce how artificial neural networks (ANNs), which are a tool used in ML, can be used to predict particle acceleration in fluid flow. The ANNs are shown to learn the acceleration predictions with acceptable accuracy for the training data generated with canonical flow cases. However, the ML model struggles to become generalizable to actual CFD simulations.

Dedication

To Vanessa, who has cheered me on when I've needed it the most and gently reminded me that the remedy to writer's block is to just write.

Acknowledgments

I would like to thank my advisor, Dr. John Palmore Jr., for his support, patience, and kind mentorship throughout the research process. Thank you for igniting my interest in the wide world of fluid dynamics during my studies here at Virginia Tech. I am much obliged for your guidance in growing as an engineer and researcher. I would also like to thank Cairen Miranda for his guidance in running and debugging many simulations. This work would not have been possible without your generous help.

Many thanks are due to the Virginia Space Grant Consortium and the Sloan Scholars Mentoring Network for their funding and support of this work. This work would also not have been possible without the computational resources and gracious technical support received from Advanced Research Computing at Virginia Tech. URL: `https://arc.vt.edu/`

These acknowledgments would be sorely incomplete without thanking all of the professors, mentors, and peers who have pushed and challenged my thinking. While there are certainly too many to list, you know who you are, and I thank you for investing in me.

Most importantly, I could not possibly be where I am today, professionally or personally, without the undying encouragement of my family. For their love and instillment of faith, I am forever grateful.

Contents

List of Figures

List of Tables

Nomenclature

$\nabla \mathbf{u}$	Fluid velocity gradient tensor
$\mathbf{S}$	Strain rate tensor
A	Strain rate
c_D	Drag coefficient
d_p	Particle diameter
e	Shear rate
lr	Learning rate
Re	Reynolds number
St	Stokes number
$\mathbf{a} = \frac{d\mathbf{v}}{dt}$	Particle acceleration
$\mathbf{a}_{NN}$	Neural network-predicted particle acceleration
$\mathbf{e} = \mathbf{a} - \mathbf{a}_{NN}$	Error vector
$\mathbf{u}$	Fluid velocity
$\mathbf{v}$	Particle velocity
$\mathbf{v}_r = \mathbf{u} - \mathbf{v}$	Relative particle velocity
$\mathbf{x}_p$	Particle position
ANN	Artificial Neural Network
$\mathrm{E} = \frac{\|\|\mathbf{a}_{NN}\|\| - \|\|\mathbf{a}\|\|}{\|\|\mathbf{a}\|\|}$	Normalized magnitude error

H Height of computational domain

HL Hidden layer

ML Machine Learning

N Sample size

Greek letters

β Angle between $\mathbf{a}$ and $\mathbf{a}_{NN}$

$\boldsymbol{\epsilon} = \frac{\mathbf{e}}{||\mathbf{a}||}$ Normalized error vector

$\boldsymbol{\Omega}$ Rotation tensor

$\epsilon = ||\boldsymbol{\epsilon}||$ Magnitude of normalized error vector

μ Momentum parameter

μ_g Dynamic viscosity

ν Kinematic viscosity

ω Rotation rate

Φ Particle volume fraction

ρ Density

$\sigma()$ Activation function

τ_f Flow timescale

τ_p Particle response time

Subscripts

g Gas-phase

in Inlet

n Normal

NN Neural network-predicted quantity

p Particle

s Stream-wise

Chapter 1

Introduction

1.1 Atmospheric Ice

Ice has plagued aircraft operation since the birth of aviation. Ice accretion imposes the immediate threat of accumulated ice changing the aerodynamic properties of the wing airfoil, in the worst case leading to loss of sufficient lift. The majority of current research deals with ice accretion and means to mitigate or prevent this phenomena [2, 3, 4]. In this work, however, the focus is turned to ice that has been ingested by the gas turbine engine through means of either ice existing in the atmosphere or pieces of accretion that have been shed. Ice can enter the engine in two ways. Large pieces of ice accretion can break off aircraft surfaces and be ingested. Alternatively, existing ice in the atmosphere such as ice crystals or hailstones can be ingested into the engine. Of these two atmospheric ices, hailstones are of much more concern to safe operation of the engine as they typically range from 0.5 cm to 3.0 cm in diameter [5]. It has been well-documented that particle ingestion can result in many detrimental impacts for gas turbine engines [6]. For ice particles, the consequences can range from surface-wear abrasion to loss of power. Ice that passes through the fan stage enters the compressor, where it likely melts and can evaporate. This formation of liquid water or vapor alters the properties of the flow from the engine's design point. As the compressor is very sensitive to such changes, surge can occur as a result [7]. While it is rare for particles to reach the combustion chamber, in the case that enough ice particulate does, the engine can experience flame-out [3, 7].

While hailstones are typically a larger threat to aviation gas turbine engine operation, the impact of ice crystal ingestion cannot be ignored. Commercial pilots have reported ice crystal icing events while flying in or near deep convective clouds. Excessive mass ingestion of ice crystals has been identified as a likely reason behind engine flame-out, or power loss, events [8]. Convective clouds are formed by convective systems (rapidly rising air currents), such as cumulus or cumulonimbus (anvil-shaped) clouds. These clouds sometimes occur

in organized structures that have both cumulonimbus and nimbostratus clouds (a type of stratiform cloud), and form what are called mesoscale convective systems (MCS) [9]. The term mesoscale typically refers to weather systems spanning 10's to 100's of kilometers. The cold region at the top of the MCS is readily identified with infrared satellite imagery [9], and is thus an ideal host for high ice water content (IWC) regions. The size of ice crystals that constitute such regions is of critical importance to study the motion of such particles. Using an aircraft equipped with instrumentation such as a cloud droplet probe and a precipitation imaging probe, [8] studied the range of ice crystals sizes in high IWC regions (defined as IWC $> 1.5g/m^3$) within MCSs. They found that the median mass diameters of such ice crystals typically ranged from 250 - 500 μm. However, two of the flight missions conducted in long lasting tropical storms suggested that high IWC could also be associated with median mass diameters ranging from 400 - 800 μm but with peak values high as 2 mm. In addition to engine power loss, ice crystals also can prove hazardous when ingested into air probes such as pitot tubes or total air temperature sensors [10]. This can cause inaccurate readings of temperature, air speed, and angle of attack. Using anomalies in the total air temperature sensors (as they have been reported to read 0°C in some icing events), [10] developed an early warning ice detection method to warn pilots of potential hazard.

1.1.1 Hailstone Formation

Hailstones are formed in strong convective updrafts, typically within MCSs formed by cumulonimbus clouds; however, single cumulonimbus systems are also capable of producing hail. Small hailstones that form with a diameter of less than 0.5 cm are typically referred to as graupel. These large weather systems pull water droplets further up in the atmosphere and once sufficient altitude is reached, some droplets will freeze. The hailstones grow when water droplets collide with and coalesce with the frozen particles. This growth is typically maximized in the case that the collected droplets are supercooled, which is often the case in strong updrafts [11]. Depending on the local air temperature, the collected droplets may freeze very quickly in which case the frozen layer will appear cloudy since the air bubbles would not have had time to escape. This is appropriately called dry growth. If the hailstone embryo experiences what is known as wet growth, the collected water drops do not immediately freeze; rather, they extend across the hailstone surface and freeze slowly. Wet growth results in a layer of clear ice since any air bubbles have sufficient time to escape the liquid [12]. These two types of freezing as the hailstone moves both vertically and horizontally throughout the weather system is what gives hailstones their layered appearance. In fact, one of the lesser understood hail production mechanisms is how a hailstone embryo can remain in a region with adequate liquid water content for long enough to grow into a full sized hailstone [13]. The study of this mechanism in a multicellular (that is, more than one thunderstorm cloud) hailstorm by [13] showed that when confined to the the main convective updraft of the system, hail production was at a minimum. Their proposed model suggests that hailstone embryos originate in feeder cells and are subsequently strewn into the main

updraft region where they can continue to grow. However, their results suggest that this may be storm dependent.

1.1.2 Aviation Gas Turbine Engine Interaction

As previously mentioned, hailstones are usually regarded as a more imminent threat to gas turbine engine operation than ice crystals. For this reason, hail ingestion tests must be completed before aviation gas turbines can be certified. The certification tests for high-bypass commercial engines require safe hail ingestion for 30 seconds at a concentration of 10g/m^3 [5].

To better understand how ice particles move throughout the engine, many experimental tests have been conducted to determine how hailstones disintegrate after high speed impact with a solid surface. The key parameters that have been studied are the particle size distributions after impact and the post impact kinetics. Studies with hailstones from 1.25 - 5.0 cm (0.5 - 2.0 in) impacting a solid surface at velocities from 75 - 275 m/s at angles from 0 - 90° suggest that the size distribution of disintegrated ice can be well approximated with the Rosin-Rammler distribution function, which is typically used to define spray droplet distribution [5]. Results presented by [7] agree with this and also assert that the post-impact particle size are functions of the normal component of impact velocity. The disintegrated hailstone particles rebound in an inelastic manner, where [5] shows that the rebound angle is on the order of 1°. Additionally, the ice particles were found to expand in a circular fashion, holding the frame of reference with the motion of said particles, and correlations for the expansion velocity were possible. For low velocity impacts, ice particles and hail have been observed to bounce off of the impact surface. This occurs when the energy in the normal velocity component is not large enough to shatter the ice [5]. To quantify when an ice particle should be expected to break, [5] presents a model shown in Eq.(1.1), where $\mathbf{v}_c$ is the velocity threshold below which ice particles will bounce. This model is based on experimental data that relates the particle's kinetic energy to a threshold under which, the particle does not shatter.

$$||\mathbf{v}_c||^2 d_p = 0.516 \frac{m^3}{s^2} \tag{1.1}$$

These sets of experiments also provided sufficient data with which to determine the coefficient of restitution (COR) for non-shattering low-velocity impacts, which was found to be a piece wise function dependent upon the ratio $\frac{\mathrm{v}_n}{\mathbf{v}_c}$, where v_n is the velocity component normal to the impact surface [5].

Inertial Particle Separators

Inertial particle separators are devices that utilize the momentum of solid particles to separate them from a flow. This type of separator is the most applicable to gas turbine engines

because of their ability to remove particles from the flow without substantial pressure loss [1]. There are predominantly two styles of particle separators that have been studied: cyclone and particle deflectors with some form of collector. In practice, particle separators are built into the inlet of the engine [1], as would be most practical for ice particle separation. Figure 1.1 shows an example of a particle deflector type separator. This particular geometry is known as a louver particle separator because of its similarity to the louvers of a window shutter or blind. When it comes to the design of particle separators, there is still much to be learned. The prediction of the ice particle motion inside of the turbulent air flow is challenging and of critical importance to accurately assess particle separation efficiency. One strategy is through numerical simulation using computational fluid dynamics (CFD); however, such studies are computationally expensive. This work is part of an effort to create ML surrogates for particle motion that can be used to simplify separator design. The initial work presented here focuses on developing models that can accurately recreate particle trajectory.

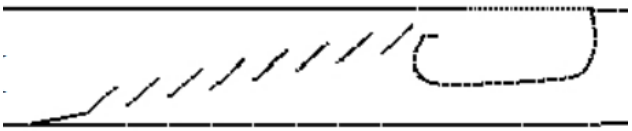


Figure 1.1: Louver particle separator geometry with collector bin [1]. In successful operation, the louvers deflect the particles upwards such that they flow into the collector bin and do not escape. Modified from [1].

1.2 Particle Motion Theory

It is known that the motion of particles suspended in a fluid can be characterized by the non-dimensional Stokes number, St, defined by Eq.(1.2), where τ_p is the particle relaxation or response time defined by Eq.(1.3) and τ_f is the appropriate timescale of the flow [14]. Physically, τ_p can be thought of as the time that it takes for the particle to respond to changes in the gas flow. Additionally, as expressed in Eq.(1.3), ρ_p and d_p are the particle density and diameter respectively, and μ_g is the dynamic viscosity of the gas [14].

$$St = \frac{\tau_p}{\tau_f} \tag{1.2}$$

$$\tau_p = \frac{\rho_p d_p^2}{18\mu_g} \tag{1.3}$$

The governing equation for a particle moving through gas, neglecting gravity, can be shown to be:

$$\frac{d\mathbf{v}}{dt} = -f\frac{\mathbf{v}-\mathbf{u}}{\tau_p} \tag{1.4}$$

where $\mathbf{v}$ is the particle velocity, $\mathbf{u}$ is the local gas phase velocity, and the function f introduces a nonlinear drag forcing dependence on the velocity difference $\mathbf{v}-\mathbf{u}$. For very small Stokes numbers ($St \ll 1$), the particle motion very closely follows the fluid trajectory, while large particles ($St \gg 1$) act in a more ballistic manner and are not especially driven by the flow streamlines [15]. The goal of this work is to predict particle trajectories using artificial neural networks (ANNs). The approach taken in this study is inspired by the asymptotic solution found by Maxey for the zero-gravity case [16]

$$\mathbf{v} = \mathbf{u} - \tau_p\left(\frac{D\mathbf{u}}{Dt}\right) + O(\tau_p^2) \tag{1.5}$$

where D/Dt is the material derivative. When Eq.(1.5) is substituted back into Eq.(1.4), the resulting expression gives

$$\frac{d\mathbf{v}}{dt} = \frac{D\mathbf{u}}{Dt} + O(\tau_p) \tag{1.6}$$

which confirms that the behavior of sufficiently small particles (small τ_p) closely follows that of the streamlines. However, Eq.(1.6) also hints at a strategy for larger particles. By considering τ_p explicitly, higher order terms in the asymptotic expansion can be derived. Rather than manually deriving these terms, they are implicitly derived using machine learning. To achieve this objective, simulations of particles in steady laminar flow fields are performed in the scientific Python environment (*SciPy*), and is discussed in more detail in Chapter 2. Particles of various diameters corresponding to Stokes numbers from 0.05 - 500 are considered to capture the influence of St. These simulations are used to extract relevant particle data to train and validate a neural network for the various flow configurations and combinations thereof.

1.3 Machine Learning

The field of artificial intelligence (AI) has captured audiences around the world in wonder of the science fiction-like works that are being developed. Researchers turn to AI in hopes to automate tasks once only possible by humans such as audio transcription, hand-written text recognition, and image classification. Machine learning (ML), a subset of AI, is the process whereby such intelligent systems extract useful patterns from raw data [17]. The structure and representation of this input data is highly important and can affect how well the ML model can interpret the data. One example of a simple machine learning algorithm is that of

logistic regression, which is a model that predicts the probability of an event. For example, logistic regression can be used by financial institutions to determine whether a transaction is fraudulent or legitimate. However, for more complex ML tasks, simple regressions are not sufficient. In ML applications dealing with computer vision, there are often multiple features in an image that the algorithm must recognize in order to make a decision. For example, if one is trying to design a computer vision model that determines whether or not there is a human face in the image, the model will need to identify if there is a head with two eyes and ears, a nose, and a mouth. Additionally, it should also determine if these features are arranged in the proper configuration. In terms of pixels, this model would be nearly impossible to hard code because slight variations, such as a shadow cast over the person's face, could throw the pixel values off enough to fool the model [17]. The solution to these more complex tasks involves what is known as deep learning. Deep learning allows the ML model to create more complex representations in terms of simpler representations. The classical deep learning example is the feedforward artificial neural network (ANN), which is sometimes called the multi-layer perceptron [17]. There has been much excitement in using ANNs for various applications across the scientific community, and not without cause. Hornik et al. established that feedforward ANNs are a class of universal approximators, given that a sufficient number of neurons, or hidden nodes, are provided [18]. Thus, many in the computational physics community are interested in harnessing the ability of ANNs to better model various highly non-linear phenomena.

1.3.1 Multiphase Flow Machine Learning

The idea of using ML to compute multiphase particle data is certainly not a new one. Much of the existing literature is focused on developing ML models that can better predict drag forces on particles. For instance, Minakowska et al. [19] developed deep ANNs capable of predicting hydrodynamic forces and torques on non-spherical particles. Specifically, they consider fully-connected (another term for feedforward) ANNs with three hidden layers (HL). Additionally, two different structures of the HL are utilized in their study. The first exhibits a large first HL followed by two smaller HLs of the same size. The second exhibits a rectangular structure, which means that each HL has the same number of neurons or nodes. Separate ANN models were considered because one of the desired model outputs naturally had slightly different dependencies that the others. Minakowska and colleagues reported average errors of less than 1% [19].

Raissi et al. [20] introduce a deep learning framework for data-sparse scenarios that combine discrete known data points with the governing equations in the loss function, termed physics-informed neural networks (PINNs). For clarity, the loss function of a given ANN governs how the model trains because the optimization algorithm seeks to minimize this function. Typical loss functions and optimization algorithms will be explored in more detail in Section 2.2. The PINN approach has become very popular for combining ML with problems that require an understanding of the governing physics. In some ways, this approach mitigates

the "black-box" approximation style that ANNs are often criticized for. Seyed-Ahmadi and Wachs [21] apply the PINN concept in a slightly different manner to model forces and torques in particle-laden flows with particle volume fractions (Φ) on the order of 0.1. However, they place much more emphasis on the structure of the ANN such that it is termed a physics-inspired neural network. The position vectors of neighboring particle locations, which are very important for predicting the force imparted by the fluid in high particle volume loading, are fed into the network. The ANN is designed such that the stream-wise and corresponding normal components are implicitly computed within the network before being projected onto the Cartesian coordinate system in the output layer. This was inspired by their previous success found in developing probability-driven models of hydrodynamic forces and torques in particle-laden flows [22]. In this study, the authors extract statistical information from particle-resolved direct numerical simulation (DNS) to create probability distribution maps as a basis for regression. This method was shown to predict up to 85% of the force and torque variations. It should be noted that the model inputs, as in the ANN approach, were the neighboring particle locations.

He and Tafti present a ML method for improving the prediction of drag forces on spherical particles in suspension [23]. Improving the prediction of these forces is useful for the computational fluid dynamics (CFD) discrete element method (DEM), because the particles are considered in a point-mass Langrangian scheme. This makes the motion of such particles directly dependent on the drag model. They train an ANN using data from particle resolved simulations (PRS). As with many studies that develop ML models to predict particle drag in heavy volume loadings, the model inputs include Re_p, Φ, and the coordinates of neighboring particles. Interestingly, they test two models: one where the ANN input uses the five nearest particle coordinates, while the other uses the fifteen nearest particle coordinates. They find that including the nearest fifteen particle coordinates results in superior performance. Additionally, the ANN is composed of 1 HL with 25 hidden nodes that uses the hyperbolic tangent activation function. The results of the relatively simple ANN model show a considerable improvement over using the mean drag [23].

A similar physics-guided neural network approach for modeling particle drag forces in high volume loading flows is introduced by Muralidhar et al. [24] and is dubbed "PhyNet." They warn that arbitrarily designed ANNs can fail to generalize trends in small sets of training data. This is often the case when training ANNs with particle resolved simulation data because of their high computational expense. Much like the approach described before, the ANN inputs are the coordinates of the 15 nearest particles along with Φ and Re_p. However, the novel approach presented is based on the fact that the resulting drag force is essentially the superposition of pressure and shear forces acting on the particle. It is known that the drag force corresponding to the pressure force on the particle is highly dependent upon the local pressure field, and likewise the shear force is dependent on the local velocity field [24]. Thus, the ANN model is structured such that the local pressure and velocity fields are to be determined by the ANN, followed by the pressure and shear forces. More specifically, the local pressure and velocity fields (represented by vectors) are fed into subsequent 1D

convolutional and maximum pooling layers that return what should be the shear and pressure components of the drag force. Strategically, they define the ANN loss function to include the intermediate variables such as the pressure and velocity fields. Muralidhar et al. show that PhyNet outperforms other state of the art models by 8.4% on average [24].

Balachandar et al. highlight the importance of creating ML models for particle-laden multiphase flow because of how standard force and torque models do not capture strong pairwise interaction from unique particle arrangements [25]. They found that the lack of particle-resolved DNS data caused their proposed ANN model to over-fit the training data. However, they also explored an alternative approach to directly predicting forces imparted on the particles. This two-step approach was designed to first predict the perturbation flow due to each neighboring particle, which was then used to construct an accurate prediction of the flow using superposition of the individual perturbations. Secondly, a similar perturbation force on each particle from a summation of each of its neighbors' perturbation flows was used for force prediction. This two-step approach proved much more viable because of the abundance of flow field data available to the model [25].

In another application, Wu et al. [26] apply a ML sub-grid scale (SGS) model to a large eddy simulation (LES) of turbulent particle laden flow at high Reynolds numbers (Re) with particles of $St = 0.01 - 20$. While DNS can resolve turbulence to its smallest scale and provide reliable particle-laden simulations, the computational cost can be extremely expensive especially at high Re. The LES method allows for lower computational expense at the cost of modeling the smaller turbulent structures instead of fully resolving them. They found that the ML SGS model, which had been previously validated with particle-free isotropic turbulent flows, improved the LES simulation agreement with DNS over LES with the dynamic Smagorinsky model (DSM) for all St considered [26]. This suggests that ML has multiple avenues of use for the particle-laden multiphase flow community. Although traditionally used for computer vision, current research is showing how ML can be used with strictly data-based applications. This work trains ANNs with well-defined, redundant data sets from simple flows to explore how these ANNs perform in scenarios with simple combined and complex flows using flow feature extraction.

Chapter 2

Methods

2.1 Governing Equations

The scope of particle motion studies conducted assume that the particle volume fraction, Φ, is small enough such that the dispersed particles do not affect the fluid physics; this is generally referred to as one-way coupled flow. Additionally, the particles considered are assumed to be perfectly spherical ice particles with an isotropic density of $\rho_p = 917 kg/m^3$ [27]. Particle rotation is not considered, and gravitational forces are also neglected because only 2D cases are explored. The governing particle dynamics equations are shown in Eq.(2.1) and Eq.(2.2) where $\mathbf{x}_p$ and m_p are the particle position and mass, respectively. The particle mass is computed as $m_p = \rho_p \pi d_p^3/6$. The drag force, $\boldsymbol{f}_D$, is given by Eq.(2.3) where the relative velocity is given by $\mathbf{v}_r = \mathbf{u} - \mathbf{v}$ and $A_p = \pi d_p^2/4$ is the projected particle area normal to the flow. The drag coefficient, c_D, is taken to be the Schiller-Naumann relation given in Eq.(2.4) [28]. The fluid density and dynamic viscosity of air are taken to be $\rho_g = 1.20\, kg/m^3$ and $\mu_g = 1.51\text{E}{-5}\, kg/(ms)$, respectively.

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{v} \tag{2.1}$$

$$m_p \frac{d\mathbf{v}}{dt} = \boldsymbol{f}_D \tag{2.2}$$

$$\boldsymbol{f}_D = \frac{1}{2} c_D \rho_g ||\mathbf{v}_r||^2 A_p \frac{\mathbf{v}_r}{||\mathbf{v}_r||} \tag{2.3}$$

$$c_D = \frac{24}{Re_p}(1 + 0.15 Re_p^{0.687}) \tag{2.4}$$

Additionally, the particle Reynolds number was taken to be

$$Re_p = \frac{\rho_g ||\mathbf{v}_r|| d_p}{\mu_g} \tag{2.5}$$

The velocity gradient tensor given by $\nabla\mathbf{u}$, can be written as the sum of a symmetric and an anti-symmetric tensor given by the strain rate tensor $\mathbf{S}$ and rotation tensor $\mathbf{\Omega}$, respectively. These tensors are shown in Eq.(2.6) and Eq.(2.7).

$$\mathbf{S} = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T) \tag{2.6}$$

$$\mathbf{\Omega} = \frac{1}{2}(\nabla\mathbf{u} - \nabla\mathbf{u}^T) \tag{2.7}$$

We use the second principal invariant, defined in Eq.(2.8), as an objective measure of the strength of the flows

$$I_2(\nabla\mathbf{u}) = \frac{1}{2}(\big(tr(\nabla\mathbf{u})\big)^2 - tr(\nabla\mathbf{u}^2)\big) \tag{2.8}$$

where $tr()$ represents the trace of the tensor. For an incompressible flow, Eq.(2.8) can be shown to simplify as $I_2(\nabla\mathbf{u}) = -\frac{1}{2}(\nabla\mathbf{u}^T : \nabla\mathbf{u})$. Note that the convention for the inner product is taken from [29] as shown in Eq.(2.9)

$$A : B = A_{ij}B_{ji} \tag{2.9}$$

where Einstein's summation convention is used. For use as a flow time-scale, we define J as shown in Eq.(2.10). Similarly, S and Ω are defined as metrics to describe the relative contributions of fluid strain and rotation at a given particle location as shown in Eq.(2.11) and Eq.(2.12), respectively.

$$J = \sqrt{|I_2(\nabla\mathbf{u})|} \tag{2.10}$$

$$S = \sqrt{|I_2(\mathbf{S})|} = \sqrt{\frac{1}{2}(\mathbf{S}^T : \mathbf{S})} \tag{2.11}$$

$$\Omega = \sqrt{|I_2(\mathbf{\Omega})|} = \sqrt{\frac{1}{2}(\mathbf{\Omega}^T : \mathbf{\Omega})} \tag{2.12}$$

It can be shown that for a general incompressible flow, $J = \sqrt{S^2 + \Omega^2}$. Therefore, this work asserts that the decomposition of a complex flow into its rotational and sheared components can be thought of as the classical unit circle as suggested in Figure 2.1. The positive convention for θ is counterclockwise measured from the e axis and can be defined as $\theta = \arctan 2(\omega/e)$.

2.2 Feedforward Artificial Neural Networks

Artificial neural networks (ANNs) got their name from the idea that the networks' function resembles that of biological neurons in the human brain. In fact, early ML algorithms were inspired by computational models of the brain's learning process [17]. The mapping of input data to the often abstract features that ultimately allow the ML model to serve as

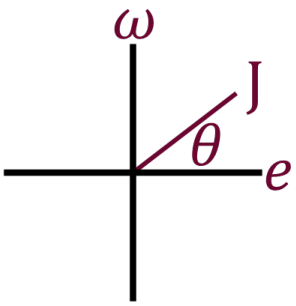


Figure 2.1: Rotation and shear component decomposition.

a universal approximator [18] can be quite complex; therefore, deep learning models can require many hidden layers, as shown in Figure 2.2. ANN model hyperparameters are the parameters that are used to specify the model structure and how it will train. The highest level hyperparameters that determine the model structure are the number of hidden layers (HL) and the number of hidden neurons in each HL, denoted by the circles in Figure 2.2. Some guidelines exist for choosing these hyperparameters, such as the data storage capacity for a 2 HL feedforward ANN shown by [30]. Additionally, certain models have been shown to prefer a large first HL followed by successively smaller HLs [31]. However, the optimal ANN architecture is often specific to the ML application at hand. Additionally, the percentages of data to train and validate the model, the learning rate (lr), optimization algorithm, activation function to be applied for each HL, and the loss function must be specified by the model designer. These hyperparameters and how they relate to the ANN model will be detailed in this section.

Feedforward ANNs are one of the simplest configurations of artificial neural networks. Just as the name suggests, the input data propagates forward through each layer. More specifically, each neuron has associated weights w and a bias b such that the output z of a given neuron is given by

$$z = \sum_{i}^{j} w_i x_i + b \tag{2.13}$$

where j is the size of the previous layer in the ANN. However, the output of the neuron is usually $\sigma(z)$, where σ is the activation function chosen for the model [32]. The primary choice for the activation function in the early days of ANNs, before computing technology enabled training deep neural networks, was the sigmoid activation function $S(z)$ defined by Eq.(2.14). The output of this activation function is designed such that $S(z \to \infty) = 1$ and $S(z \to -\infty) = 0$. This activation function, while no longer used in practice, can be loosely related back to the biological inspiration of ANNs. If the output of the activation function

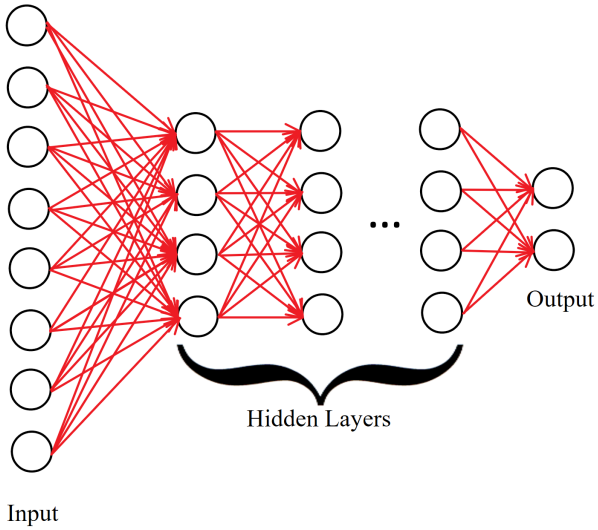


Figure 2.2: Deep feedforward neural network architecture. The input data propagates through the ANN layer by layer until the output is reached.

is close to 1, this can be considered analogous to a neuron in the brain "firing."

$$S(z) = \frac{1}{1 + e^{-z}} \tag{2.14}$$

The premise of this activation function is that certain neurons will be activated by particular features of the input data, which made this a good activation function for simple classification tasks. However, most ANNs being used for computer vision tasks today use the rectified linear unit (ReLU) function as defined in Eq.(2.15)

$$ReLU(z) = \left\{ \begin{array}{ll} z, & z \geq 0 \\ 0, & z < 0 \end{array} \right\} \tag{2.15}$$

namely because ReLU is a non-linear activation function that works well in practice. However, the optimal activation function is often case specific. While any function can be specified as the activation function, some additional standard choices such as the hyperbolic tangent function (tanh) are shown in Figure 2.3. It is important to note that three variations of ReLU are shown in Figure 2.3 including Leaky ReLU, the exponential linear unit (ELU) function, and the scaled exponential linear unit (SELU) function. These variations exist to remedy what is known as the dying ReLU problem. Because the ReLU activation function returns 0 for any input less than 0, sometimes enough input information is truncated such that the ANN is unable to sufficiently train because too many neurons have "died." Deep ANNs tend to suffer more often from the dying ReLU problem [33]. The main advantages that ReLU boasts over the sigmoid activation function are that it is simpler and it doesn't suffer from what is called the vanishing gradient problem. Before explaining

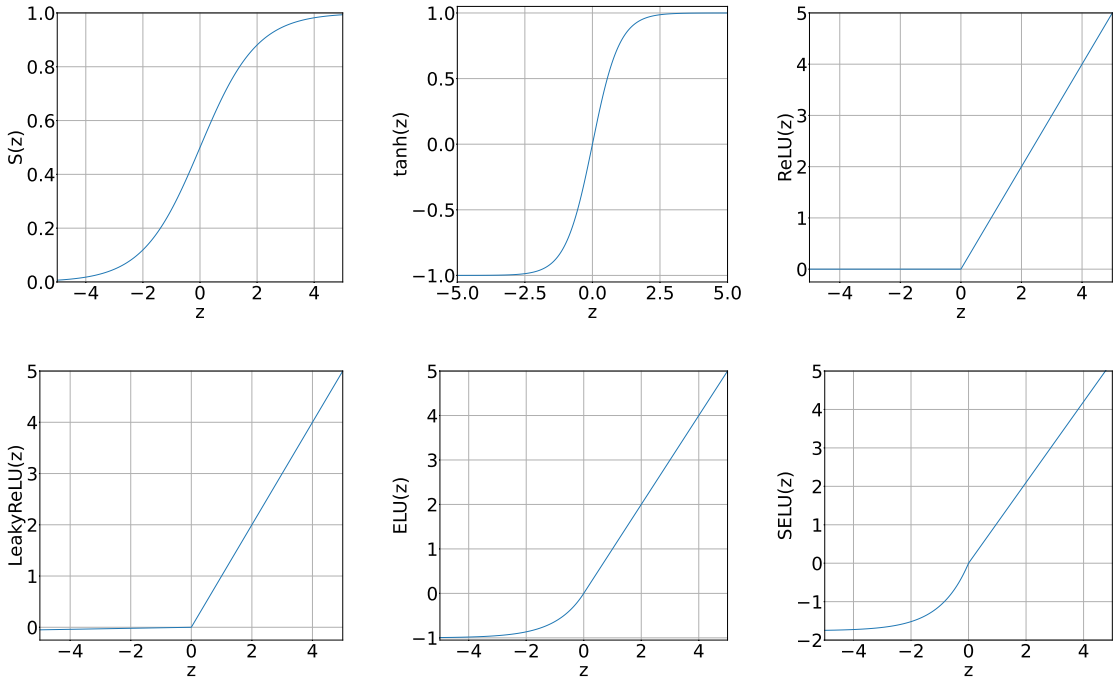


Figure 2.3: Various standard activation functions used in ML.

what the vanishing gradient problem entails, it is first necessary to explain how the ANN is trained.

In order to quantify how well the ANN is modeling the data and actually train the ANN, a loss function (sometimes referred to as the cost function) must be defined that quantifies the error between the model outputs and the true values. The loss function should be defined such that its value is always greater than or equal to zero. Typically, the mean squared error (MSE) loss function as defined in Eq.(2.16) is used to ensure that the function behavior is smooth, where N is the sample size, z_{NN} is the model output, and z is the true value. However, there are other loss functions that may be preferred depending on the application at hand. For example, when applying ANNs to classification problems, the cross entropy loss function is a popular choice because it is designed to represent the difference between two probability distributions (i.e. the probabilities that a given image contains a cat, dog, or horse). Additionally, it is important to note that the key tenant of the physics-informed neural networks (PINNs) proposed by [20] is that the PINN's loss function directly depends upon governing equations for the given problem. The importance of the loss function selection cannot be overstated, as the ANN will perform according to its definition.

$$\text{MSE} = \frac{\sum_{i=1}^{N} ||z_{NN,i} - z_i||^2}{N} \tag{2.16}$$

The goal of training any ANN is to adjust the weights and biases to minimize the loss function. A standard algorithm for this minimization is known as gradient descent [32]. A simple case of gradient descent can be thought of as attempting to find the global minimum of a multi-variable parabolic function. Such a function is demonstrated in Figure 2.4. We know that the minimum value for a parabola occurs at the vertex. However, this minimum can be found numerically as well by computing the derivative at one's initial position and moving opposite the direction of the gradient and repeating until the minimum is reached as suggested by the vectors in Figure 2.4. This is obviously a very simple example, as the loss function that is being minimized for an ANN can be a function of hundreds of thousands and even millions of weights and biases. It becomes increasingly difficult to visualize the mechanism of gradient descent for functions of more than two variables. However, the process by which such loss functions are minimized is similar.

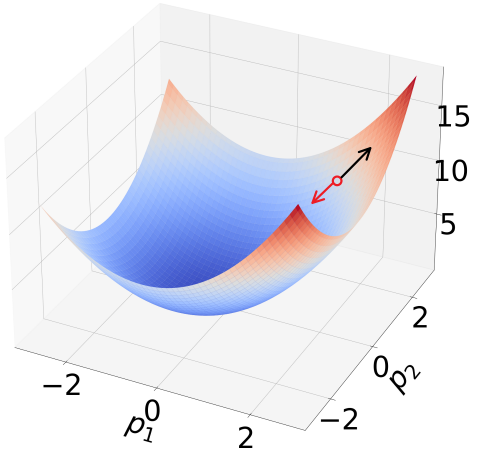


Figure 2.4: The multi-variable parabolic function $f(p_1, p_2) = (p_1 + 0.5)^2 + p_2^2$ is shown with an arbitrary starting point. The gradient $\nabla f(p_1, p_2)$ is shown at the point with the black vector. The opposite direction of the gradient vector, which is the direction that gradient descent utilizes, is shown by the red vector.

Back-propagation is the computation of the loss function gradient for each weight and bias, and from this computation the weights and biases can be updated to move towards the minimum. To speed up the training process, the gradient is often estimated by using a sample of the training data. This is known as stochastic gradient descent (SGD) [32]. The amount that the weights and biases are changed proportionally to the loss function gradient $\nabla loss$ is dependent upon the learning rate (lr) hyperparameter as shown in Eq.(2.17),

$$p_i^{k+1} = p_i^k - lr\, \nabla loss \tag{2.17}$$

where p_i^{k+1} is the i^{th} updated model parameter and p_i^k is the i^{th} parameter from the k^{th} iteration. The lr must be chosen to be small enough such that the gradient descent algorithm does not overshoot the minimum, but large enough to ensure convergence is not too slow.

Often, the lr is also problem dependent and should be tuned accordingly. There are many other optimization algorithms standard in machine learning that are variations of SGD, such as Adam [34], Adagrad [35], Adadelta [36], RMSProp [37], and SGD with Nesterov momentum [38]. The Adam optimization algorithm is one of the more standard approaches of those listed because of how efficient it is. It was intended to combine the best features of Adagrad and RMSProp, and computes adaptive learning rates for individual parameters [34]. Therefore, the hyperparameters associated with the Adam optimizer should require minimal tuning, which makes it an attractive algorithm for ML research. These algorithms are explored in some capacity and compared to the standard SGD optimizer as a baseline. Some optimization algorithms, such as SGD with Nesterov momentum, require additional hyperparameters such as momentum (μ), which can be tuned for faster training convergence.

It is worth mentioning at this point that there are generally two overarching categories of ML models: supervised and unsupervised. Unsupervised ML is often used to find hidden patterns in data, or when it is unclear what patterns should even be searched for. For example, medical data might be used to better understand what factors and combinations of factors are more significant to a given patient's risk of cancer. In supervised ML, which is exclusively used in this work, the majority of the data set is typically labeled as training data and the rest is designated as validation data. The model is trained with an optimization algorithm using only the training data, and the validation data is used to ensure that the model is generalized and not over-fitted to the training data. Typically, at least 70% of the total data is designated as training data because the success of many ML models requires large amounts of data. Circling back to the activation function concept, it is worth noting that ReLU is not good for our application because it removes a significant portion of information from the input data as the nature of vectors such as velocity and acceleration have negative values. Kim and Lee [39] stated that the exponential linear unit (ELU) function, which is similar to ReLU, slightly outperforms ReLU. Other multiphase flow physics related ML models such as PhyNet [24], as described in Section 1.3.1, also use ELU.

2.2.1 Vanishing Gradients and Xavier Initialization

As previously mentioned, the ReLU activation function has one clear advantage over the logistic, or sigmoid function. While it can succumb to the dying ReLU problem previously described, it does not suffer from the vanishing gradient problem. The vanishing gradient problem can result of back-propagation in ANNs with many layers. The problem is especially prevalent for the sigmoid activation function described earlier, because as shown in Figure 2.5, the maximum value of the derivative of the sigmoid function is 0.25. As more layers are added, the back-propagation derivatives become progressively smaller, making it nearly impossible for the ANN to learn.

While the sigmoid activation function is the classical example to illustrate vanishing gradients, this problem does occur with others such as tanh. In the early 2000s, it was thought

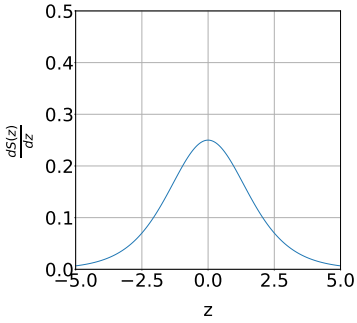


Figure 2.5: Derivative of the sigmoid function $\frac{dS(z)}{dz}$

that deep ANNs could not be successfully trained; however, in 2010 Glorot and Bengio developed a weight initialization scheme that allowed for faster training convergence of deep neural networks. Specifically, this scheme initializes the weight matrix in each HL with a normalized uniform distribution in range $[-\frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}]$, where n_j is the size of the HL and n_{j+1} is the size of the subsequent HL [40]. This uniform distribution initialization will hereafter be referred to as Xavier initialization. Their results show that Xavier initialization helps keep a similar distribution of activation values in each HL and avoids the vanishing gradient problem for the weight gradients [40].

2.2.2 ANN Hyperparameter Testing

The overall design space of deep feedforward neural networks is far too vast to be completely searched. Therefore, multiple design of experiments strategies are employed. First, artificial neural networks (ANNs) arranged in 2-6 hidden layer structures are compared and a local optimum for the number of hidden layers (HL) is determined. The total number of hidden neurons is conservatively determined by the learning capability relations proven by Huang [30] for a two HL feedforward ANN as shown in Eq.(2.18) and Eq.(2.19),

$$L_1 = \sqrt{(m+2)N} + 2\sqrt{\frac{N}{m+2}} \tag{2.18}$$

$$L_2 = m\sqrt{\frac{N}{m+2}} \tag{2.19}$$

where N is the number of distinct training samples and m is the number of output neurons. The configurations detailed in Table 2.1 are used to evaluate the effect of the number of HLs. Note that the total number of neurons in each network and the decreasing ratio (approximately 2.5:1) of neurons in the HLs are both kept constant. The ratio of $\frac{L_n}{L_{n+1}} \approx 2.5$ was determined by directly calculating the ratio from Huang's expressions and considering the results of a brute-force optimization of a two layer neural network presented by Heaton

[31]. Using Huang's equations, the ratio of Eq.(2.18) to Eq.(2.19) can be found to satisfy $\frac{L_1}{L_2} = 1 + \frac{4}{m}$ and yields $\frac{L_1}{L_2} = 3$ for this application. The optimal configuration that Heaton found for a two HL ANN had a ratio closer to 2:1, but in general it was found that favorable architectures had a large first HL followed by smaller HLs [31]. Proceeding with the observed optimal structure after training each ANN for one hour, several activation functions such as the rectified linear unit (ReLU), hyperbolic tangent (tanh), and Softmax are compared to determine an optimal function for this application. Additionally, each activation function considered is tested with and without Xavier initialization [40]. Finally, using the combination of the best structure and activation function, various data pre-processing techniques are employed. These pre-processing methods are discussed in more detail later.

Table 2.1: Hidden layer configurations for single flow ANNs

Number H.L.	L_1	L_2	L_3	L_4	L_5	L_6
2	425	142	-			
3	364	145	58	-		
4	349	140	56	22	-	
5	343	138	55	22	9	-
6	341	136	55	22	9	4

PyTorch

We use the open source ML library *PyTorch* to design ANNs that are trained using multiphase flow data generated from the canonical flows and combinations thereof. *PyTorch* is a widely deployed open-source ML framework with a strong research support community. Additionally, *PyTorch* supports single- and multi-GPU training, which is particularly useful for larger models with extensive amounts of data.

2.3 Training Data Preprocessing

Although not a main objective, this work does consider high level ANN hyperparameter tuning and the results presented in this work should be used as a guide for similar machine learning applications. The ANN structure is analyzed, as described above, along with the activation function and data preprocessing. For each flow type considered, two main configurations of the *PyTorch* ANN will be tested. The first will be a data-driven (DD) model, in which each of the relevant parameters are given directly to the ANN to compute the particle

acceleration, $\frac{d\mathbf{v}}{dt}$, or a mapping of the form:

$$\frac{d\mathbf{v}}{dt} = g\left(\mathbf{v}, \mathbf{u}, d_p, \frac{D\mathbf{u}}{Dt}, \rho_p, \rho_g, \mu_g\right) \tag{2.20}$$

This configuration's performance will be compared with a physics-driven method, where many of the above variables are condensed into parameters with more physical significance, as the following mapping shows:

$$\frac{d\mathbf{v}}{dt} = g\left(\mathbf{v} - \mathbf{u}, \tau_p, \frac{D\mathbf{u}}{Dt}\right) \tag{2.21}$$

It is expected that the ANN trained with physics-driven preprocessed (PDP) data will perform just as well or better than the DD method because it contains essentially the same information, albeit with less data which is advantageous. In addition to testing the physics-informed approach against the data-driven method in the default Cartesian coordinates of the simulation, ANNs are also trained using the same data with a transformation of coordinates to streamline and path coordinates as shown in Figure 2.6. The solid line in Figure 2.6 represents a fluid streamline and the dashed line represents a particle path. The streamline coordinates are represented by $\hat{s}$-$\hat{n}_s$ and the path coordinates are given by $\hat{t}$-$\hat{n}_t$. $\hat{s}$ and $\hat{t}$ are tangent to the fluid streamline and particle path (i.e. the directions of $\mathbf{u}$ and $\mathbf{v}$), respectively and $\hat{n}_s$, $\hat{n}_t$ are the corresponding normal vectors. That is, $\hat{s} = \frac{\mathbf{u}}{||\mathbf{u}||}$ and $\hat{t} = \frac{\mathbf{v}}{||\mathbf{v}||}$. The corresponding normal vectors are computed in the direction of the streamline and path curvatures as suggested in Figure 2.6. This direction is determined via the direction of the fluid and particle accelerations, respectively, as shown in Eq.(2.22) and Eq.(2.23).

$$\hat{n}_s = \frac{\frac{D\mathbf{u}}{Dt} - (\frac{D\mathbf{u}}{Dt} \cdot \hat{s})\hat{s}}{||\frac{D\mathbf{u}}{Dt} - (\frac{D\mathbf{u}}{Dt} \cdot \hat{s})\hat{s}||} \tag{2.22}$$

$$\hat{n}_t = \frac{\frac{d\mathbf{v}}{dt} - (\frac{d\mathbf{v}}{dt} \cdot \hat{t})\hat{t}}{||\frac{d\mathbf{v}}{dt} - (\frac{d\mathbf{v}}{dt} \cdot \hat{t})\hat{t}||} \tag{2.23}$$

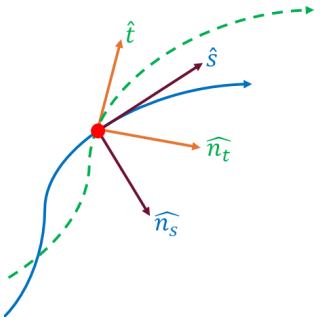


Figure 2.6: Streamline and path coordinate systems.

2.3.1 Data Scaling

Early tests using raw input data shown in Eq.(2.20) to train relatively shallow (2-3 HL) ANNs suggested that there is an inherent need to scale the data such that the machine learning model can effectively learn from the training data set. Two such scaling strategies were explored. The first was a column-wise normalization of the input and output data tensors shown by Eq.(2.24), where each variable denoted by y_i is normalized by its corresponding absolute maximum in the data-set (i.e., each d_p is normalized by the largest d_p and so forth). This normalization ensures that the data is scaled within range $[-1, 1]$. The second strategy was a Gaussian standardization approach shown in Eq.(2.25), where $\bar{y}$ and s are the sample mean and sample standard deviation of the data set, respectively. This form of scaling does not impose any predefined limits on the scaled data set; however, for a normally distributed data set the standardized inputs $\tilde{y}_i$ should primarily be in range $[-3, 3]$.

$$\hat{y}_i = \frac{y_i}{max(|\mathbf{y}|)} \tag{2.24}$$

$$\tilde{y}_i = \frac{y_i - \bar{y}}{s} \tag{2.25}$$

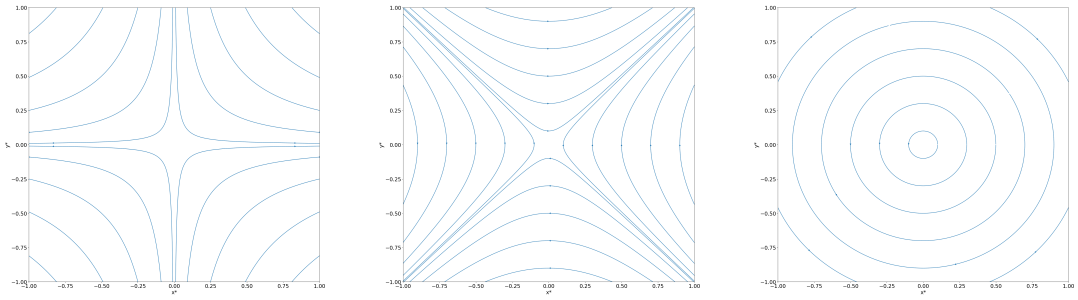
Initial testing suggested that the absolute maximum normalization was superior to the standardization approach. This is likely due in part to the wide range of particle sizes considered. It's unlikely that the data would assume a Gaussian distribution for this reason, as small particles will experience a much larger acceleration than the larger particles. The ANN training results presented in this work utilize the normalization approach for preprocessing the training data, unless otherwise specified. It is also important to note that the ANN-predicted particle accelerations are un-normalized for any post-processing error analysis presented, as the error between scaled data is *not* the same as the un-normalized, or full-scale, data.

Chapter 3

Canonical Flows

In order to assess the viability of using feedforward artificial neural networks (ANNs) to predict particle acceleration in particle-laden flow, simple tests are in order. The simplest flow cases are steady, laminar canonical flows that are either purely rotational or irrotational. The ANN hidden-layer structure and activation function are iteratively explored to determine best practices for subsequent trials. Since ANNs are only able to learn patterns in the data that they are presented, it is also important to compare different data preprocessing methods and data representations.

Following [15], the motion of particles in the following steady canonical flows are used to assess the feasibility of using ANNs to predict particle acceleration: stagnation point, pure (irrotational) shear, and forced (rotational) vortex. Stagnation point flow is an irrotational flow characterized by the strain rate, A, in the following 2D velocity field: $\mathbf{u} = [Ax_1, -Ax_2]$. Pure shear, similarly, is an irrotational flow given by $\mathbf{u} = [ex_2, ex_1]$ where e is the shear rate. Lastly, the forced vortex is a purely rotational flow characterized by the rotation rate, ω, in $\mathbf{u} = [-\omega x_2, \omega x_1]$. Figures 3.1a, 3.1b, and 3.1c show the velocity streamlines of the stagnation point, pure shear, and forced vortex for $A = e = \omega = 1.0s^{-1}$. For the stagnation point, pure shear, and forced vortex flow, the second principal invariant of the velocity gradient tensor, $I_2(\nabla\mathbf{u})$, can be shown to simplify as A^2, e^2, and ω^2, respectively. The metrics S and Ω that describe the contribution of fluid strain and rotation are also considered, and have been previously defined in Eq.(2.11) and Eq.(2.12), respectively. Not surprisingly, $\Omega = 0$, $S = A$, $J = A$ and $\Omega = 0$, $S = e$, $J = e$ for the stagnation point and irrotational shear flows, respectively. Similarly, $\Omega = \omega, S = 0, J = \omega$ for the forced vortex case. For this reason, the directly comparable flow parameters A, e, and ω are taken to be $\frac{1}{\tau_f}$ for each canonical flow case, where τ_f is the appropriate flow timescale. To generate the ANN training data, the governing equations (Eq.(2.1) and Eq.(2.2)) of the particle dynamics are solved using numerical integration techniques in the scientific Python computing environment (*SciPy*). The particle initial conditions are taken as $\mathbf{v}(t_0) = [0, 0]$ and $\mathbf{x}_p(t_0) = [x_{1,0}, x_{2,0}]$, where the initial positions $x_{1,0}$ and $x_{2,0}$ are randomly selected. However, $x_{1,0}$ and $x_{2,0}$ are both restricted



(a) Stagnation Point (b) Pure Shear (c) Forced Vortex

Figure 3.1: Streamlines of the canonical flows used for generating ML data.

to initialize in range $[-1, 1]$. Specifically, because canonical flow systems can present stiff systems of differential equations for the particle trajectory, the LSODA method was employed [41]. Some brief testing was performed with stiff ODE solvers in *SciPy* including LSODA, Radau, and backwards differentiation. It was found that LSODA was more computationally efficient for this problem, and hence it was selected for the study. Five different particle sizes corresponding to $St = 5 * [10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}]$ are considered in this study. It should be noted that the ice particle diameters corresponding to these St range from 122 μm to 1.22 cm. This range includes small atmospheric ice crystals that exist on the order of 100s of μm [8] through hailstones that exist on the order of 1 cm [42]. The relevant parameters are extracted for 5,000 particles of each St at $t^* = 1.0$ ($t^* = tJ$), and 80% of the data are used for training the ANNs and the remaining 20% are used for validation.

3.1 Neural Network Design

3.1.1 Hidden Layer Structure

The hidden layer (HL) structure testing detailed in Table 2.1 was performed using training data from the forced vortex case ($\omega = 1.0s^{-1}$) of the Cartesian data-driven (DD) approach. Each ANN was trained using the hyperbolic tangent activation function (tanh) on an NVIDIA V100 GPU for one hour. As shown in Figure 3.2, increasing the number of HL does appear to have some marginal benefit for this particular structure. This benefit is especially clear between the 2 HL and 3 HL cases as shown in Table 3.1. As the number of HL is increased to 3, the number of epochs (training iterations over the entire data set) completed in the allotted training time increased by about 20%. Subsequently, the final training loss, computed as mean squared error (MSE), is smaller. Additionally, it can be seen that ANNs with fewer HLs train more quickly during the first part of training but train more slowly

near the end of the allotted time frame. This can likely be attributed to the lower number of back-propagation calculations necessary for the shallow ANNs and the increased capability of deep neural nets to more easily capture complex phenomena in training data. From the ANN structures tested here, the 4 HL appears to facilitate the fastest training and lowest overall loss.

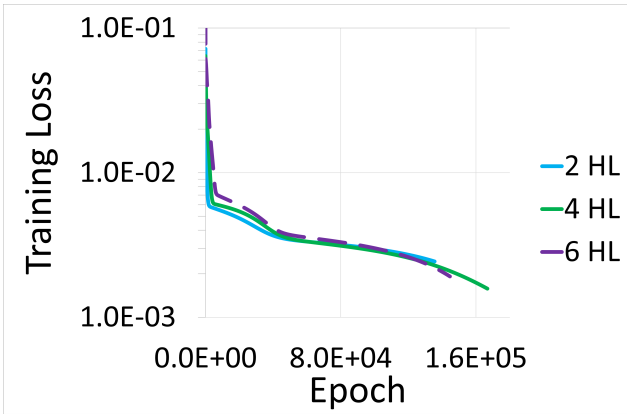


Figure 3.2: ANN HL structure training performance comparison. Trained with Cartesian DD approach data set using tanh activation function.

Table 3.1: HL structure performance. Trained with Cartesian DD approach data set using tanh activation function.

Number H.L.	No. Iterations	Training Loss
2	1.36E+05	2.43E-03
3	1.64E+05	1.91E-03
4	1.67E+05	1.58E-03
5	1.57E+05	1.65E-03
6	1.45E+05	1.91E-03

3.1.2 Activation Function and Xavier Initialization

Once the 4 HL structure was deemed the local optimum of the design space considered, several activation functions were compared as shown in Figure 3.3. The forced vortex Cartesian DD training set was again used to compare activation functions. As before, each neural network shown was trained on an NVIDIA V100 GPU for one hour. Additionally, the ANN training results shown with dashed lines for each corresponding activation function were

generated using HL weights generated using the uniform Xavier (also known as Glorot) initialization previously discussed in Section 2.2.1 [40]. Clearly, the Softmax and LogSigmoid function fail to train the ANN in any meaningful way. Since the 4 HL net is within the realm of deep neural networks, it was expected that the ReLU class of activation functions would be the best for training; however, this is not exactly the case. The Rectified Linear Unit (ReLU) and LeakyReLU activation functions perform slightly better than the Softmax and LogSigmoid functions, only to suffer from what appears to be the dying ReLU problem. This is the case for both the standard and Xavier initializations. One possible explanation for the lack of training with the ReLU functions is that their function structure is not well-suited to handling data where positive and negative values are present as is the case for velocity and acceleration data. This is interesting because the ReLU activation function is the most widely used in deep learning applications for both research and practice [43]. The tanh activation function appears to have performed reasonably well. Additionally, it is clear that the Xavier initialization provides some advantage for use with tanh because the loss decreases faster than the standard case. Lastly, the Scaled Exponential Linear Unit (SELU) activation function, which causes the ANN to exhibit some self-normalizing properties [44], outperforms the other activation functions considered and achieves a loss of less than 1.0E−4 in the 1 hour of training time. The Xavier initialization does not appear to provide any immediately obvious advantages in use with SELU; however, it does not negatively impact the training and may provide some benefits in a case trained for a longer duration. For this reason, SELU and the uniform Xavier initialization are used in subsequent training.

3.1.3 Data-Driven vs Physics-Driven Preprocessing

Figure 3.4 shows the training results of the six different combinations of data preprocessing methods for the stagnation point, irrotational shear, and forced vortex flows. Table 3.2 shows the final training loss for each flow case where $A = e = \omega = 1.0s^{-1}$. Again, each neural network shown was trained on an NVIDIA V100 GPU for one hour. It is worth noting that the trends shown here may not be comprehensive because the loss is only sampled every 50 epoch. As shown in Table 3.2, the training data in the streamline and path coordinate systems generally achieve lower loss than their Cartesian counterparts. Stagnation point flow is the only real exception in this case as the ANN trained with Cartesian DD training data closely compares to the streamline DD trained ANN. This can likely be attributed to the fact that the Cartesian and streamline coordinate systems are similar near the x and y axes for stagnation point flow. It is also clear from Figure 3.4 that while the physics-driven preprocessing (PDP) approach tends to perform slightly better during the initial training, the DD approach achieves a considerably lower loss by the end of the training time. Hence, path and streamline DD emerge as the top two approaches considered. Their ANN training results are quite similar for the forced vortex and irrotational shear cases. However, this study proceeds with the streamline DD approach because it appears to have less indeterministic behavior overall compared to the path DD approach, especially in the forced vortex case

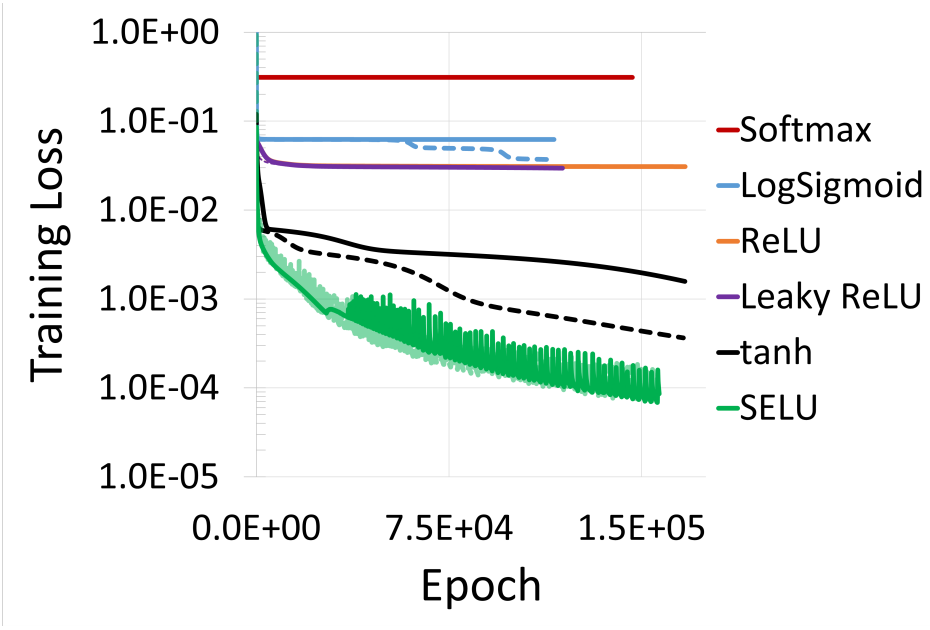
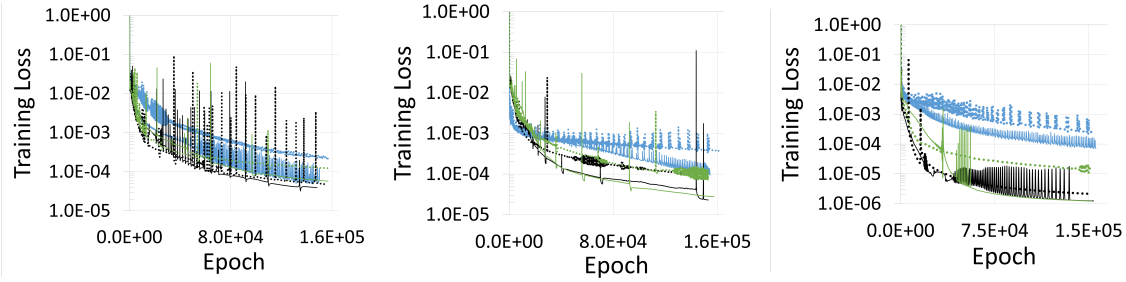


Figure 3.3: Activation function comparison. Standard SGD was used as the optimizer for each case shown here, and the dashed lines indicate use of the uniform Xavier weight initialization. Note that the SELU activation function with Xavier initialization is shown by the translucent green line.

shown in Figure 3.4c.



(a) Stagnation Point (b) Irrotational Shear (c) Forced Vortex

Figure 3.4: Data preprocessing effects. The Cartesian, path, and streamline coordinates are denoted by the blue, black, and green lines, respectively. The DD and PDP data are indicated with solid and dotted lines, respectively.

Table 3.2: Data preprocessing effects. Trained with SGD using SELU activation function. Final training loss after one hour is shown for each flow case and data preprocessing method.

Forced Vortex	DD	PDP
Cartesian	6.98E-05	2.39E-04
Path	1.23E-06	2.12E-06
Streamline	1.21E-06	1.94E-05

Irrotational Shear	DD	PDP
Cartesian	9.41E-05	3.71E-04
Path	2.30E-05	9.35E-05
Streamline	2.75E-05	7.14E-05

Stagnation Point	DD	PDP
Cartesian	6.36E-05	2.44E-04
Path	3.91E-05	4.98E-05
Streamline	5.80E-05	1.22E-04

3.1.4 Optimization Algorithm

Having now determined an optimum activation function and data preprocessing method, the forced vortex ($\omega = 1.0s^{-1}$) streamline DD training data set was used to explore the performance of typical optimization algorithms. For useful comparison, all optimizers are assigned $lr = 10^{-3}$, and the stopping loss criteria was 10^{-6}. As shown in Figure 3.5, the Adam algorithm trains the network faster than all other algorithms, stopping in just 597 epoch. Adagrad and SGD with Nesterov momentum (μ=0.9) exhibit similar performance and each meet the stopping criteria in around $9.0E4$ epoch. Note that standard SGD and SGD with standard momentum were also tested and SGD with Nesterov momentum outperformed these two. Lastly, the Adadelta and RMSProp algorithms do not train the ANN to the stopping criteria. RMSProp exhibits heavy oscillation during training that was found to be independent of lr. While Adam does exhibit some oscillation, better shown in Figure 3.6, the overall training trend is obvious. In the case that one has access to computing resources without time restrictions, Adam is the obvious choice. However, if time usage of computing resources is limited, SGD with Nesterov momentum is likely the better choice as its convergence seems to be monotonic. Additionally, Adam is a good development tool to use to determine how much convergence is possible for a given ANN.

Figure 3.6 shows the effect of lr with the Adam algorithm and the validation loss for the $lr = 10^{-3}$ case. During this testing, it has been found that Adam is sensitive to lr. It is therefore appropriate to iteratively tune the lr for the particular ANN that one is developing, which was surprising since Adam is described as needing little tuning. Also note that the validation loss closely follows that of the training data. This was expected, as the overall data set is slightly redundant and the validation and training data sets should be very similar.

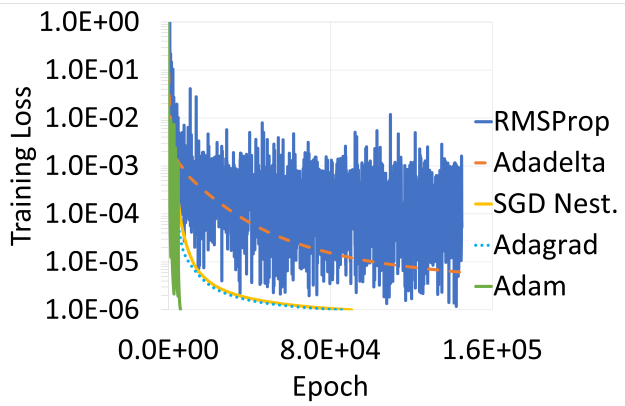


Figure 3.5: Optimization algorithm comparison ($lr = 10^{-3}$).

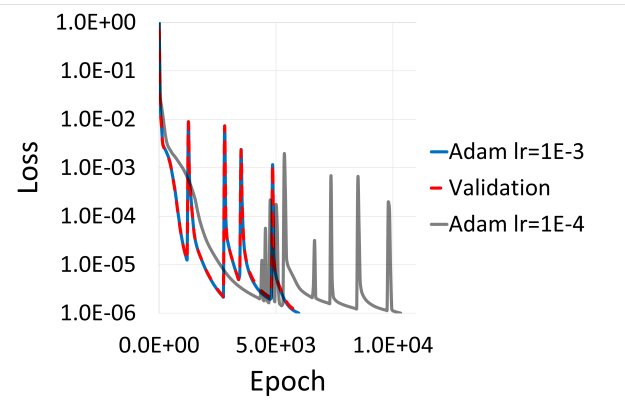


Figure 3.6: Adam validation and learning rate comparison. The blue and grey lines indicate the training MSE loss for their respective learning rates. The dotted red line indicates the validation loss corresponding with the blue line.

3.2 ANN Validation

After training ANNs for each of the canonical flows considered to MSE $< 10^{-6}$ with the Adam algorithm, the particle acceleration from the governing equations was compared to the ANN-predicted acceleration. Figure 3.7 shows this visual comparison for the forced vortex case by Stokes number (St), where the simulation acceleration and ANN-predicted acceleration are shown with the blue and red arrows, respectively. Upon visual inspection, the ANN-predicted accelerations appear to match the actual accelerations well. However, it is difficult to see some of the smaller vectors near the origin.

Now that a qualitative analysis of the ANN performance has been demonstrated, a quantitative analysis is in order to better characterize the model error. The directional error β and

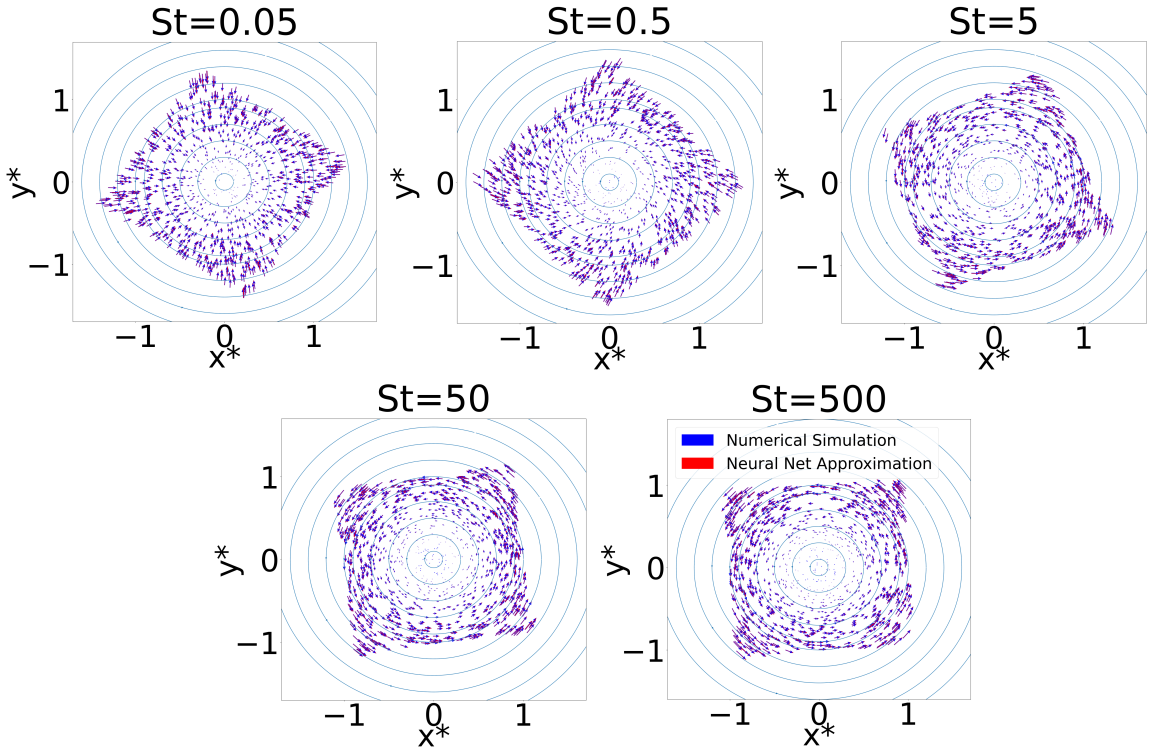


Figure 3.7: Particle acceleration vector field - Forced vortex simulation results vs ANN prediction. The blue vectors indicate the true acceleration $\mathbf{a}$ and the red vectors represent the ANN prediction $\mathbf{a}_{NN}$

the error vector $\mathbf{e} = \mathbf{a} - \mathbf{a}_{NN}$ are defined as shown in Figure 3.8 and Eq.(3.1), where $\mathbf{a} = \frac{d\mathbf{v}}{dt}$ as defined in Eq.(2.2) and $\mathbf{a}_{NN}$ is the particle acceleration predicted by the ANN.

$$\beta = \arccos\left(\frac{\mathbf{a} \cdot \mathbf{a}_{NN}}{||\mathbf{a}||\,||\mathbf{a}_{NN}||}\right) \tag{3.1}$$

The relative error is defined as $\boldsymbol{\epsilon} = \frac{\mathbf{e}}{||\mathbf{a}||}$ and is used to ensure that the metric is comparable for various St. Additionally, a third relative scalar error metric is defined as $E = (||\mathbf{a}_{NN}|| - ||\mathbf{a}||)/||\mathbf{a}||$, which describes how well the magnitude of the acceleration was predicted relative to $\mathbf{a}$ and if it was over- or under-predicted.

Figure 3.9a shows the mean value of β calculated from the validation data sets (5000 total particles) as a function of St for each canonical flow considered. As shown, the mean directional error remains less than 0.5° for St $O(1)$ or lower, and increases for larger St. It is also useful to know if the error is entirely random or if the mean error is influenced by outliers. Figure 3.10 shows surface plots of directional error β as a function of $\mathbf{x}_p^*$ for the rotational vortex, where the red color indicates higher error. Clearly, the ANN struggles to accurately predict the acceleration direction near the origin of each canonical flow. This is likely the

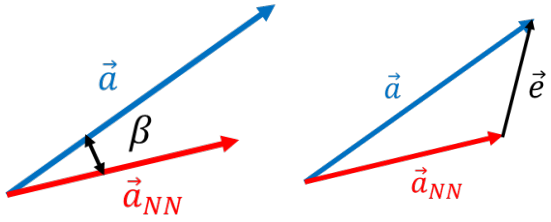
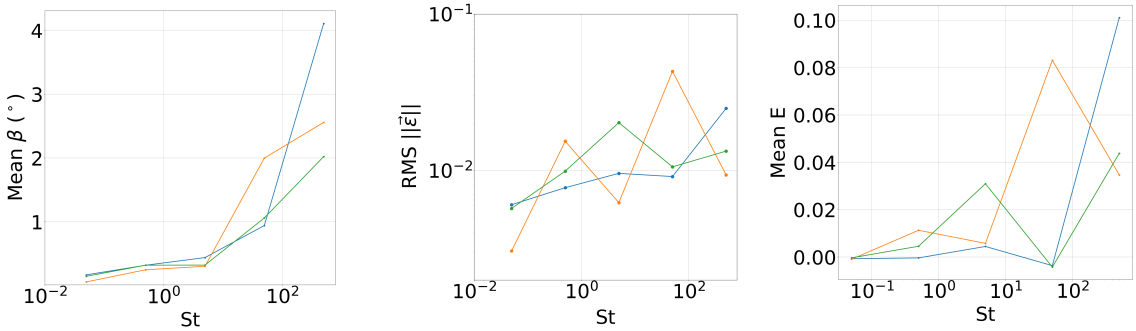


Figure 3.8: ANN error definitions of β and $\mathbf{e}$.

case because $\mathbf{v}_r$ is small in this region of the flow, and is near the edge of the model's working range. Note the varying z-axis (and corresponding colorbar) scales, which show that the maximum directional error for $St = 0.05$ is much smaller than that of $St = 500$. Going one step further, Figure 3.11 shows the forced vortex directional error histograms by St with bin widths of 1°. Similar histograms for stagnation point and irrotational shear flow can be found in Appendix A. Note that Figure 3.11's y-axis is shown with a log scale for the purpose of highlighting the range of β experienced by each St in the respective flow case. The largest β bin for each case is from 0°-1°, and the second largest β bin has a count that is an order of magnitude smaller for every histogram except for the $St = 500$ particles in the forced vortex. The small $St = 0.05$ particles are especially well-characterized for the forced vortex and irrotational shear as the directional error does not exceed 5° for either case. One interesting observation for the $St = 0.5$ particles in irrotational shear (see Appendix A) and the forced vortex is that there is a larger maximum β observed for these than for the $St = 5$ particles which is a break in the trend of increasing maximum β with St. This could possibly be attributed to the more complicated preferential concentration observed in the literature for particles where St $O(1)$ but $St < 1$ [45]. However, this trend was not observed in stagnation point flow.



(a) Mean directional error β. (b) RMS of ϵ vs St. (c) Mean E vs St.

Figure 3.9: Averages of the three error metrics vs St. The results of the forced vortex, irrotational shear, and stagnation point flows are shown with by the orange, green, and blue lines, respectively.

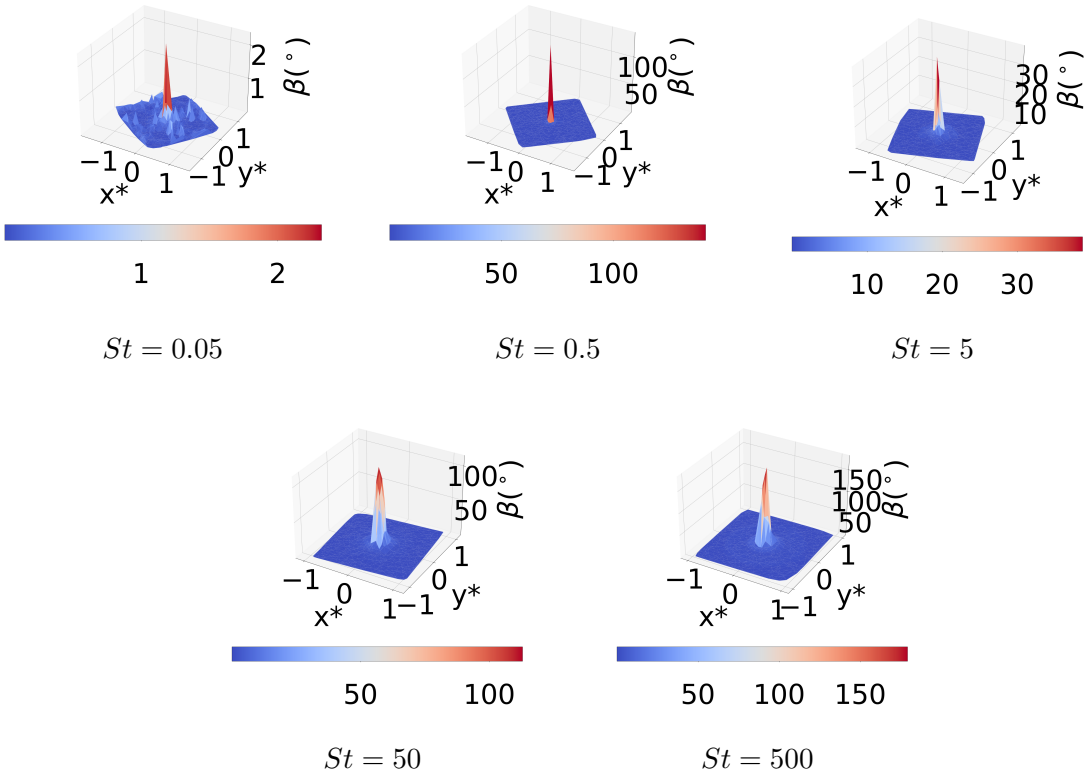


Figure 3.10: Directional error (β) forced vortex position dependence.

Figure 3.9b shows the root mean square (RMS) of the normalized error vector magnitude ϵ vs St for the three single canonical flows. Like β, the RMS error generally increases with St. The most glaring exception shown in the graph is for the forced vortex case at $St = 0.5$ and $St = 50$. Considering the y-axis scale of Figure 3.9b, this could be a reflection of statistical noise. Again, it is useful to know how this error depends on position in the flow and if there are any systematic trends. Figure 3.12 shows the magnitude of $\mathbf{e}$ as a function of $\mathbf{x}_p^*$ for the stagnation point and forced vortex flows. For the stagnation point flow, the highest errors occur along the y-axis primarily near the origin and near the limits of existing particle position. For the stagnation point flow, the y/x axes are streamlines that define and divide the overall motion of the flow. These patterns in the error are thought to be the result of particles deviating from and crossing these streamlines. The forced vortex case exhibits its peak error near the origin for all St and does not suggest error correlation with the streamlines.

Lastly, Figure 3.9c shows an error averaged by St previously defined as E. E describes how magnitude of the ANN-predicted acceleration compares to the actual particle acceleration. If E is negative, the model has under-predicted the acceleration magnitude and vice-versa.

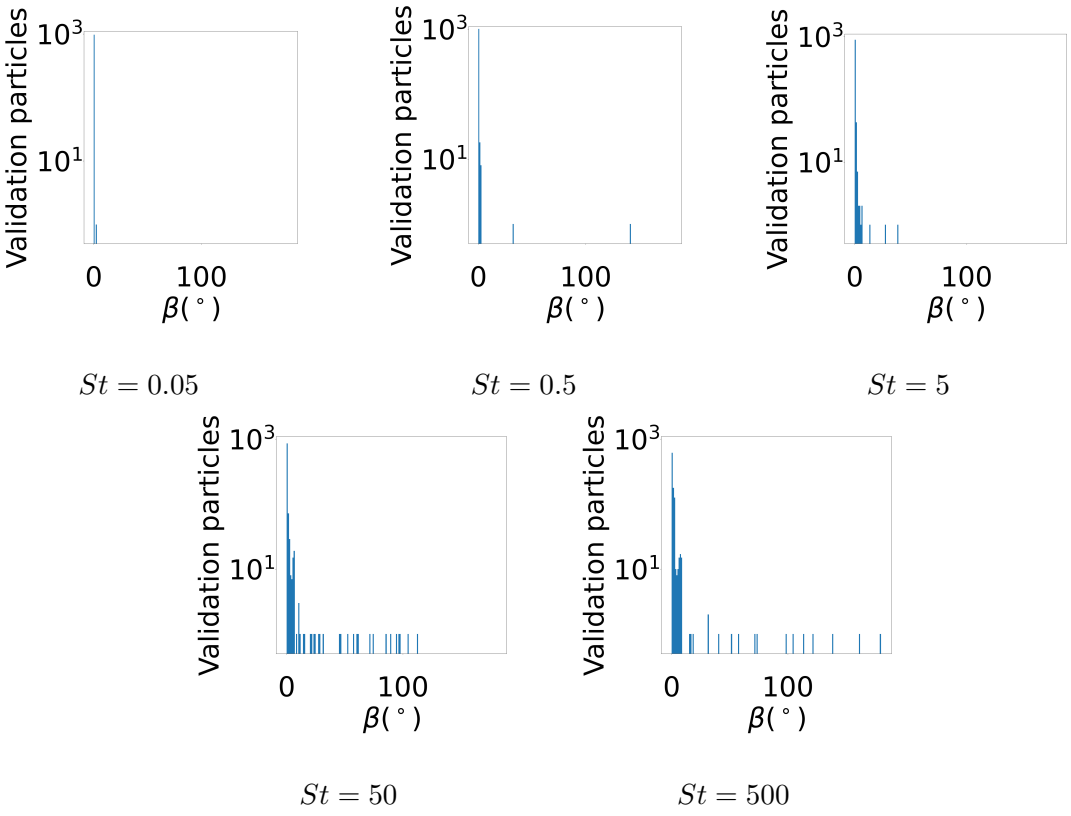


Figure 3.11: Forced vortex directional error (β) distribution. The bin widths are 1°.

While no strong trends can be identified, this plot does suggest that the ANN is producing accelerations on the correct order of magnitude as the maximum average error observed is approximately 0.1 or 10%. This maximum error was found for $St = 0.05$ in stagnation point flow. The stagnation point and irrotational shear seem to follow similar trends, and the forced vortex appears to trend in the opposite manner. Additionally, it does appear that the models generally tend to over-predict **a**.

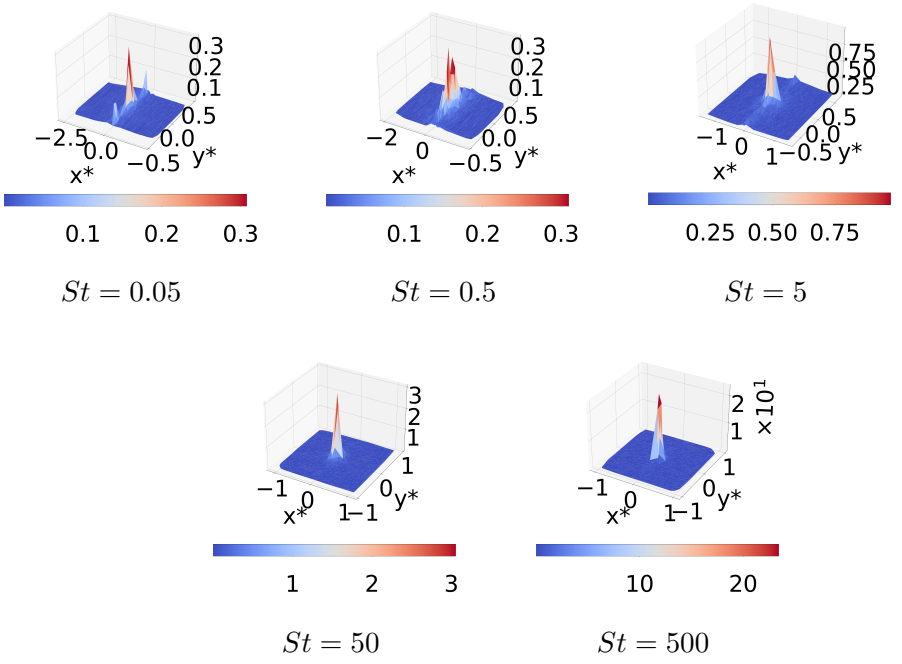
As a means to validate the end goal of predicting particle motion with machine learning, the ANN trained for the forced vortex was substituted for Eq.(2.2) in the LSODA solver. Figure 3.13 shows the ANN's prediction of particle motion for ten particles of each St, and the figures on the left column were initialized with no velocity while the right column figures were all initialized with random velocity in range $\mathbf{v} = [\pm 1, \pm 1]\, m/s$. See Appendix 6.1.1 for the exact initial positions and velocities. The ANN does seem to predict the initial direction of motion for most cases; however, the predicted trajectories quickly diverge. As shown in Figure 3.13, the $St = 500$ case with zero initial velocity matches the governing equation solutions much better than the other zero initial velocity cases. Additionally, the $St = 5$ case with random initial velocity appears to match the governing equation solutions better

than the other random initial velocity cases.

Conclusion

In this chapter, it has been shown that feedforward ANNs with 4 HLs that have a decreasing number of neurons in each HL are capable of predicting particle acceleration for St ranging from 0.05 to 500 with reasonable accuracy. The analysis of the HL structure suggested an optimum configuration in terms of iteration speed and convergence. Furthermore, of the activation functions tested, SELU showed the best convergence performance. The fast, but volatile, convergence of the Adam optimizer is compared to the slower, monotonic convergence of SGD (with and without momentum). Perhaps most interestingly, the results suggested that the ANNs can more easily learn when the vectors in the training data are preprocessed with a transformation to the local streamline coordinate system. Additionally, the surrogate use of the trained ANN in lieu of the particle acceleration governing equation in the LSODA solver shows that the ANN is able to predict general trends for the transient particle position. That is, the initial direction of motion appears consistent for the most part, but the actual trajectories quickly diverge. This is encouraging given that the ANN was only trained with data taken from a single point in time. The largest errors associated with each metric (ϵ, E, and β) were found mainly near the origin, which is near the edge of the model's working range. Additionally, relatively large errors were observed along streamlines, and especially the dividing streamlines of the stagnation point and irrotational shear flows. Each of the error metrics was found to increase with St, which is an undesirable dependence that should be mitigated.

Stagnation



Forced Vortex

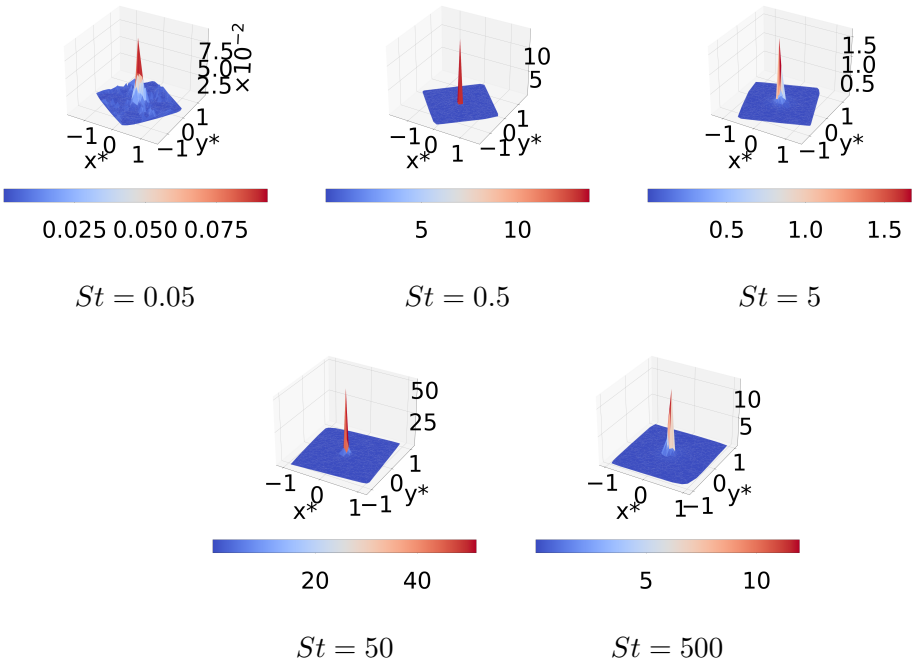
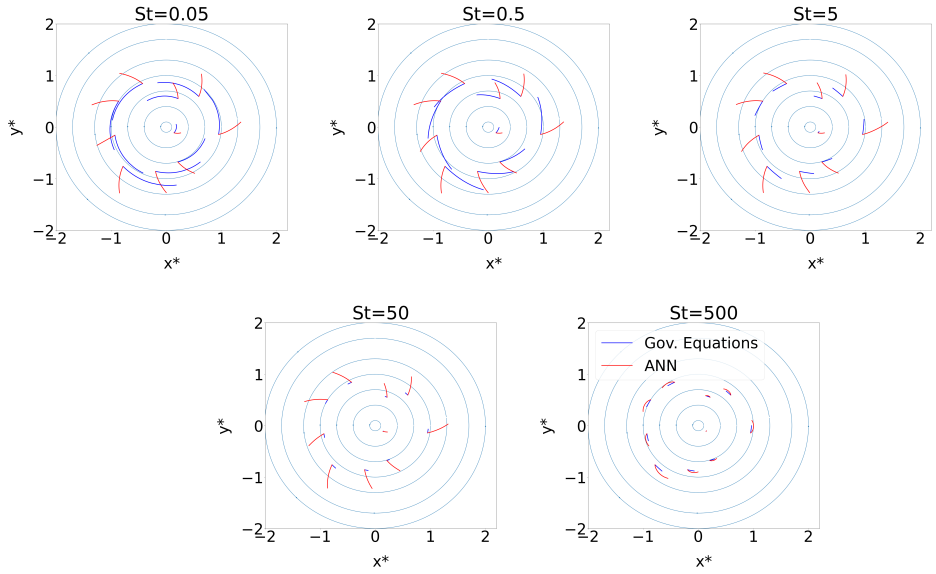
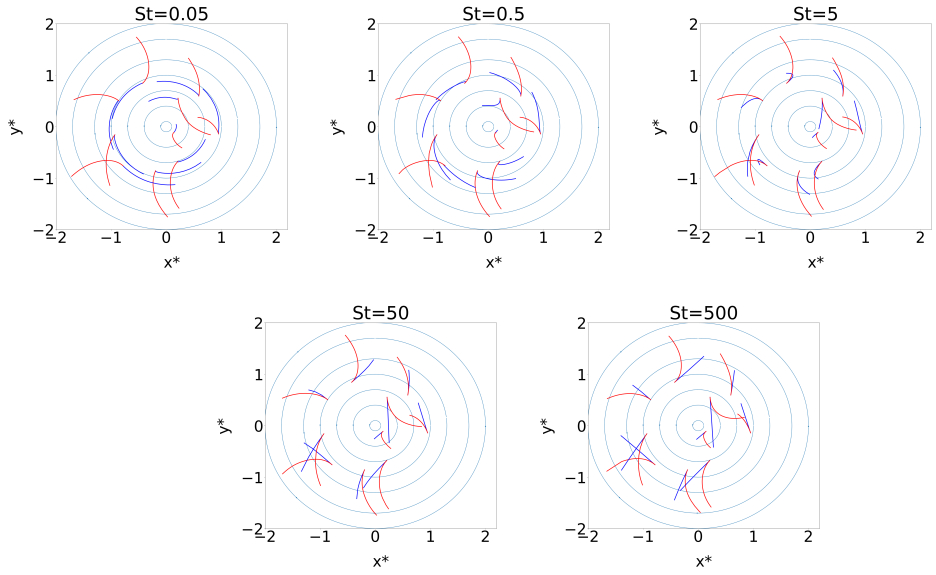


Figure 3.12: ϵ position dependence for stagnation point and forced vortex flows.



(a) $\mathbf{v}(t=0) = [0, 0]$



(b) Random $\mathbf{v}(t=0)$

Figure 3.13: ANN particle motion validation. The counter rotating vortex is indicated by circular streamlines. The particle trajectories of the governing equation solution and ANN approximation are shown by the blue and red lines, respectively. Figure a) shows particles that were initialized with zero velocity, and the particles shown in Figure b) were initialized with a random velocity. All plots show the particle trajectory for $t = [0, 1]$ except for the $St = 500$ zero initial velocity case which is shown for $t = [0, 3]$

Chapter 4

Combined Laminar Flows

Now that the capability of feedforward artificial neural networks (ANNs) to predict particle acceleration in the simplest steady, laminar flows has been demonstrated, the next step is to explore their capability for combined flows. This is motivated by the idea that complex flows can be decomposed according to the relative contributions of strain and rotation. In this section, the ability of ANN's to predict particle accelerations within multiple combined laminar flow composed of the forced vortex and irrotational shear flows. The multiple combinations are differentiated to the ANN via two new metrics that describe the relative contribution of strain and rotation in the velocity gradient tensor. The governing equations are non-dimensionalized to generate the training data such that the ANN should be generalizable. A new ANN structure is presented out of necessity for the more complex function space that it must learn. Section 4.3 introduces an improved physics-inspired loss function (not to be confused with the physics-informed neural network approach of [20]) that considers the nature of vector quantities and is more suitable for training the ANN. The ANN trained using the loss function designed for vector error minimization is analyzed after validation to understand the model's sensitivity to the various inputs. The ANN is also evaluated with particle data from a canonical turbulent flow.

4.1 Generalized Approach

Expanding upon the idea that complex flows can be decomposed according to the relative contributions of rotation and strain at a given point, the forced vortex and irrotational shear flows were combined for a generalized approach. This velocity field is given by $\mathbf{u} = [(e-\omega)x_2, (e+\omega)x_1]$. Similarly to previous discussion, $S = e$, $\Omega = \omega$, and $J = \sqrt{e^2+\omega^2}$. While keeping $J = 1.0s^{-1}$ in accordance with the unit circle concept presented in Figure 2.1, combined flows from $\theta = [0°, 360°)$ in $15°$ increments are considered. Furthermore, to

non-dimensionalize the model inputs the governing equations are modified as

$$\frac{d\mathbf{v}^*}{dt^*} = \frac{1}{2}\frac{\rho_g}{\rho_p}\frac{L_{nd}A_p^* c_D^* ||\mathbf{v}_r^*||^2}{V_p^*}\frac{\mathbf{v}_r^*}{||\mathbf{v}_r^*||} \tag{4.1}$$

$$\frac{d\mathbf{x}_p^*}{dt^*} = \mathbf{v}^* \tag{4.2}$$

$$c_D^* = \frac{24}{Re_p^*}(1 + 0.15(Re_p^*)^{0.687}) \tag{4.3}$$

$$Re_p^* = \frac{\mathbf{v}_r^* d_p^*}{\nu^*} \tag{4.4}$$

where the dynamic viscosity $\nu = \mu_g/\rho_g$, $()^*$ indicates a non-dimensionalized parameter (e.g. $d_p^* = \frac{d_p}{L_{nd}}$). The natural choice for the timescale for which to non-dimensionalize with is $\frac{1}{J}$, and the length scales are non-dimensionalized with $L_{nd} = 1.0cm$ based on the order of magnitude that hailstones normally exist at [42]. The relevant equations are solved in non-dimensional form using $L_{nd} = 1.0cm$ and $\frac{1}{J}$ as the characteristic values. As before, the appropriate data are extracted for 5,000 particles of each St at $t^* = 1.0$ ($t^* = tJ$) for each θ, and 80% of the data are used for training the ANNs and the remaining 20% are used for validation. As before, 5 discrete particle sizes are considered corresponding to $St = 5 * [10^{-2}, 10^{-1}, 10^0, 10^1, 10^2]$. However, the ANN model inputs are altered to reflect the increased complexity of the flow as

$$\frac{d\mathbf{v}^*}{dt^*} = g\left(\mathbf{v}^*, \mathbf{u}^*, d_p, \frac{D\mathbf{u}^*}{Dt^*}, \rho_p, \rho_g, \mu_g, \frac{e}{J}, \frac{\omega}{J}\right) \tag{4.5}$$

such that the model can use the relative contributions of rotation and strain at the particle location as a means of flow feature extraction for general flows. This ANN will be referred to as the Master ANN.

4.2 Combined Flow ANN Validation

The original decreasing hidden layer (HL) size ANN structure proved impractical for training with multiple sets of combined flow data even though the total number of neurons in the network were increased according to Eq.(2.18) and Eq.(2.19) [30]. Therefore, following [44], a rectangular-like block structure ANN was tested and slightly tuned. A rectangular block structure has the same number of neurons in each HL, and this work refers to the structure as rectangular-like because the last two HL are made smaller such that some form of abstract generalization is forced. This ANN structure proved more feasible with the combined flow particle data from $\theta = 15° - 360°$. Figure 4.1 shows the result of training a 7 HL rectangular-like block structure ANN (150 neurons per HL in the first five HL and 100 per HL in the

last two HL) with the Adam optimizer and stochastic gradient descent (SGD) with Nesterov momentum. The Adam optimizer converged reached the stopping criteria of 10^{-5} in around 10.25 hours, while SGD reached a loss of 5.24E−5 after training for 48 hours. It should be noted that the streamline data-driven (DD) preprocessing approach presented in Section 3.1.3 is used for all results shown in this chapter, unless otherwise noted. Specifically, the ANN input vector is $[\hat{v_s}^*, \hat{v_n}^*, \hat{u_s}^*, \hat{d}_p, \frac{D\hat{u_s}^*}{Dt^*}, \frac{D\hat{u_n}^*}{Dt^*}, \hat{\rho}_p, \hat{\rho}_g, \hat{\mu}_g, \frac{\hat{\epsilon}}{J}, \frac{\hat{\omega}}{J}]$, and the model output is $[\frac{d\hat{v_s}^*}{dt^*}, \frac{d\hat{v_s}^*}{dt^*}]$ where the $(\hat{-})$ notation refers to data normalized by the absolute maximum values of the corresponding variable in the data set as previously defined by Eq.(2.24). Thus, all normalized ANN inputs are scaled within range $[-1, 1]$. The subscripts s and n indicate the stream-wise and normal vector components, respectively. That is, $\mathbf{v} = v_s\hat{s} + v_n\hat{n}_s$, where the unit vectors $\hat{s}$ and $\hat{n}_s$ are defined in Section 2.3. It should be noted that $\hat{u_s}^* = ||\hat{\mathbf{u}}^*||$ because the streamline coordinate $\hat{s} = \frac{\mathbf{u}}{||\mathbf{u}||}$ is defined as the unit vector in the direction of fluid velocity.

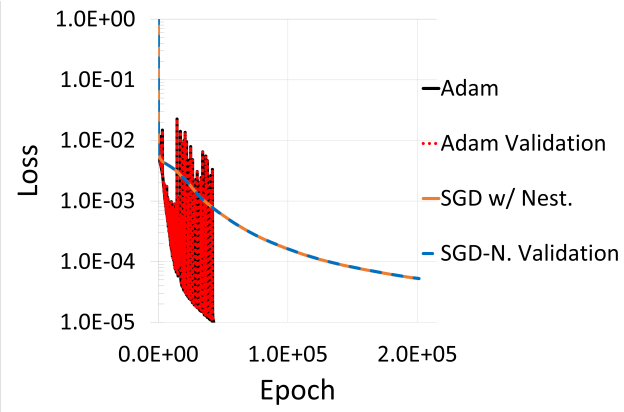


Figure 4.1: Combined flow ANN training with Adam and SGD optimizers. Validation loss shown for reference.

The following data analysis comes from the Master ANN trained with the Adam optimizer as shown in Figure 4.1. As before, the directional error (β, see Eq.(3.1)) is first considered. Figure 4.2 shows β as a function of $\mathbf{x}_p^*$ for the five different Stokes number (St) particles considered. The Master ANN seems to be much more adept at predicting the acceleration direction for the smaller particles. For the three largest particle categories, it appears that about half of the model predictions are in the exact opposite direction of the actual acceleration ($\beta = 180°$). Figure 4.3 shows the histograms of β by St and confirms the previous observation. For $St = 0.05$ and $St = 0.5$, the distribution is concentrated around 0°. For $St = 5$ and above, a bimodal distribution is observed, where the concentrated regions are around 0° and 180°. While the model performance is clearly sub-par, it is encouraging to see that most densely populated region of the histogram is near the lower error.

Additionally, $||\boldsymbol{\epsilon}|| = \epsilon$ is shown as a function of $\mathbf{x}_p^*$ by St in Figure 4.4. Note that a more tilted view of the scatter plot is given for $St = 0.05$ and $St = 0.5$ because the isometric

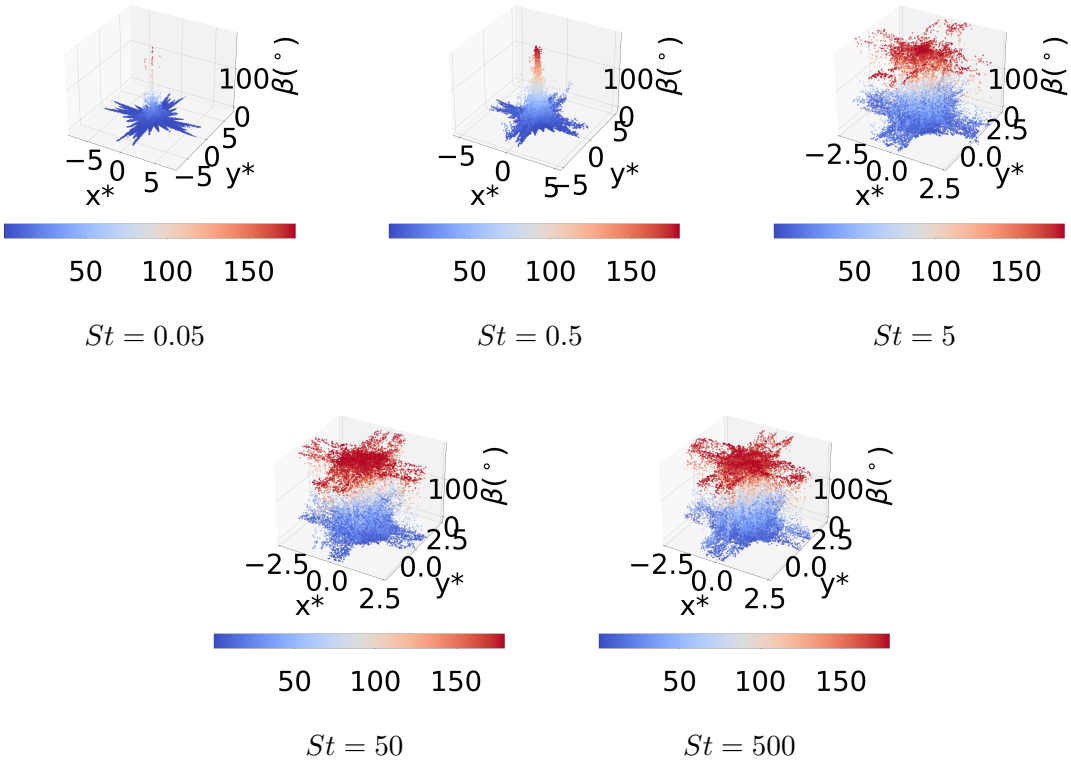


Figure 4.2: Combined flow directional error (β) position dependence.

view is not easy to interpret for these St. While these errors are much higher than observed in the single flow cases, several interesting trends can be observed. First, the magnitude of $\boldsymbol{\epsilon}$ generally increases with St. However, it should also be noted that the $St = 0.5$ case produced several outliers higher than any value observed in the $St = 5$ case. This suggests that the complex preferential concentration at St $O(1) < 1$ previously mentioned may be affecting the ANN's performance. Additionally, many of the peaks observed on these error plots occur near the origin, which agrees well with the results shown in Figure 3.12. Beyond this, it is not possible to identify error dependence on position because all 24 variations of combined flow ($0° < \theta < 360°$) are shown in this plot.

It follows that the error dependence on θ, as defined in Figure 2.1 is examined. As shown in Figure 4.5a, the error β appears to have some dependence on θ for $St = 0.05$ and $St = 0.5$. For these St, error decreases as θ increases from 0° to 180° and then increases from 180° to 360°. This does not indicate any preference by the ANN for flow dominated by strain or rotation, as $\theta = 0°$ and $\theta = 180°$ are both irrotational. It could simply mean that the ANN has some implicit numerical preference for the streamlines configured with $\theta = 180°$. Additionally, while not as strong as the smaller St, $St = 5$ appears to exhibit some symmetry

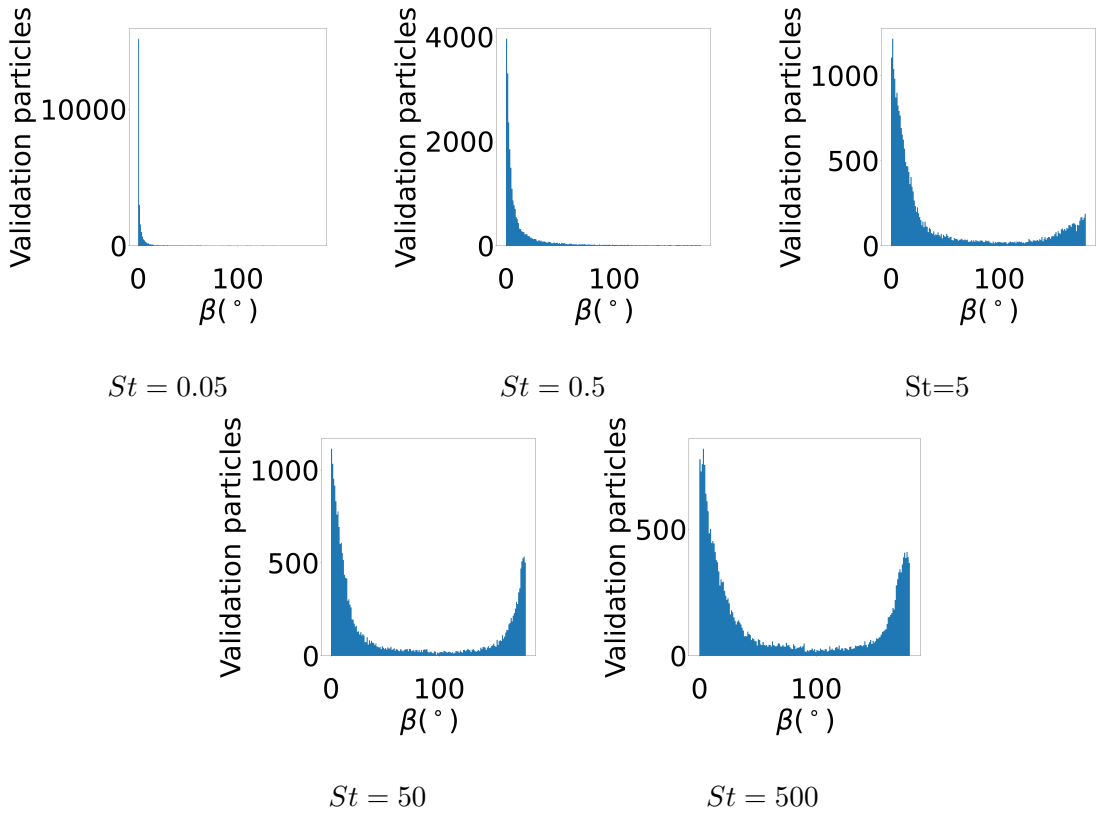


Figure 4.3: Combined flow directional error distribution. The bin widths are 1°.

about $\theta = 180°$.

Figure 4.5b shows the the RMS of ϵ vs θ by St. The RMS of ϵ is again shown to increase with St. Similarly to the trends observed in Figure 4.5a, the error for $St = 0.05$ and $St = 0.5$ exhibits some symmetry about $\theta = 180°$, and the higher St have no discernible dependence on θ.

Lastly, Figure 4.5c shows the mean magnitude error E vs θ by St. Deviating from the trends previously observed, $St = 0.05$ appears to have no dependence on θ and remains around 0. Similarly to Figures 4.5a and 4.5b, $St = 5, 50$ and 500 do not appear to depend on θ. However, this plot does show that the ANN consistently over-predicts the accelerations for these three St, on average.

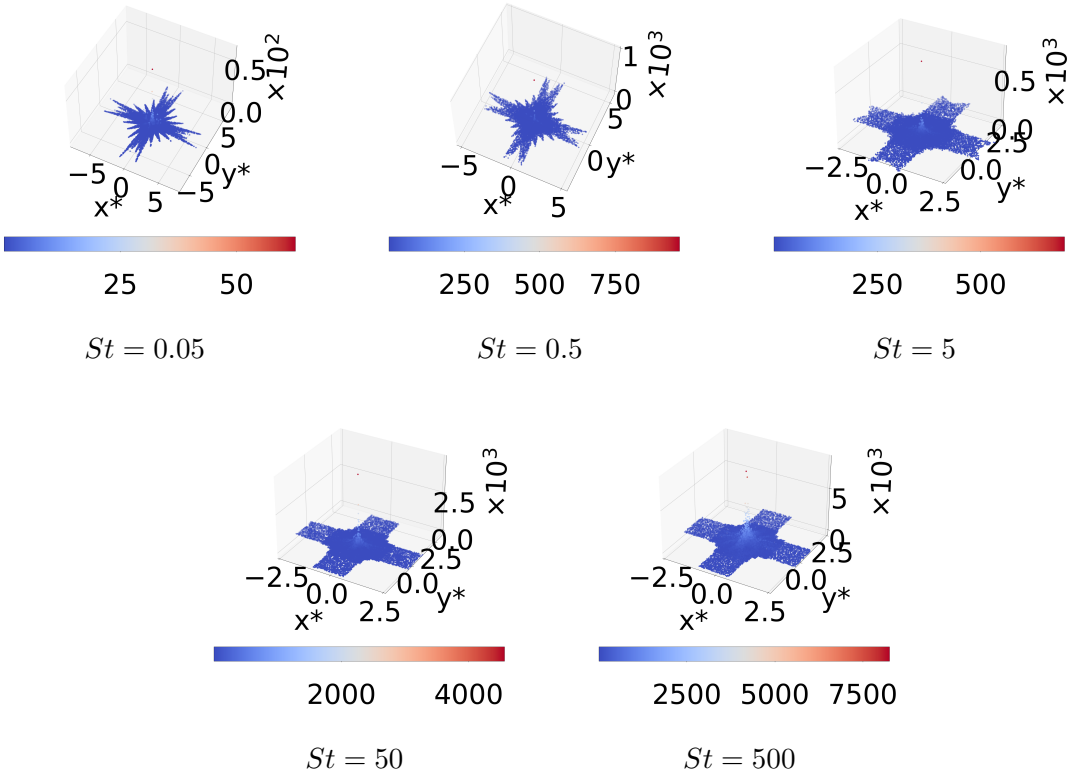
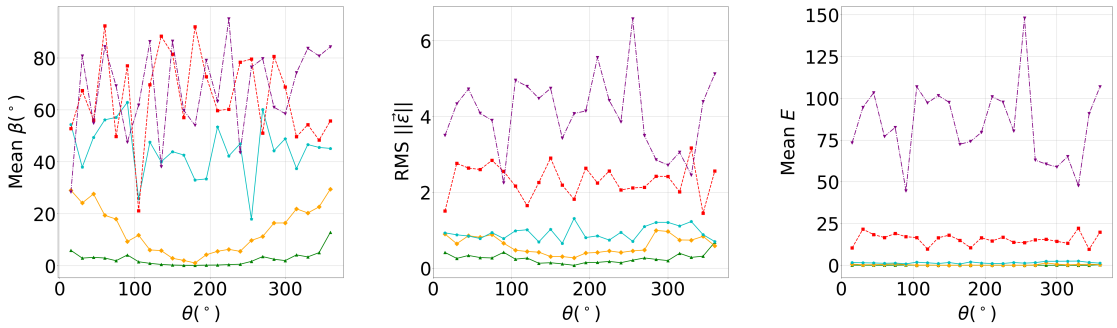


Figure 4.4: Combined flow ϵ position dependence.



(a) β vs. θ (b) RMS of ϵ vs. θ (c) Mean E vs. θ

Figure 4.5: Combined flow errors for $St = 0.05$ (▲), 0.5 (◆), 5 (⬟), 50 (■) and 500 (▼).

4.3 ANN with Improved Loss Function

In order to improve training of the ANN, it was thought necessary to consider what error is being computed in the loss function which the ANN is attempting to minimize. Rather than using the pure mean squared error (MSE) between the actual and ANN predicted accelerations, a custom loss function still using the mean squared (MS) criteria (also known as the squared L_2 norm) was defined to account for the physical nature of acceleration. In order to equally penalize the weights and biases for accelerations of small and large particles, (small particles will experience much larger accelerations compared to the large particles) a custom loss function is defined for the ANN. Initially, a loss function defined as $loss = c_1 * \mathrm{MS}(E) + c_2 * \mathrm{MS}(1 - \cos\beta)$ was used, where c_1 and c_2 are hyperparameters that should be tuned. However, this approach did not work well as it is impossible to know how E scales *a priori*. Thus, a custom loss function that penalizes the ANN based on acceleration direction and magnitude *relative* to the actual acceleration was defined as shown in Eq.(4.6) and Eq.(4.7). Additionally, the magnitude of $\boldsymbol{\epsilon}$ is defined as ϵ.

$$\boldsymbol{\epsilon} = \frac{\mathbf{e}}{||\mathbf{a}||} \tag{4.6}$$

$$loss = \mathrm{MS}(\epsilon) \tag{4.7}$$

Initial tests with the same ANN structure as before using the loss function defined with E and β (where $c_1 = c_2 = 1$) were used to show the effect of using a linear output layer. A linear output layer means that effectively no activation is applied to the output layer, or formally the activation function can be defined as $\sigma(z) = z$. The idea behind trying this configuration was to ensure that the SELU activation function was not limiting the output, as the model is effectively a high order regression model. Clearly, the linear output allowed the model to train at a faster rate per epoch as shown in Figure 4.6. It also nearly tripled the number of iterations possible, as the two ANNs were trained for the same amount of time. This is likely due to reduced computational expense in the back-propagation calculations as the derivative of SELU is not cheap.

It was also found that manually limiting the learning rate (lr) was conducive to further convergence. This was surprising based on the fact that the Adam optimizer computes adaptive learning rates of its own for various parameters [34]. As the convergence appeared to level off, the lr was decreased by a factor of 10. Additionally, because the SELU activation function was being used, it was of interest to know if SELU's self-normalizing properties [44] would be effective for this application. All studies before this had been trained using normalized data, meaning that each input parameter had been divided by the maximum value in the data set as shown in Eq.(2.24). Figures 4.7a and 4.7b show the ANN trained with normalized data and unnormalized data, respectively. For the unnormalized data, the ANN input vector is modified as $[{v_s}^*, {v_n}^*, {u_s}^*, d_p, \frac{D{u_s}^*}{Dt^*}, \frac{D{u_n}^*}{Dt^*}, \rho_p, \rho_g, \mu_g, \frac{e}{J}, \frac{\omega}{J}]$, and the model output is modified as $[\frac{d{v_s}^*}{dt^*}, \frac{d{v_n}^*}{dt^*}]$ (Eq.(2.24) is no longer employed). The unnormalized, or full-scale, data allowed the network to achieve a final training loss an order of magnitude lower than with the normalized data set.

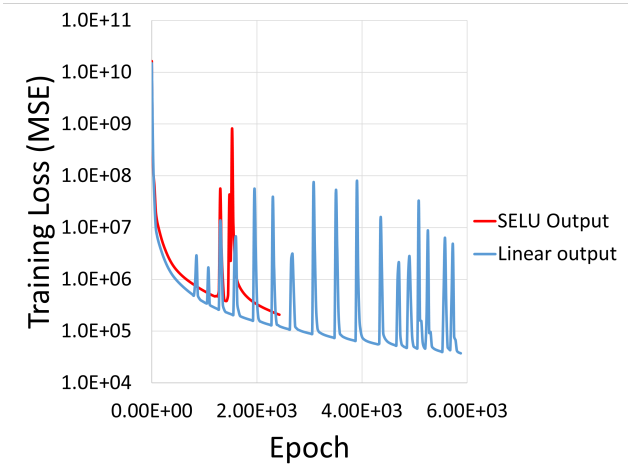
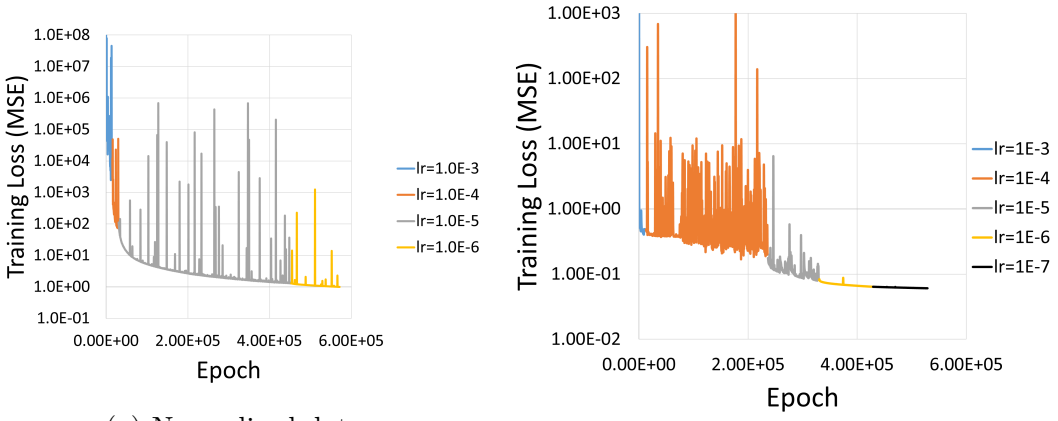


Figure 4.6: 7 HL ANN Training MSE ($loss = \text{MS}(E) + \text{MS}(1 - \cos\beta)$) with lr=1E-4. Each of HLs 1-6 are exactly the same, but HL 7's activation function is set as SELU for the red line and $\sigma(z) = z$ for the blue line.



(a) Normalized data.

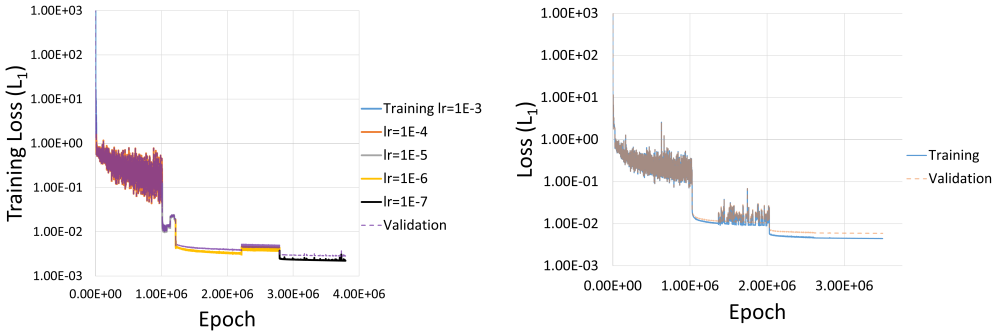
(b) Unnormalized data.

Figure 4.7: Comparison of training the improved ANN with normalized and unnormalized data while iteratively limiting the lr.

Now that the use of unnormalized data has been established as the better practice, the loss function was again of interest. Through all of the previous studies, the MSE loss function (the squared L_2 norm) had been used. The L_1 norm, or mean absolute error (MAE), as defined by Eq.(4.8) was implemented to determine if the ANN could better approximate the particle accelerations when trained with a fundamentally different loss function. Figures 4.8a shows the result of training the network with Eq.(4.8). The final L_1 training loss achieved with the unnormalized streamline data was 2.2E-3 as shown in Table 4.1, which is an order of magnitude lower than the same network trained using the MSE loss. Additionally, since

the ANN structure had changed considerably since the initial test to determine the best data preprocessing, another test was in order. The same ANN was retrained with the Cartesian data set, and the resulting training and validation trends are shown in Figure 4.8b. As expected, the final training loss was higher, in fact about twice as large at 4.4E-3, than the streamline data test as shown in Table 4.1. This agrees with and bolsters the results shown in Section 3.1.3. The validation loss for the Cartesian data test is also proportionally higher than that of the streamline data.

$$L_1(\boldsymbol{\epsilon}) = \frac{\sum_{j=1}^{N} \sum_{i=1}^{m} ||\epsilon_i||}{N} \tag{4.8}$$



(a) Unnormalized streamline data. (b) Unnormalized Cartesian data.

Figure 4.8: Use of L_1 loss function. The L_1 loss function is less sensitive to outliers and allows the ANN to achieve a lower loss. Note that the lr in b) was iteratively limited similarly to a).

Table 4.1: Master ANN Data Preprocessing Effects. Trained with Adam using SELU activation function and custom L_1 loss function. Final training and validation loss after iteratively decreasing lr to 1.0E-7 is shown.

Coordinate System	Training Loss	Validation Loss
Cartesian	4.45E-03	5.91E-03
Streamline	2.22E-03	2.83E-03

4.3.1 Improved ANN Validation

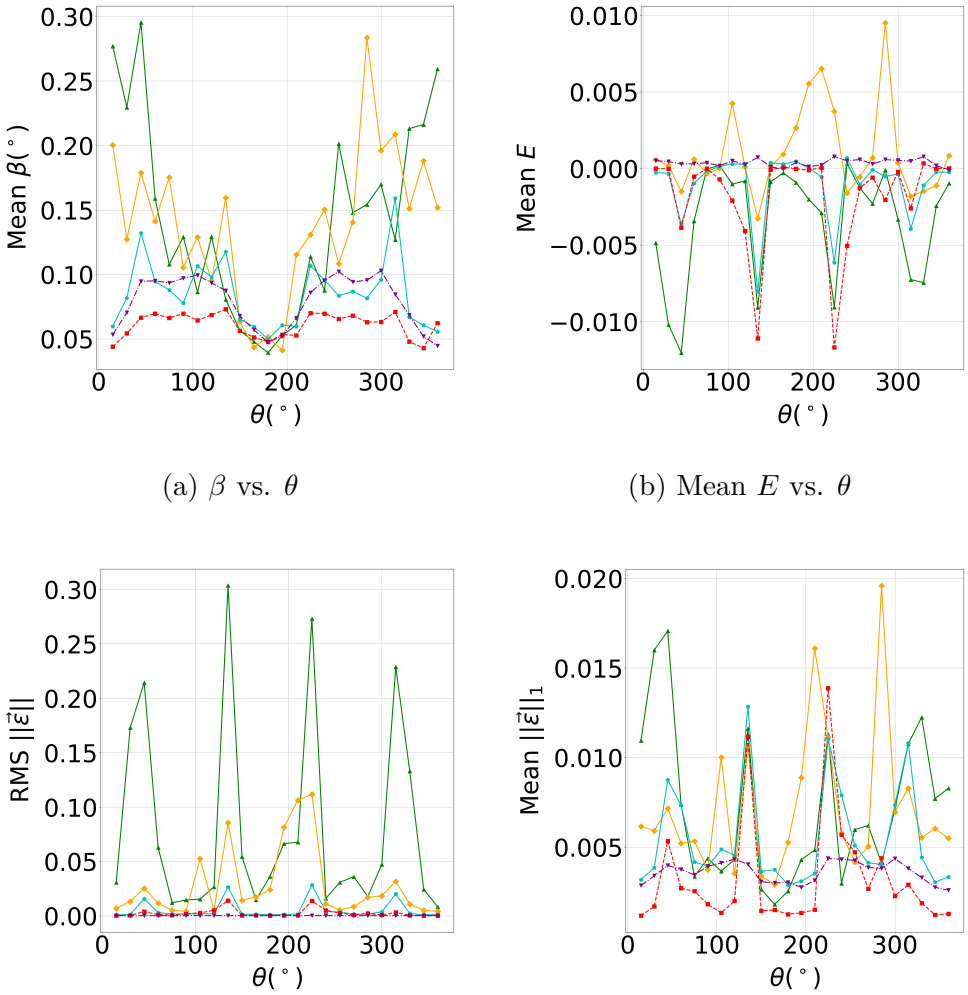
Using the trained ANN from Figure 4.8a, the validation data were analyzed in detail as before as shown in Figure 4.9. This time, however, it was appropriate to include a plot of

the L_1 error vs. θ norm as shown in Figure 4.9d. Much like in Figure 4.5a, the smallest two St exhibit some symmetry about $\theta = 180°$, but for this ANN all St directional errors have some amount of symmetry about $\theta = 180°$. Additionally, the values of β are much more acceptable than before. The maximum average β for any combination of St and θ does not exceed $0.3°$, while previous results for the larger St had most average β on the order of $60 - 80°$. As shown in Figure 4.9b, the mean absolute value of E only slightly exceeds 0.01 for three combinations of St and θ. This means, on average, the predicted acceleration magnitude is within 1% of $\mathbf{a}$. In fact, most of the values are below 0.005. The main values of θ where the peaks occur are around $45°, 135°, 225°$, and $315°$. For each of these values of θ, $|\omega| = |e|$ and it can therefore be shown that $\frac{D\mathbf{u}}{Dt} = 0$. The model likely has difficulty dealing with zeros in the inputs $\frac{D\mathbf{u}}{Dt}$ as it likely heavily relies on it for all other data points. Similar trends can be observed in Figure 4.9c, and it should also be noted that these errors are generally being observed to decrease with St. This is the exact opposite trend that was observed earlier. In Figure 4.9d, the dependence of the L_1 norm of $\boldsymbol{\epsilon}$ on St appears to be weakened. This could be for two reasons. Since the ANN was trained with this loss function, it may have successfully balanced the loss from each St data set equally. Alternatively, it could be simply that the L_1 norm is less sensitive to outliers.

As before, histograms of the directional error β were plotted by St as shown in Figure 4.10. Note that unlike the histograms shown earlier in this chapter, the y axis is displayed with a log scale to indicate the largest errors that occurred. For each St, the maximum error is no more than $65°$. The histograms are remarkably similar, which is likely an ode to the custom loss function. For each case, the fullest bin is the $0° - 1°$, which in each case holds clearly over 10^4 validation particles.

Particle Trajectories

In addition to analyzing the Master ANN performance with the validation data, the ANN was tested as a surrogate for the governing equation Eq.(4.1) as in Section 3.2. First, using $\theta = 90°$ to serve as a useful comparison to the ANN trained for the forced vortex in Section 3, the surrogate Master ANN is used to solve the same particle positions with the same initial conditions presented in Figure 3.13 (see Appendix 6.1.1), except it is compared to the results of solving non-dimensional governing equations given by Eq.(4.1) and Eq.(4.2). The results for all St particles considered in $\theta = 90°$ are shown in Figure 4.11. For the zero initial velocity case in Figure 4.11a, the particle track predicted with the ANN appears to be very short compared to the actual plot of $\mathbf{x}_p(t)$ for the smaller St particles. For $St = 50, 500$, the governing equations particle tracks are also very short, and will be explored in more detail. For those particles given a random initial velocity shown in Figure 4.11b, the ANN appears to perform well for six of the ten $St = 0.05$ particles considered. Additional flows corresponding to various values of θ are shown for $St = 0.05$ particles with random initial velocities in Appendix 6.1.1. However, for the other St considered, the use of $\mathbf{a}_{NN}$ appears to result in very similar particle location paths as the $St = 0.05$ particles. To better understand



(a) β vs. θ

(b) Mean E vs. θ

(c) RMS of ϵ vs. θ

(d) L_1 norm of $\boldsymbol{\epsilon}$ vs θ

Figure 4.9: Improved ANN combined flow errors for $St = 0.05$ (▲), 0.5 (◆), 5 (⬟), 50 (■) and 500 (▼).

the performance of the Master ANN for the larger St with $\mathbf{v}_0 = [0, 0]$, a single particle track (particle 1 as listed in Appendix 6.1.1) is shown in detail for $St = 50$ and $St = 500$ in Figure 4.12. Clearly, the Master ANN performs well for the $St = 500$ particle shown while less than half of the particle track is predicted for the $St = 50$ particle. The superior performance of the ANN for $St = 0.05$ particles with an initial velocity and the $St = 500$ particles with zero initial velocity may be an indicator that the ANN is not truly deciphering the dependence of $\mathbf{a}$ on d_p. The ANN was trained using data extracted at $t^* = 1.0$ from the combined canonical flows where the particles were initialized with $\mathbf{v}_0 = [0, 0]$. This means that the larger particles shown to the ANN would have a much lower velocity at $t^* = 1.0$ compared

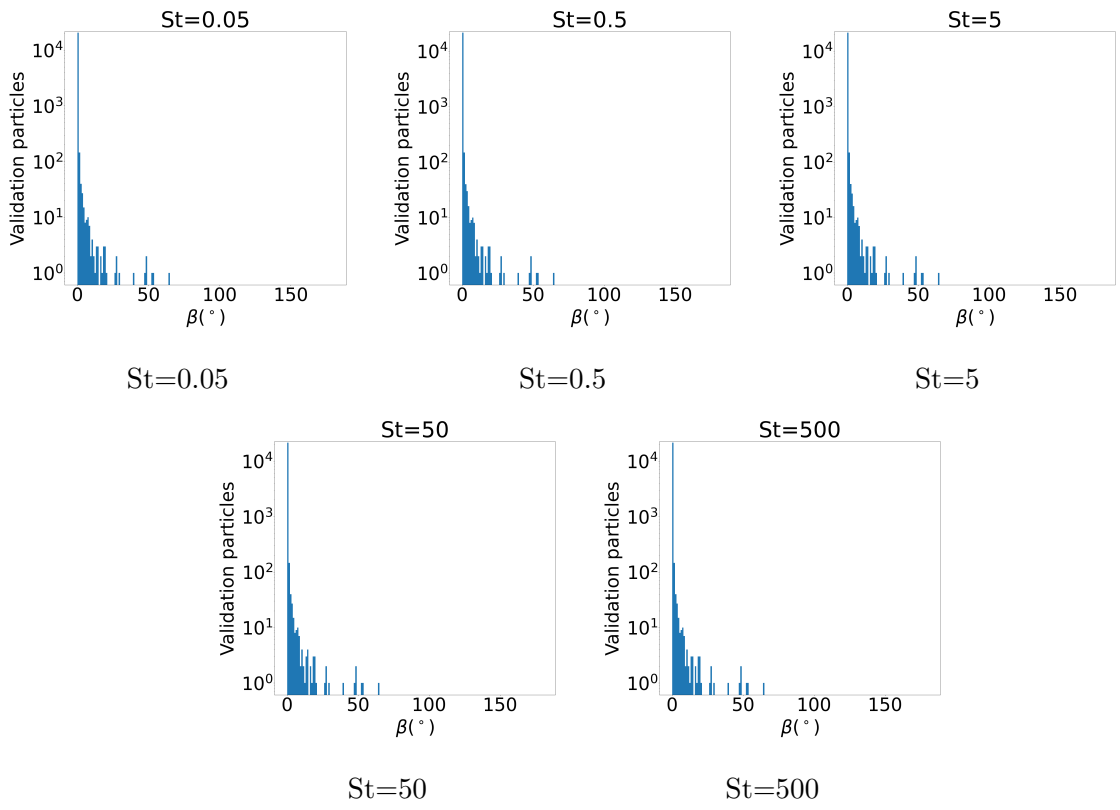


Figure 4.10: Combined flow directional error distribution for the improved ANN. The bin widths are 1°

to the smaller particles since large particles experience smaller acceleration in the same flow. As a result, the ANN may have placed unwarranted weight on acceleration's dependence on particle velocity. Thus, the particles initialized with zero velocity in Figure 4.11a were all determined to have small accelerations and subsequently small displacements. Similarly, the particles initialized with a random velocity were predicted to have large accelerations in the direction of the flow regardless of particle size as shown in Figure 4.11b. This supports the idea that the Master ANN was over-trained based on $\mathbf{v}$ and did not decipher the influence of d_p.

ANN Model Sensitivity

To further validate the idea that the ML model may have over-trained the dependence of $\mathbf{a}$ on $\mathbf{v}$, a simple sensitivity study was conducted. The automatic differentiation package in *PyTorch* was used to compute the sum of the partial derivatives of the two model outputs with respect to each input. Figure 4.13 shows a box and whisker plot (fliers excluded

for figure clarity) of this quantity for the entire data set (training and validation) for each model input. Clearly, the model is most sensitive to the stream-wise and normal particle velocity components. While the model does show sensitivity to the particle diameter d_p, it is less dependent on this than the gas velocity magnitude and normal acceleration component. Additionally, it is not surprising that almost no dependence is shown on the physical properties ρ_p, ρ_g, and μ_g because these were constant throughout the data. Interestingly, the model did not develop any significant dependence on the non-dimensionalized contributions of strain and rotation, $\frac{e}{J}$ and $\frac{\omega}{J}$, respectively. This may suggest that the idea of decomposing complex flows into simple canonical flows is fundamentally flawed. For one-way coupled flow, the particle trajectory may be dominated by local flow information such that the flow gradients do not affect the instantaneous acceleration $\mathbf{a}$. Alternatively, because the data set was constructed from particle data at $t^* = 1.0$, it is also possible that this data set did not demonstrate the dependence of $\mathbf{a}$ on $\frac{e}{J}$ and $\frac{\omega}{J}$. If this is the case, the lack of dependence shown on the tensor invariants is likely due to over-training on $\mathbf{v}$.

4.3.2 Homogeneous Isotropic Turbulence

Any flow can be decomposed into a purely rotational and purely straining component, in theory. However, in practice, the use of laminar rotational and strain may or may not be an appropriate model for real flows. Part of the complication is that while the network was primarily trained for steady laminar flows, real flows often become unsteady and turbulent. To test the efficacy of the neural network in turbulent flows, a canonical turbulent case must be generated. One such case is homogeneous isotropic turbulence (HIT). HIT is an idealized turbulent flow that has the property that its statistics are invariant with respect to position (homogeneous) and orientation (isotropic). In real flow, HIT is often a good approximation of flow behavior away from boundaries. In practice, HIT is a canonical flow type that is used to study and gain knowledge of turbulent flows [46].

An advantage of HIT compared to other flow types is that it is easy to generate such a flow field computationally. Strategies for generating such a flow are discussed in [47]. This work uses an improved version of that strategy from [48]. One disadvantage of HIT as a testing mechanism is that true HIT is inherently three dimensional, whereas the neural net developed has been developed in two dimensions. In addition to the true 3D HIT, simulations have also been performed with a 2D version of the HIT approach from [48].

For HIT there are two natural time scales [46]. The time scale of the smallest and fastest turbulent structure is given by the Kolmogorov time scale τ_k. The time scale of the largest and slowest timescales is given by the integral timescale, τ_{HIT}. Here the appropriate choice of τ_f for computing St is τ_k. Better predictions can be made if the particle response time is corrected with the Schiller-Naumann drag model to be $\tau_{p,HIT}$ as defined in Eq.(4.9) [49] where u' is used for the characteristic velocity. As discussed in [47], for the forcing technique deployed here, the integral length ℓ is around 40% of the domain size, and the mesh size is

chosen to be about half the Kolmogorov length scale to ensure proper grid resolution.

$$\tau_{p,HIT} = \frac{\tau_p}{1 + 0.15Re_p^{0.687}} \tag{4.9}$$

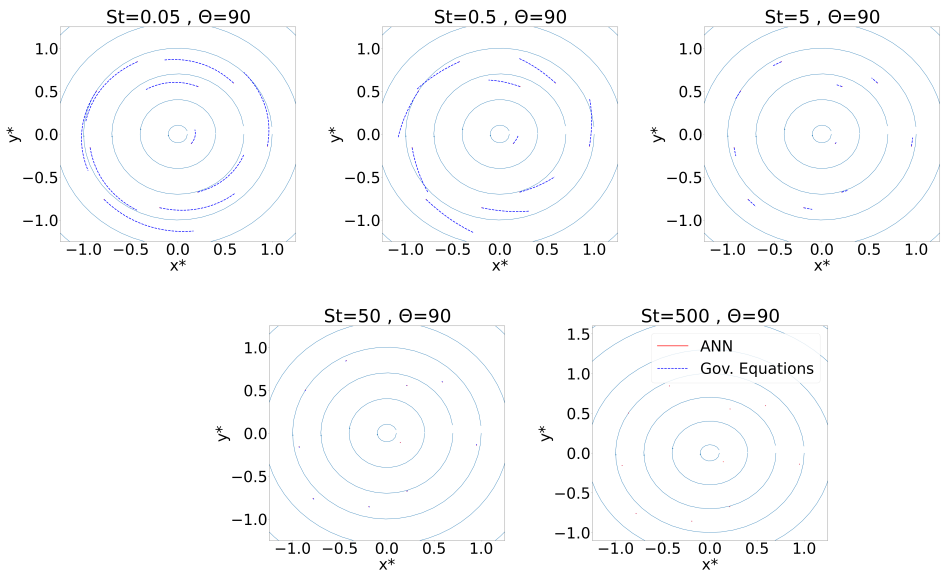
For HIT simulations in this study, Eulerian-based gas phase transport equations and Lagrangian-based particle tracking equations (Eq.(2.1) & Eq.(2.2)) are utilized for the unique challenges that particle laden flows present [50] [51]. This Euler-Lagrange method uses Direct Numerical Simulation (DNS) to solve the transport equations, which are given by the Navier-Stokes equations. 1000 particles each of $St = 2 * [10^{-2}, 10^{-1}, 10^{0}, 10^{1}]$ are considered in a cubic domain. Particle data are extracted at multiples of τ_k to evaluate the master ANN.

It is not surprising that the neural network trained with 2D data does not do a great job of capturing the behavior of the 3D HIT. Since it cannot predict motions out of plane, the predicted acceleration will almost always be incorrect. The three error metrics for the first ANN presented in Section 4.2 are shown as a function of θ in Figure 4.14. Figure 4.14a shows the directional error vs θ by St. The average directional error is shown to be no less than 80°. Interestingly, the highest error is associated with the smallest St in this case. No obvious trends with θ are observed other than β for St = 0.2, 2, and 20 increase as θ approaches 90°, which corresponds to flow where rotation is dominant. The $St = 0.02$ directional error decreases as θ approaches 90°. Figure 4.14b shows the average magnitude of $\mathbf{e}$ as a function of θ by St. $St = 2$ results in the highest overall error for this metric and appears to be increasing with θ. The other three St do not appear to have any direct correlation with θ; however, they do seem to decrease as θ approaches 90°. Additionally, it should be noted that a large spike in error occurs around $\theta = 60°$ for $St = 20$. Lastly, Figure 4.14c shows the mean of E as a function of θ by St. The lowest and highest errors for this metric occur for $St = 0.2$ and $St = 2$, respectively. The data for these two St, while different in value, follow similar trends. Interestingly, the data for $St = 0.02$ and $St = 20$ also follow the same overall trend, although not as closely.

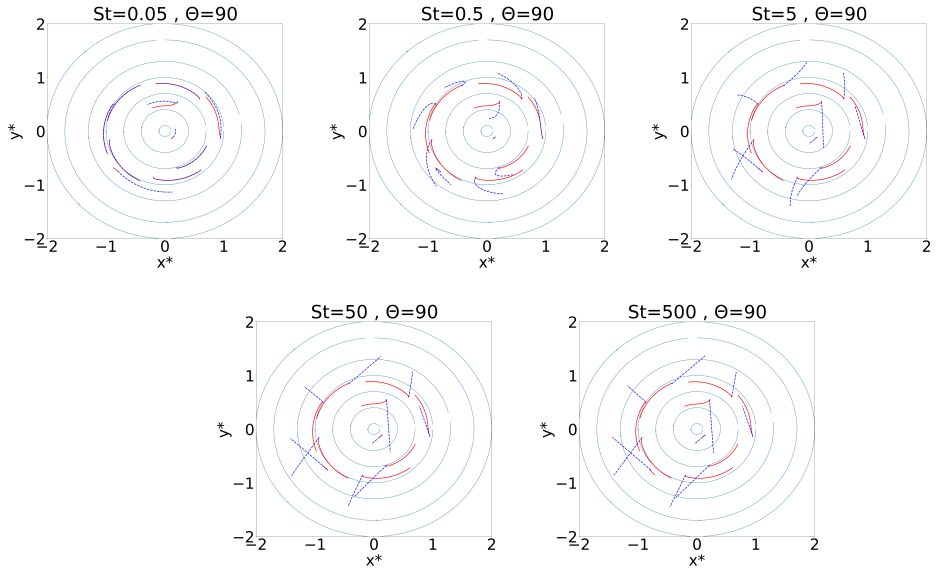
The three error metrics were evaluated once again for the improved Master ANN, and the results are shown in Figure 4.15. It appears as though the improved ANN did in fact decrease the directional errors (β) significantly for all St except $St = 0.2$ as shown in Figure 4.15a. This could be once again indicative of the more complex preferential concentrations observed at $St\ O(1) < 1$ [45]. However, the predicted magnitude of $\mathbf{a}_{NN}$ appears to have worsened as shown in Figures 4.15b - 4.15e, perhaps suggesting data scaling or non-dimensionalization issues. In fact, the lowest observed magnitude errors suggest that the ANN predictions are 500% larger than the true particle acceleration $\mathbf{a}$. While it is encouraging that the improved ANN predictions are better indicators of the true direction of $\mathbf{a}$, it is still not surprising that the observed errors are high since this ANN was trained with 2D particle flow data.

Conclusion

Before continuing to create particle data sets in combined flow cases, the governing equations were first non-dimensionalized in hopes to generalize the usability of the ANN to be trained. When trained using the pure MSE of $\mathbf{a}$ and $\mathbf{a}_{NN}$, the rectangular-like structure of the 7 HL ANN, or Master ANN, presented in this chapter was able to learn the function space for the smallest two St with some degree of accuracy, but performed poorly for $St = 5 - 500$. Not surprisingly, it also was unable to predict the accelerations of particles in 3D HIT flow. However, removing the SELU activation function from HL 7, using unnormalized data to train the ANN, and introducing a custom loss function that penalizes the weights and biases on the L_1 norm of the normalized error vector $\boldsymbol{\epsilon}$ improved the network's accuracy significantly. The maximum average validation errors in magnitude went from 15000% to around 1%, while the maximum directional error β went from 180° to approximately 65° (but was less than 0.3° on average for any St and θ). Additionally, the improved ANN certainly weakened the error metrics' dependence on St and actually reversed the nature of the dependence. The original ANN saw errors that increased with St, while the improved ANN observed decreasing error with St.



(a) $\mathbf{v}^*(t^* = 0) = [0, 0]$



(b) Random $\mathbf{v}^*(t^* = 0)$

Figure 4.11: Master ANN particle trajectory validation for $\theta = 90°$, which corresponds to the counter rotating forced vortex. The vortex is indicated by circular streamlines. The particle trajectories of the governing equation solution and ANN approximation are shown by the dashed blue and solid red lines, respectively. Figure a) shows particles that were initialized with zero velocity, and the particles shown in Figure b) were initialized at the same position but with a random velocity with x and y components in range $[-1, 1]$. All plots show the particle trajectory for $t^* = [0, 1]$.

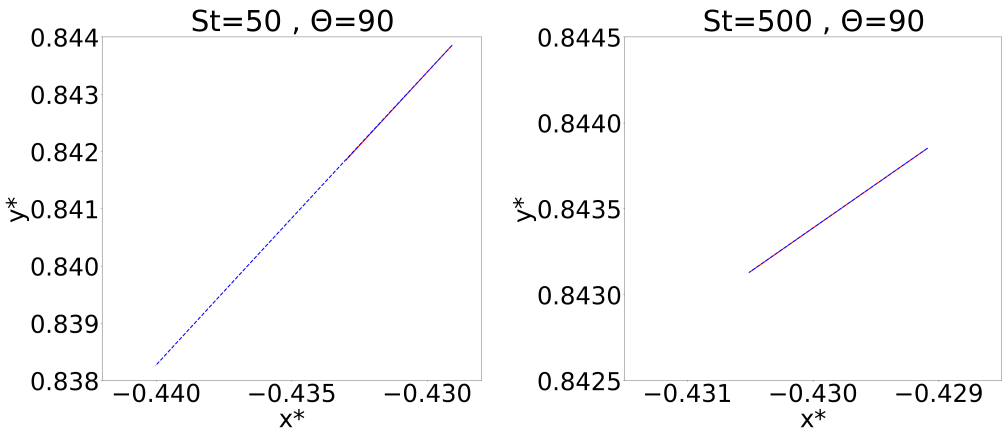


Figure 4.12: Master ANN closeups of large St single particle tracks from Figure 4.11. The particle trajectories of the governing equation solution and ANN approximation for $t^* = [0, 1]$ are shown by the dashed blue and solid red lines, respectively.

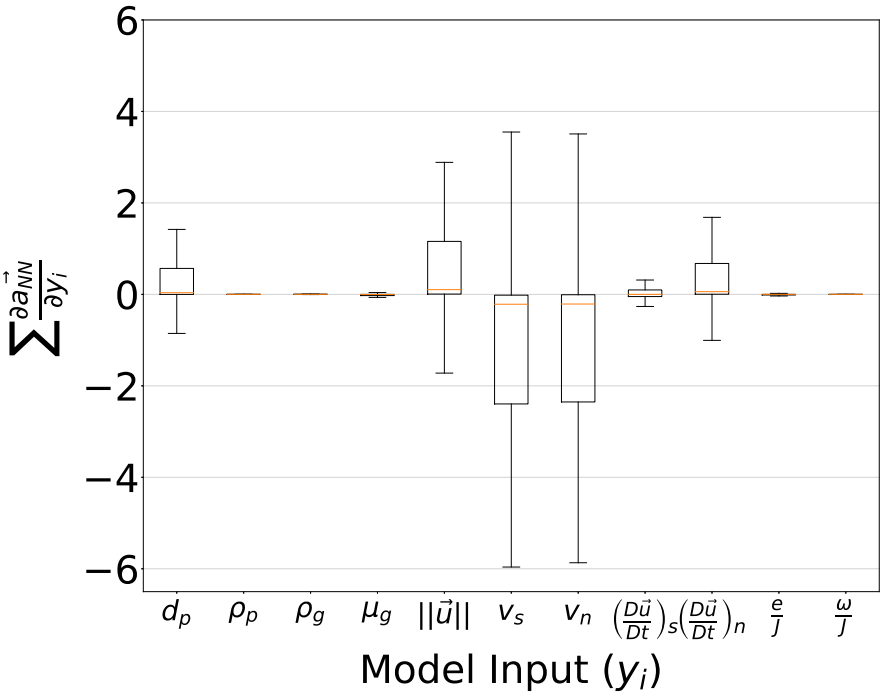
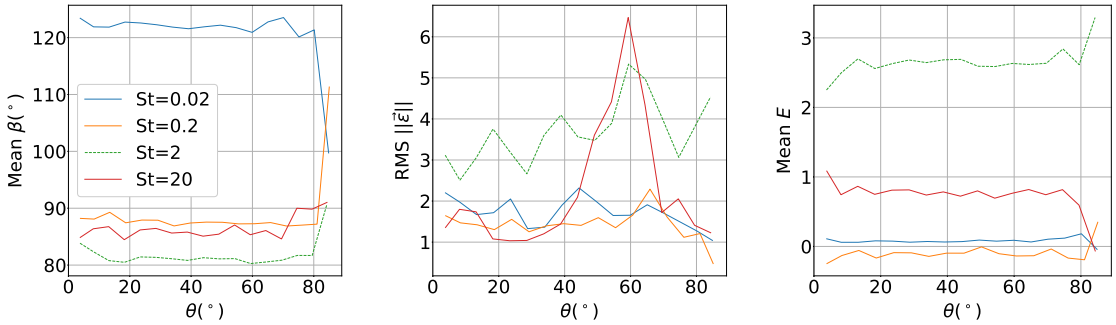
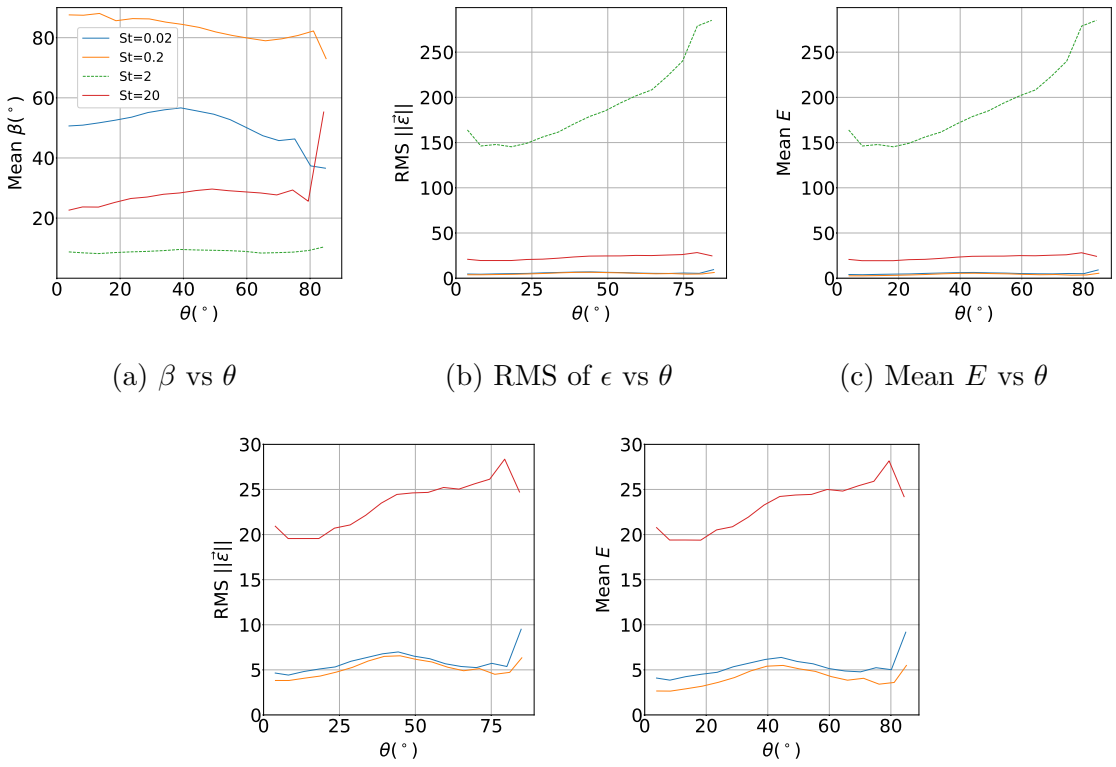


Figure 4.13: Model sensitivity to inputs. The model is clearly more sensitive to the particle velocity inputs.



(a) β vs θ (b) RMS of ϵ vs θ (c) Mean E vs θ

Figure 4.14: 3D HIT flow ANN errors from the Master ANN presented in Section 4.2. The blue, orange, dashed green, and red lines represent $St = 0.02$, 0.2, 2, and 20, respectively.



(a) β vs θ (b) RMS of ϵ vs θ (c) Mean E vs θ

(d) Zoomed-in view of ϵ vs θ (e) Zoomed-in view of E vs θ

Figure 4.15: 3D HIT flow ANN errors from the improved Master ANN presented in this section. The blue, orange, dashed green, and red lines represent $St = 0.02$, 0.2, 2, and 20, respectively.

Chapter 5

Louver Inertial Particle Separator

In this chapter, the best performing Master artificial neural network (ANN) presented in Section 4.3.1 is tested with the results of several unsteady 2D computational fluid dynamics (CFD) simulations of particle-laden flow around a louver particle separator geometry. That is, the ANN input data are extracted at the particle positions at various time intervals for ANN evaluation. Before conducting highly expensive simulations of the particle-laden flow through multiple louvers of various spacing as in [1], the fundamental laminar flow physics and particle flow interaction around a single louver was explored. An inlet Reynolds number (Re_{in}) within the laminar region for pipe flow was chosen because the ANN has only been trained with particle data from simple laminar flow cases. Additionally, particles of $St = 0.05$, 0.5, and 5 are considered.

5.1 Louver Particle Separator Geometry

The louver particle separator is characterized by Musgrove et al. with the area ratio (AR) and the number of louvers (N_L). Physically, the AR represents the ratio of total throat area between louvers to the domain height H and is defined as $AR = \frac{\delta N_L}{H}$ [1], where δ is the gap between louvers as shown in Figure 5.1. They found that an AR= 1.5 resulted in the lowest pressure loss of the ARs considered. Additionally, they show that $N_L = 8$ louvers is optimal for small particle separation efficiency [1]. Removing small particles from the flow is often the most difficult task in particle separator design because sufficiently small particles closely follow the fluid streamlines, as previously discussed in Section 1.2. The louver anchors (of length A_L) are stacked along a straight line angled to the horizontal shown by γ, which is determined by the space that the louver particle separator must fill [1]. The louvers themselves (of length L) are set at an angle ϕ, which is defined relative to γ as shown in Figure 5.1.

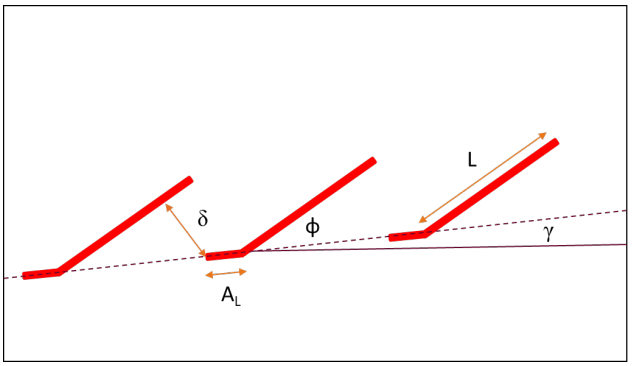


Figure 5.1: Multiple-louver particle separator geometry parameters.

5.1.1 CFD Solver and Computational Setup

For simulations of the louver geometry in this study, Eulerian-based gas phase transport equations and Lagrangian-based particle tracking equations are utilized for the unique challenges that particle laden flows present [50] [51]. This Euler-Lagrange method uses Direct Numerical Simulation (DNS) to solve the unsteady gas transport equations, which are given by the incompressible Navier-Stokes equations shown in Eq.(5.1) and Eq.(5.2)

$$\frac{\partial \rho_g}{\partial t} + \nabla \cdot \rho_g \mathbf{u} = 0 \tag{5.1}$$

$$\frac{\partial (\rho_g \mathbf{u})}{\partial t} + \nabla \cdot (\rho_g \mathbf{u} \otimes \mathbf{u}) = \nabla \cdot \boldsymbol{\tau} + \rho_g \boldsymbol{g} \tag{5.2}$$

where $\boldsymbol{\tau}$ is the fluid stress tensor [50]. Following Miranda and Palmore [52], the particles are modeled as discrete, perfectly spherical bodies that are governed by the Lagrangian equations similar to those previously defined shown by Eq.(5.3) and Eq.(5.4). Note that Eq.(5.4) is modified from its original form in Eq.(2.2) with the addition of the particle to wall collision force term $\mathbf{F}_p^C$. This term is modeled using the Miranda-Palmore method detailed in [52] which uses a soft-sphere coefficient of restitution (COR) approach to modeling particle collisions. In all studies considered here, both the tangential and normal components of the COR are set equal to 1. Thus, particle collisions with the louvers or wall boundaries are modeled as perfectly elastic collisions.

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{v} \tag{5.3}$$

$$m_p \frac{d\mathbf{v}}{dt} = \boldsymbol{f}_D + \mathbf{F}_p^C \tag{5.4}$$

The equations used by Miranda and Palmore are simplified such that the forces due to particle to particle collisions are not included because these multiphase simulations fall under the

regime of one-way coupling, and gravitational forces are neglected as well.

The inlet is specified as a velocity inlet and the top and bottom of the domain are no-slip walls as shown in Figure 5.2. The inlet velocity u_{in} is specified to fix Re_{in}, as defined in Eq.(5.5), where H is the height of the computational domain. The wall boundary conditions are specified since particle separators are designed for either inlet or secondary flow particle separation for gas turbine engine application [1]. As this setup mimics a pipe-flow case, the Re_{in} was set to 1000 to ensure that the flow was laminar [53]. This was required since the training data sets were taken exclusively from laminar flows. To ensure a reliable fluid phase solution, the simulation time step Δt was defined in order to limit the Courant-Friedrichs-Lewy (CFL) number to 0.8, as defined in Eq.(5.6) where Δx is the grid size. Moreover, the particle CFL number CFL_p defined in Eq.(5.7) was limited to 0.08, one order of magnitude less than the CFL, to ensure reliable Lagrangian particle tracking.

$$Re_{in} = \frac{\rho_g u_{in} \mathrm{H}}{\mu_g} \tag{5.5}$$

$$\mathrm{CFL} = \frac{\Delta t \ ||\mathbf{u}||}{\Delta x} \tag{5.6}$$

$$\mathrm{CFL}_p = \frac{\Delta t \ ||\mathbf{v}||}{d_p} \tag{5.7}$$

Additionally, the louver geometry is taken as a no-slip wall, achieved with an immersed boundary method, and the outlet boundary condition is a convective outflow that also allows the particles to escape the domain. Particles are initialized at the inlet with the same velocity as u_{in}. These simulations are designed ensuring that the volume fraction of particles to fluid, Φ, does not exceed 10^{-6}, such that the governing equations reflect a one-way coupling scheme [14]. Particle diameters corresponding to Stokes numbers ($St = \frac{\tau_p}{\tau_f}$) of $St = 5 * [10^{-2}, 10^{-1}, 10^{0}]$ are considered. In this case, the particle response time τ_p is defined as before in Eq.(1.3). However, the fluid time scale is defined as $\tau_f = \frac{\delta}{u_i}$ to reflect the definition of St given by [54]. The single louver, centered in the domain, is defined by the louver and anchor lengths L and $\mathrm{A_L}$, respectively, and the louver angle ϕ which is measured with respect to the louver anchor. Musgrove et al. consider louver angles ϕ from 30° to 60°, and use $\phi = 30°$ as a baseline comparison. However, note that the optimal geometry presented by [1] varies ϕ from 30-45° along the louver pattern, but this study uses $\phi = 30°$ since this was the angle used for baseline comparisons. Here, L is determined by the relation $\mathrm{L} = \frac{\delta}{\sin(\phi)}$, after the appropriate value of δ had been determined from AR= 1.5 and $\mathrm{N_L} = 8$ in accordance with [1]. While Musgrove et al. vary the louver anchor length to determine an ideal geometry, here the anchor length is taken to be $\mathrm{A_L} = 0.2\mathrm{L}$. In a multi-louver simulation, [1] define the angle of the louver anchor to the horizontal as the angle as a constant angle at which the

louvers stack, (i.e. they would stack along the dotted line). For simulating a single louver, this angle is arbitrary, and is chosen to be 10° to maintain similarity with [54].

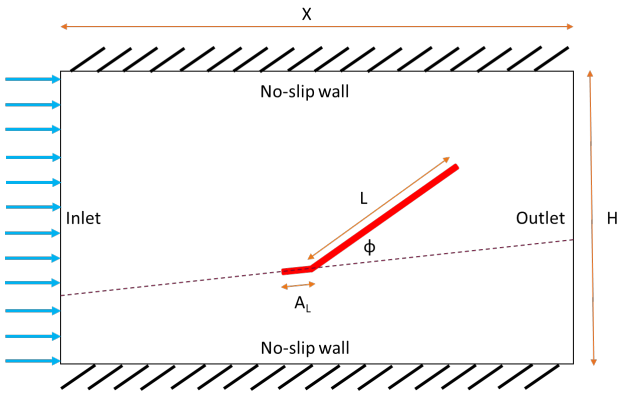
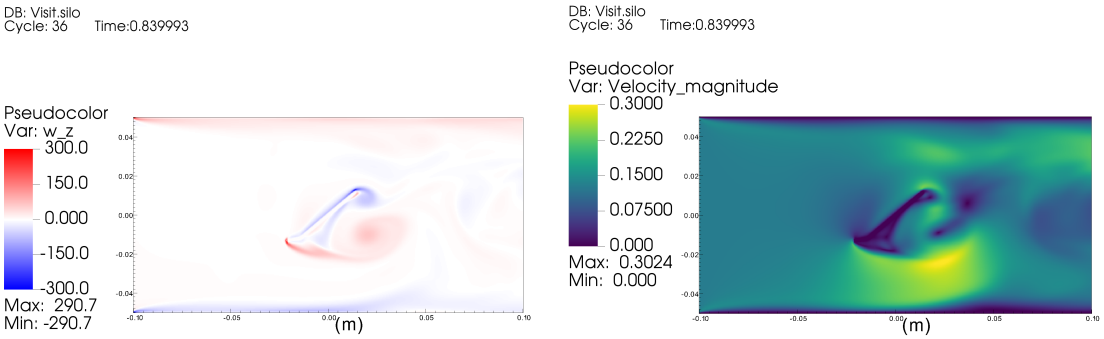


Figure 5.2: Louver particle separator computational domain. The top and bottom of the domain are no-slip walls, the flow into the domain is a velocity inlet where particles are also initialized with the same inlet velocity as the fluid, and the outlet allows particles to escape the domain.

5.2 Flow Feature Extraction for ANN Evaluation

In order to evaluate the Master ANN's performance for particles in real flow, it was necessary to compute the non-dimensionalized parameters $\mathbf{v}^*$, $\mathbf{u}^*$, $\frac{D\mathbf{u}^*}{Dt^*}$, $\frac{d\mathbf{v}^*}{dt^*}$, S^*, and Ω^* at each particle position for multiple points in time greater than τ_p. Specifically, the particle data was extracted every 2000 simulation time-steps ($2000\Delta t$) which corresponds to t = 4.8E−2, 1.6E−1, and 2.5E−1 s for St = 0.05, 0.5, and 5, respectively. These intervals were chosen since they are much greater than τ_p for each particle and capture the particles at various points throughout the domain. It was important to record data points at multiple simulation times in order to capture the particle interaction with developing flow and the unsteady wake behind the louver geometry. Recall the definitions of the principal invariants of the strain rate and rotation tensors, S and Ω respectively, given by Eq.(2.11) and Eq.(2.12). As before, time scales are non-dimensionalized with $\frac{1}{J}$ and length scales are non-dimensionalized by $L_{nd} = 1.0cm$. The square root of the velocity gradient tensor invariant, J, is defined by Eq.(2.10). As expected, the flow around the louver was inherently unsteady due to vortex shedding that occurs with blunt bodies immersed in flow. Figure 5.3a shows the z-component of vorticity for developing flow in the single louver configuration. The vortex region can be clearly seen developing behind the louver. The corresponding velocity magnitude is shown in Figure 5.3b.

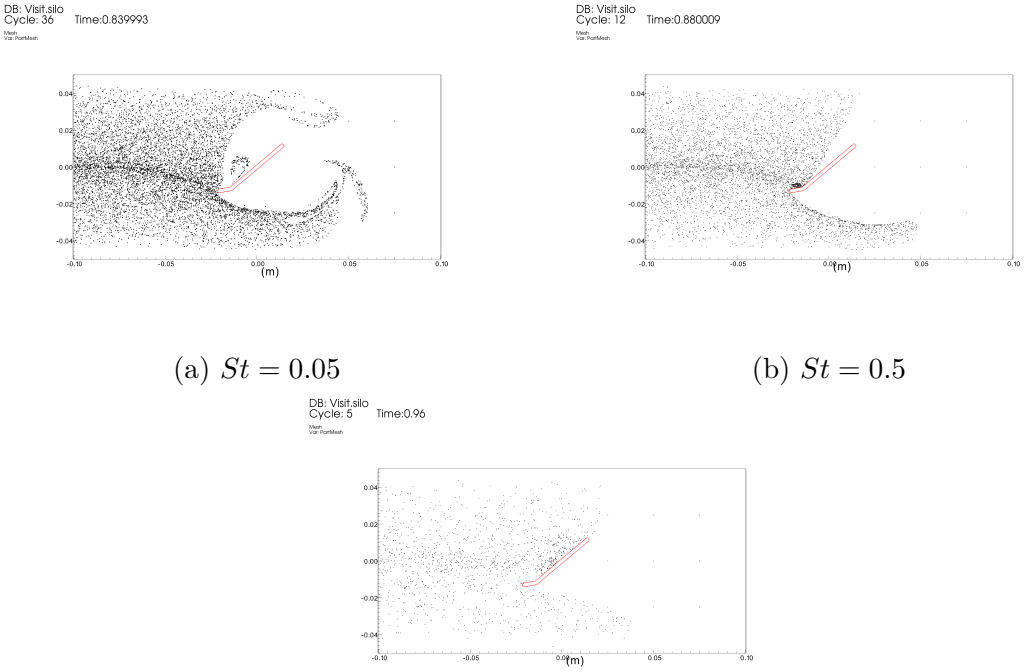
Snapshots of the St = 0.05, 0.5, and 5 particles are shown after approximately 0.9 seconds of flow time in Figure 5.4. The St definition agrees with theory, as the small St particles



(a) $(\nabla \times \mathbf{u})_z$ (s^{-1}) (b) $||\mathbf{u}||$ (m/s)

Figure 5.3: Developing flow around the louver geometry.

appear to trace the flow, even near the louver boundary, while the $St = 5$ particles appear largely unaffected by the flow streamlines.



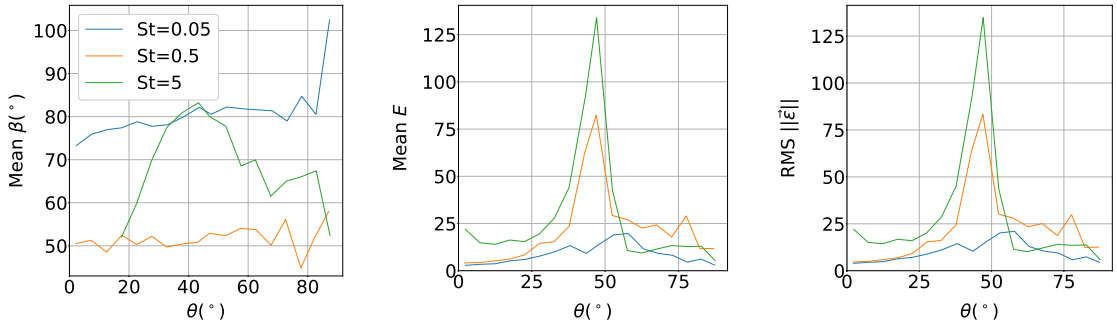
(a) $St = 0.05$ (b) $St = 0.5$

(c) $St = 5$

Figure 5.4: Particle interaction with flow around louver.

5.2.1 ANN Evaluation

As before, the directional error β, magnitude error E, and normalized error vector magnitude ϵ were evaluated as a function of the flow characteristic θ. It is important to note that the flow characteristic is calculated as $\arctan\left(\frac{\Omega}{S}\right)$ and is therefore limited to the range $(0°, 90°)$ since Ω and S are positive by definition. Figure 5.5 shows these three averaged error metrics as a function of θ and by St. From Figure 5.5a, it is clear that the ANN fails to predict the direction of the acceleration vector. The average value of β is no less than 45° for any value of θ. The averaged values of E and ϵ are remarkably similar, and suggest a peak around θ =45°. This agrees with the plots of E and ϵ previously shown in Figures 4.9b and 4.9c, where the error peaked for flows with $|\tan(\theta)| = 1$. One possible explanation of this is that the ANN trained on steady state laminar flows is unable to account for the time dependence of $\frac{D\mathbf{u}}{Dt}$. Another more concrete explanation can be related to the model sensitivity analysis presented in Section 4.3.1. Because the model is likely over trained on the dependence of $\mathbf{v}$, many of the observed directional and magnitude errors can probably be attributed to this. Additionally, the errors E and ϵ are shown to increase with St. This suggests that the non-dimensional data scaling is not similar enough for the ANN to make a meaningful prediction.

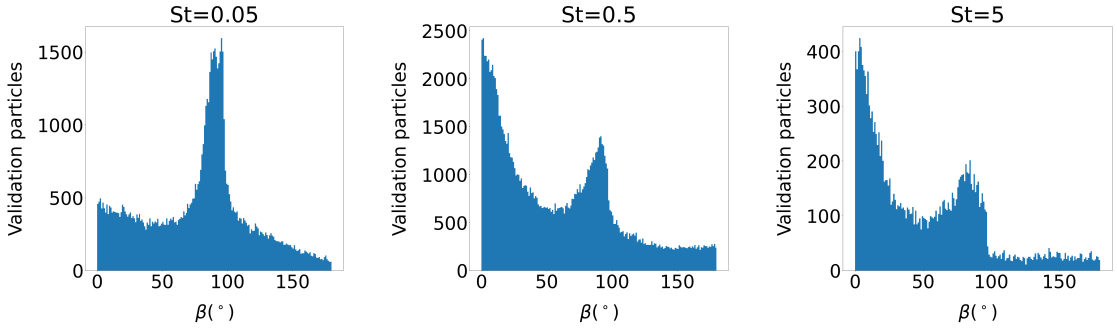


(a) β vs. θ (b) Mean E vs. θ (c) RMS of ϵ vs. θ

Figure 5.5: Master ANN error metrics when applied to particle data extracted from flow around the single louver. The blue, orange, and green lines correspond to St = 0.05, 0.5, and 5, respectively.

Additionally, to understand how the directional error is distributed, Figure 5.6 shows histograms of β by St. Figure 5.6a suggests that the majority of ANN-predicted accelerations, $\mathbf{a}_{NN}$, are orthogonal to the true particle acceleration $\mathbf{a}$ for St = 0.05. For St = 0.5 and 5, however, the majority of $\mathbf{a}_{NN}$ are closely aligned with the direction of $\mathbf{a}$ as shown in Figures 5.6b and 5.6c. Each of these distributions also suggests a smaller peak around β = 90°. This secondary peak could be observed because $\frac{S}{J}$ and $\frac{\Omega}{J}$ are positive while the ANN were trained with positive and negative values of e and ω. Accordingly, multiple combinations of e and

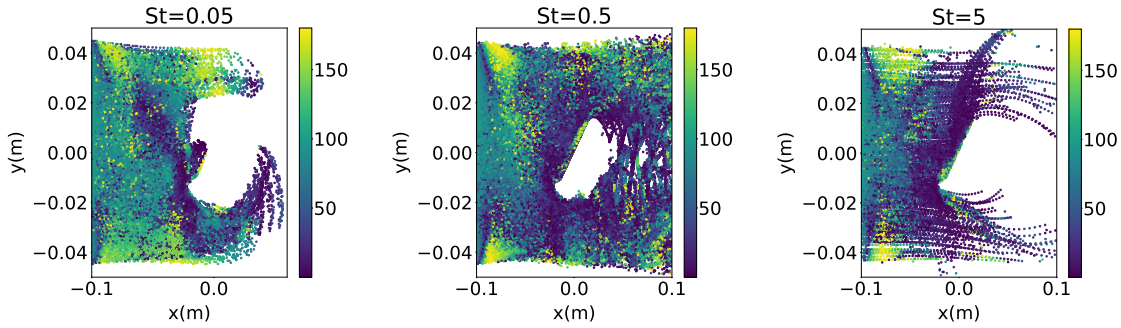
ω collapse onto a single tuple of S and Ω values in the louver flow since the unit circle is limited to the first quadrant as shown in Figure 2.1.



(a) $St = 0.05$ (b) $St = 0.5$ (c) $St = 5$

Figure 5.6: Distribution of β by St for particle data extracted from flow around the single louver. The bin widths are 1°.

Now that the average and distribution of directional error β has been characterized, it is of interest to understand the relationship between β and the particle position $\mathbf{x}_p$. Figure 5.7 shows how β depends on $\mathbf{x}_p$ by St. The general trends indicate that the errors are higher near the inlet and near the top and bottom wall boundaries. As the particles approach the louver geometry and pass over or under it, the errors appear to decrease. It is not surprising that the ANN cannot predict the acceleration direction as well when near inlet, outlet, and wall boundaries because the canonical flows with which the ANN was trained are formulated as continuous flow fields free of any wall boundaries. It should also be noted that large errors are observed for those particles colliding with the louver or top/bottom walls. This is also not surprising because the ANN is not trained in any way to handle collisions. It is worth noting some implications of this formulation. For larger particles, the post-impact trajectory is driven primarily by the collision model. Since large particles behave ballistically, the velocity of the particle immediately after impact mostly predicts its trajectory. Hence ANN predictions show high accuracy in the post-impact region for the larger particles in Figure 5.7c, even though the ANN was not explicitly trained with collision data. Small particles respond more quickly to the flow field, and as such their post-impact trajectory will depend on both the collision model and the flow field. Hence lower accuracy was demonstrated for the trajectory prediction of these particles post-impact. Future work could explicitly incorporate the effect of collisions into the ANN formulation.



(a) St=0.05 (b) St=0.5 (c) St=5

Figure 5.7: Scatterplots of β at the particle position $\mathbf{x}_p$ by St for particle data extracted from flow around the single louver.

Chapter 6

Conclusion

The viability of deep, feedforward neural nets to predict particle acceleration has been studied for several canonical flows. For each canonical flow, a four hidden-layer artificial neural network (ANN) with successively smaller hidden layers was built and trained using *PyTorch*. The directional error and the RMS of the normalized error vector were found to increase with Stokes number (St). Three-dimensional representations of the relative error ϵ as a function of particle position for the stagnation point flow suggest that there is some correlation between the observed error and the fluid streamlines. The error is slightly larger for particles located on the dividing streamline (y-axis), which could be a result of the particles deviating from this streamline. Despite not being trained with transient data sets, the ANN can predict the trajectory of $St = 500$ particles with no initial velocity and $St = 5$ particles initialized with a random initial velocity with some accuracy.

For laminar flow combining shear and vortex flows, the decreasing hidden-layer size network architecture was not sufficient for meaningful model training with the more complex combined flow case. Therefore, a seven hidden-layer ANN with a rectangular block structure (same size hidden layers) was used. However, it is clear that the ANN does not fully model the function space. Of the particles considered, the combined flow ANN's performance was the best for the two smallest St considered in the combined laminar flows. For the larger St in the combined flow cases, the predicted particle acceleration was often in the exact opposite direction of the actual vector. This led to the formulation of a custom loss function that penalized the ANN based on the L_1 norm, or mean absolute value, of the normalized error vector $\boldsymbol{\epsilon} = \frac{\mathbf{a}-\mathbf{a}_{NN}}{||\mathbf{a}||}$. This loss function, along with manually limiting the learning rate lr allowed acceptable accuracy to be reached with the training and validation data. However, the model was not found to be generalizable as it could not serve as a suitable replacement for the acceleration governing equation. A model sensitivity analysis suggested that the ANN had been over-trained on the dependence of $\mathbf{v}$. The results from solving the governing equations with the ANN substituted for the true acceleration agree with this. Additionally, for the 3D HIT case, the combined flow ANN failed to predict the particle accelerations

in any meaningful manner. This is not surprising, as the combined flow ANN was trained with 2D laminar combined flow data and that network struggled to learn even the 2D space. It should be noted though, that the improved ANN that was trained with the custom loss function did show some improvement in the directional error β. However, the magnitude error increased for the improved ANN compared to the original when evaluated with the 3D HIT data. This suggests that some scalings in the model might be incorrect. Alternatively, it could be another symptom of the model's over-dependence on $\mathbf{v}$.

Finally, the particle-laden 2D CFD simulations of channel flow around the louver geometry only reaffirmed that the ANN is not generalizable in its current state. While the simulation results appeared to agree well with St theory, the ANN failed to accurately predict the acceleration using extracted particle data. However, for $St = 0.5$ and 5, the distribution of directional errors was most populated around 0°. While this is encouraging, the distributions exhibit a bi-modal distribution with a secondary peak around 90°, entirely orthogonal to the actual vector. The highest errors in the directional error prediction for any case appeared to be near the inlet or wall boundaries. This makes sense as the canonical flows with which the ANN was trained have no features similar to these.

While the ANNs presented are shown to fit the highly redundant and simplistic data set generated in this work, they are not generalizable. This could be implicitly because the training data fails to span the entire function space. The model sensitivity analysis supports the idea that the model was over-trained on particle features at $t^* = 1.0$, which is the time at which all of the training and validation data was taken. Alternatively, it could suggest that the method of flow decomposition presented here is fundamentally limited. In either case, further studies need to be conducted to elucidate why the model does not generalize.

6.1 Future Recommendations

The results of this study suggest that the the deep feedforward neural nets presented here were not sufficient to capture the entire function space of particle acceleration, namely because of training data limitations. It will be the topic of future work to use more advanced strategies for developing neural nets. For example, the current work focused only on feedforward ANNs; however, convolutional neural networks are more popular in practice for deep learning. The first alternative solution is to consider more complex variations of ANN models such as generative adversarial networks (GANs), which have demonstrated capability of forecasting high-fidelity simulations [55]. Separately, a physics-informed neural network (PINN) approach [20] with a loss function informed by the relevant governing equations may offer some advantage. Lastly, improvements can be made by training models with three-dimensional unsteady data. These three approaches could first be implemented independently to demonstrate their unique advantages and downfalls, and they can be combined such as in [56].

This work avoided explicitly providing the particle Reynolds number, Re_p, as a model input in hopes that the ANNs considered could implicitly determine this important non-dimensional parameter and its effect on drag forces. In general, it seems as though the ANNs were effectively able to determine Re_p because the accelerations of the various size particles in the validation data set were learned with reasonable accuracy. Additionally, the trained model was shown to be highly sensitive to the fluid velocity $\mathbf{u}$ and particle velocity $\mathbf{v}$ compared to the other inputs. This is enough information for the ANN to establish the relative velocity $\mathbf{u} - \mathbf{v}$. In fact, the model sensitivities to $\mathbf{u}$ and $\mathbf{v}$ seemed to have opposite signs which is consistent with the nature of relative velocity. The ANN model was also shown to be somewhat sensitive to the particle diameter d_p. With constant physical properties, these learned dependencies are analogous to learning Re_p and St. We know that the drag, and therefore the acceleration, are highly dependent upon Re_p. It is also known that the particle interaction with the flow is dependent on St. Explicitly including Re_p and St as human-knowledge based model inputs could increase model prediction accuracy and accelerate training convergence similarly to how transforming vectors to the local streamline coordinate system did. For more sophisticated ANN models that could be applied in two- and four-way coupling regimes, the particle volume fraction Φ should be included in the model input vector. Future studies should consider explicitly incorporating all known domain dependencies in the machine learning model and comparing to the results shown here.

The model sensitivity results and LSODA solver validation suggested that the ANN model was over fit on the dependence of particle velocity, $\mathbf{v}$. Therefore, it follows that the generation of a new data set is in order. In such a data set, the range of particle velocities for each St should cause the ratio of relative velocity to gas velocity, $\frac{||\mathbf{u}-\mathbf{v}||}{||\mathbf{u}||}$, to range from 0 to 1. That is, data points should be taken from when the particle velocity is 0 until $\mathbf{v} \approx \mathbf{u}$. This method should prevent over training the model on $\mathbf{v}$. Furthermore, instead of choosing five discrete St, a uniform range should be considered such that a similar number of particles with St on the order of 10^{-2}, 10^{-1}, and so forth are considered. This should better train the ANNs for the dependence of $\mathbf{a}$ on τ_p. Additionally, because the particle data taken from steady, laminar, canonical flows did not generalize very well, the lid-driven cavity flow could be used as a segue to more complex flows. Specifically, the lid-driven cavity could be used to determine how to best perform flow-feature extraction rather than assuming the non-dimensionalizations and appropriate invariant flow parameters *a priori*.

6.1.1 Particle Separator Design

This research lays the groundwork and establishes best practices for applying ANNs to predicting particle dynamics. However, there is clear room for the expansion of model capabilities that more directly relate to the task of inertial particle separator design. For example, while the models developed in this work exist to predict the instantaneous particle acceleration, more complex machine learning (ML) models could implicitly use the predicted particle

acceleration in order to compute the particle trajectory $\mathbf{x}_p(t)$. In such a case, entire particle trajectories would need to be generated for training and validation data sets. Additionally, the model would need to be modified such that particle to wall collisions are considered and that the effect of St on post-impact trajectory is captured. Model inputs would need to include the CFD flow-field solution and particle initial conditions. The structure of the ML model could be as the feedforward ANNs presented here. Orthogonally, a classifier ANN could be appended that would attempt to classify the generated output as either a true particle trajectory or an ANN-predicted trajectory. The loss function of this ANN would describe the amount of ML-generated trajectories that the classifier was able to identify as ANN predictions. That is, the objective is to minimize the number of "fakes" (ML-generated trajectories) that the classifier can correctly identify. This class of ANNs, known as GANs, have demonstrated success in generating high-quality images of human faces as shown in [57], and it is thought that high resolution Lagrangian particle tracking can also be achieved with an analogous approach.

Additionally, higher-order models that use the various louver geometry parameters as inputs could be developed to determine the optimum design without explicitly testing each configuration. In this case, particle trajectories could be considered as the model output. Alternatively, the particle separator collection efficiency could be designated as the output. The challenge with this approach, in either case, would be generating a sufficient amount of sample data to train the ANN on the entire design space of the louver geometry. Before proceeding with this approach, some analysis on the computational expense to generate training data should be completed.

Bibliography

[1] Musgrove, G. O., Barringer, M. D., Thole, K. A., Grover, E., and Barker, J., 2009. "Computational design of a louver particle separator for gas turbine engines". *Proceedings of the ASME Turbo Expo,* ***3***(PART B), pp. 1313–1323.

[2] Nilamdeen, S., and Habashi, W. G., 2011. "Multiphase approach toward simulating ice crystal ingestion in jet engines". *Journal of Propulsion and Power,* ***27***(5), pp. 959–969.

[3] Flegel, A. B., 2017. "Ice crystal icing research at NASA". *9th AIAA Atmospheric and Space Environments Conference, 2017*(November).

[4] NATO, 1978. "Aircraft Icing". *AGARD Conference Proceedings*.

[5] NATO, 1995. "Recommended practices for the assessment of the effects of atmospheric water ingestion on the performance and operability of gas turbine engines". *AGARD Conference Proceedings*.

[6] NATO, 1994. "Erosion Corrosion and Foreign Object Damage Effects in Gas Turbines". *AGARD Conference Proceedings*.

[7] Pan, H., and Render, P. M., 1996. "Impact characteristics of hailstones simulating ingestion by turbofan aeroengines". *Journal of Propulsion and Power,* ***12***(3), pp. 457–462.

[8] Leroy, D., Fontaine, E., Schwarzenboeck, A., Strapp, J. W., Korolev, A., McFarquhar, G., Dupuy, R., Gourbeyre, C., Lilie, L., Protat, A., Delanoe, J., Dezitter, F., and Grandin, A., 2017. "Ice Crystal Sizes in High Ice Water Content Clouds. Part II: Statistics of Mass Diameter Percentiles in Tropical Convection Observed during the HAIC/HIWC Project". *Journal of Atmospheric and Oceanic Technology,* ***34***(1), 1, pp. 117–136.

[9] Houze, R. A., 2014. "Types of Clouds in Earth's Atmosphere". In *International Geophysics*, Vol. 104. Academic Press, 1, ch. 1, pp. 3–23.

[10] Rodríguez-Sanz, ., Arnaldo, R. M., Ayra, E. S., and Comendador, F. G., 2018. "Detecting High-Altitude Ice Crystal (HAIC) Icing Events from Total Air Temperature (TAT) Anomalies". In 31st Congress of the International Council of the Aeronautical Sciences.

[11] Kennedy, P. C., Rutledge, S. A., Petersen, W. A., and Bringi, V., 2001. "Polarimetric Radar Observations of Hail Formation". *The Journal of Applied Meteorology and Climatology,* ***40***(8), 8, pp. 1347–1366.

[12] NOAA. Hail Basics.

[13] Heymsfield, A. J., Jameson, A. R., and Frank, H. W., 1980. "Hail Growth Mechanisms in a Colorado Storm: Part II: Hail Formation Processes". *Journal of the Atmospheric Sciences,* ***37***(8), 8, pp. 1779–1807.

[14] Brandt, L., and Coletti, F., 2021. "Particle-Laden Turbulence: Progress and Perspectives". *Annual Review of Fluid Mechanics.*

[15] Esmaily-Moghadam, M., and Mani, A., 2016. "Analysis of the clustering of inertial particles in turbulent flows". *Physical Review Fluids,* ***1***(8), 12, p. 084202.

[16] Maxey, M. R., 1987. "The gravitational settling of aerosol particles in homogeneous turbulence and random flow fields". *Journal of Fluid Mechanics,* ***174***, pp. 441–465.

[17] Goodfellow, I., Bengio, Y., and Courville, A., 2016. *Deep Learning.* MIT Press.

[18] Hornik, K., Stinchcombe, M., and White, H., 1989. "Multilayer feedforward networks are universal approximators". *Neural Networks,* ***2***(5), 1, pp. 359–366.

[19] Minakowska, M., Richter, T., and Sager, S., 2021. "A Finite Element/Neural Network Framework for Modeling Suspensions of Non-spherical Particles: Concepts and Medical Applications". *Vietnam Journal of Mathematics,* ***49***(1), 3, pp. 207–235.

[20] Raissi, M., Perdikaris, P., and Karniadakis, G. E., 2019. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". *Journal of Computational Physics,* ***378***, 2, pp. 686–707.

[21] Seyed-Ahmadi, A., and Wachs, A., 2022. "Physics-inspired architecture for neural network modeling of forces and torques in particle-laden flows". *Computers & Fluids,* ***238***, 4, p. 105379.

[22] Seyed-Ahmadi, A., and Wachs, A., 2020. "Microstructure-informed probability-driven point-particle model for hydrodynamic forces and torques in particle-laden flows". *J. Fluid Mech,* ***900***, pp. 21–22.

[23] He, L., and Tafti, D. K., 2019. "A supervised machine learning approach for predicting variable drag forces on spherical particles in suspension". *Powder Technology,* ***345***, 3, pp. 379–389.

[24] Muralidhar, N., Bu, J., Cao, Z., He, L., Ramakrishnan, N., Tafti, D., and Karpatne, A., 2020. "PhyNet: Physics Guided Neural Networks for Particle Drag Force Prediction in Assembly". *SIAM*.

[25] Balachandar, S., Moore, . W. C., Akiki, . G., and Liu, . K., 2020. "Toward particle-resolved accuracy in Euler-Lagrange simulations of multiphase flow using machine learning and pairwise interaction extended point-particle (PIEP) approximation". *Theoretical and Computational Fluid Dynamics,* ***34***, pp. 401–428.

[26] Wu, Q., Zhao, Y., Shi, Y., and Chen, S., 2022. "Large-eddy simulation of particle-laden isotropic turbulence using machine-learned subgrid-scale model". *Phys. Fluids,* ***34***, p. 65129.

[27] Cutnell, J. D., and Johnson, K. W., 2012. *Physics*, 9 ed. Wiley.

[28] Clift, R., Grace, J. R., and Weber, M. E., 1978. *Bubbles, Drops, and Particles*. Academic Press, Inc., New York.

[29] Panton, R. L., 2013. *Incompressible Flow*, 4 ed. John Wiley & Sons, Inc., Hoboken, NJ, USA.

[30] Huang, G. B., 2003. "Learning capability and storage capacity of two-hidden-layer feedforward networks". *IEEE Transactions on Neural Networks,* ***14***(2), pp. 274–281.

[31] Heaton, J., 2015. *Artificial Intelligence for Humans: Neural Networks and Deep Learning*, 1 ed., Vol. 3. Heaton Research, Inc., Chesterfield, MO.

[32] Nielsen, M., 2015. *Neural Networks and Deep Learning*. Determination Press.

[33] Lu, L., Shin, Y., Su, Y., and Karniadakis, G. E., 2019. "Dying ReLU and Initialization: Theory and Numerical Examples". *Communications in Computational Physics,* ***28***(5), 3, pp. 1671–1706.

[34] Kingma, D. P., and Ba, J. L., 2015. "Adam: A Method for Stochastic Optimization". *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 12.

[35] Duchi, J., Hazan, E., and Singer, Y., 2011. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". *Journal of Machine Learning Research,* ***12***(61), pp. 2121–2159.

[36] Zeiler, M. D., 2012. "ADADELTA: An Adaptive Learning Rate Method". *arXiv*.

[37] Graves, A., 2014. "Generating Sequences With Recurrent Neural Networks". *arXiv*.

[38] Sutskever, I., Martens, J., Dahl, G., and Hinton, G., 2013. "On the importance of initialization and momentum in deep learning". *International Conference on Machine Learning,* ***28***.

[39] Kim, J., and Lee, C., 2020. "Prediction of turbulent heat transfer using convolutional neural networks". *Journal of Fluid Mechanics,* ***882***, 1.

[40] Glorot, X., and Bengio, Y., 2010. "Understanding the difficulty of training deep feedforward neural networks". *International Conference on Artificial Intelligence and Statistics in Proceedings of Machine Learning Research,* ***9***, pp. 249–256.

[41] Hindmarsh, A., and Petzold, L., 2005. LSODA, Ordinary Differential Equation Solver for Stiff or Non-Stiff System.

[42] Matson, R. J., and Huggins, A. W., 1980. "The Direct Measurement of the Sizes, Shapes and Kinematics of Falling Hailstones". *Journal of the Atmospheric Sciences,* ***37***(5), 5, pp. 1107–1125.

[43] Rasamoelina, A. D., Adjailia, F., and Sincak, P., 2020. "A Review of Activation Function for Artificial Neural Network". *SAMI 2020 - IEEE 18th World Symposium on Applied Machine Intelligence and Informatics, Proceedings*, 1, pp. 281–286.

[44] Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S., 2017. "Self-Normalizing Neural Networks". *Advances in Neural Information Processing Systems,* ***2017-December***, 6, pp. 972–981.

[45] Eaton, J. K., and Fessler, J. R., 1994. "Preferential concentration of particles by turbulence". *International Journal of Multiphase Flow,* ***20***(SUPPL. 1), pp. 169–209.

[46] Pope, S. B., 2000. *Turbulent Flows*. Cambridge University Press.

[47] Palmore, J. A., and Desjardins, O., 2018. "Technique for forcing high Reynolds number isotropic turbulence in physical space". *Physical Review Fluids,* ***3***(3), 3, p. 034605.

[48] Miranda, C., and Palmore, J., 2020. "High Stokes Number Droplets in Homogeneous Isotropic Turbulent Flow". *Eastern States Section of the Combustion Institute - Spring Technical Meeting*, 3.

[49] Hwang, W., and Eaton, J. K., 2006. "Homogeneous and isotropic turbulence modulation by small heavy (St=50) particles". *Journal of Fluid Mechanics,* ***564***, 10, pp. 361–393.

[50] Capecelatro, J., and Desjardins, O., 2013. "An Euler-Lagrange strategy for simulating particle-laden flows". *Journal of Computational Physics,* ***238***, pp. 1–31.

[51] Desjardins, O., Blanquart, G., Balarac, G., and Pitsch, H., 2008. "High order conservative finite difference scheme for variable density low Mach number turbulent flows". *Journal of Computational Physics,* ***227***(15), 7, pp. 7125–7159.

[52] Miranda, C. J., and Palmore, J., 2021. "Predicting Erosion from Airborne Particles on Surfaces using a Soft-Sphere Collision Model". *AIAA Aviation*, 8.

[53] Gerhart, P. M., Gerhart, A. L., and Hochstein, J. I., 2015. *Munson, Young and Okiishi's Fundamentals of Fluid Mechanics.*, 8 ed. Wiley.

[54] Borup, D. D., Elkins, C. J., and Eaton, J. K., 2020. "Experimental Analysis of a Particle Separator Design with Full-Field Three-Dimensional Measurements". *Journal of Turbomachinery,* ***142***(10), pp. 1–9.

[55] Pourbagian, M., and Ashrafizadeh, A., 2021. "Super-resolution of low-fidelity flow solutions via generative adversarial networks:". *SIMULATION: Transactions of the Society for Modeling and Simulation International*, 12.

[56] Li, M., and McComb, C., 2022. "Using Physics-Informed Generative Adversarial Networks to Perform Super-Resolution for Multiphase Fluid Simulations". *Journal of Computing and Information Science in Engineering,* ***22***(4), 8.

[57] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T., 2019. "Analyzing and Improving the Image Quality of StyleGAN". *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 12, pp. 8107–8116.

Appendix A

Additional Canonical Flow Figures

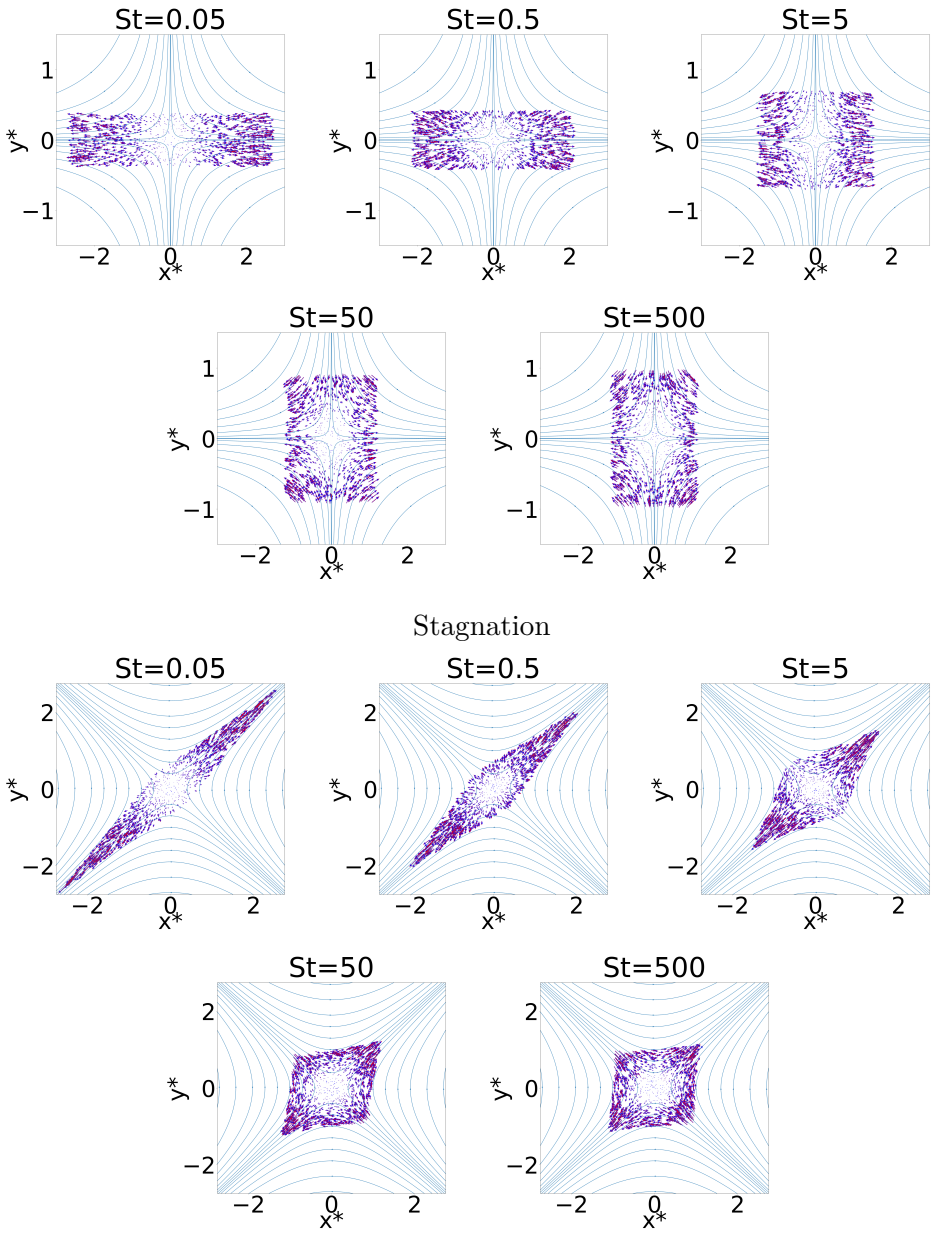


Figure A.1: Particle acceleration vector field - Simulation results vs ANN prediction.

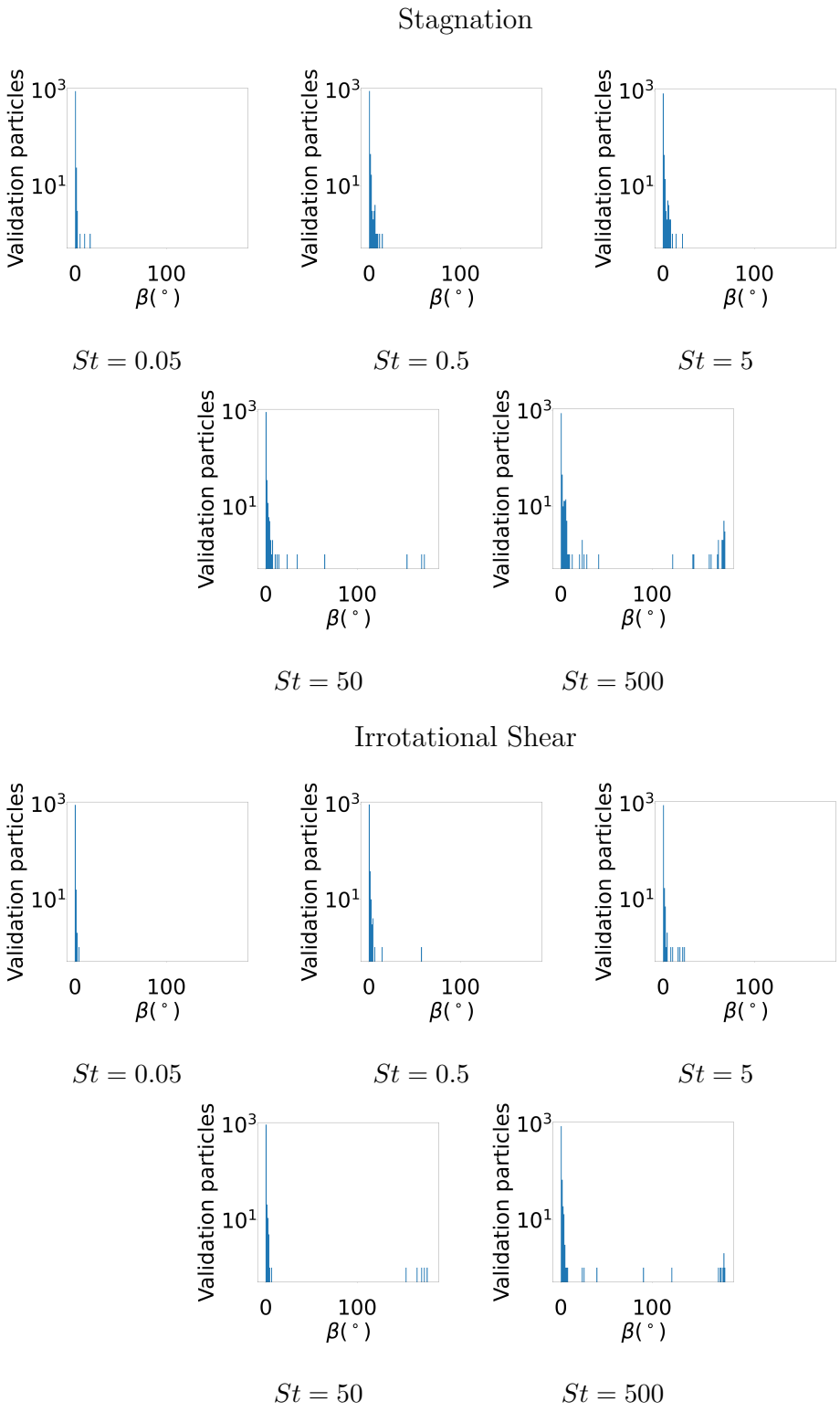
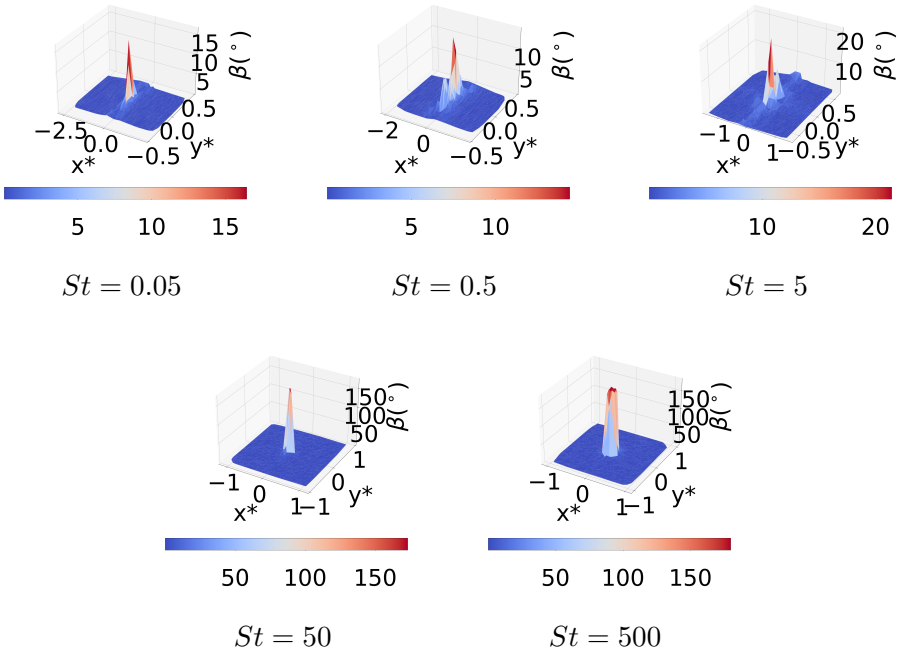


Figure A.2: Directional error (β) distribution. The bin widths are 1°.

Stagnation



Irrotational Shear

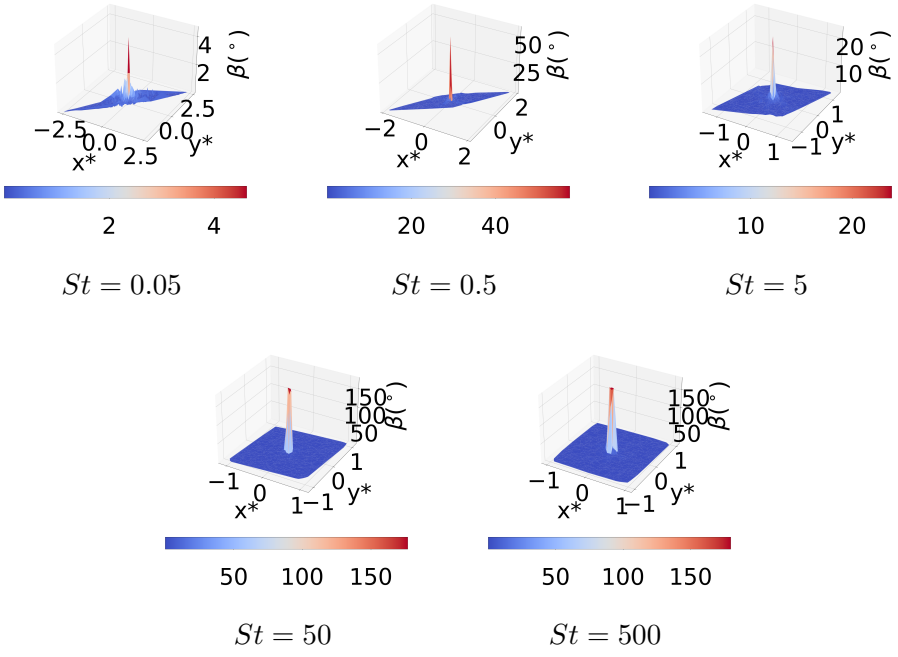


Figure A.3: Directional error (β) position dependence.

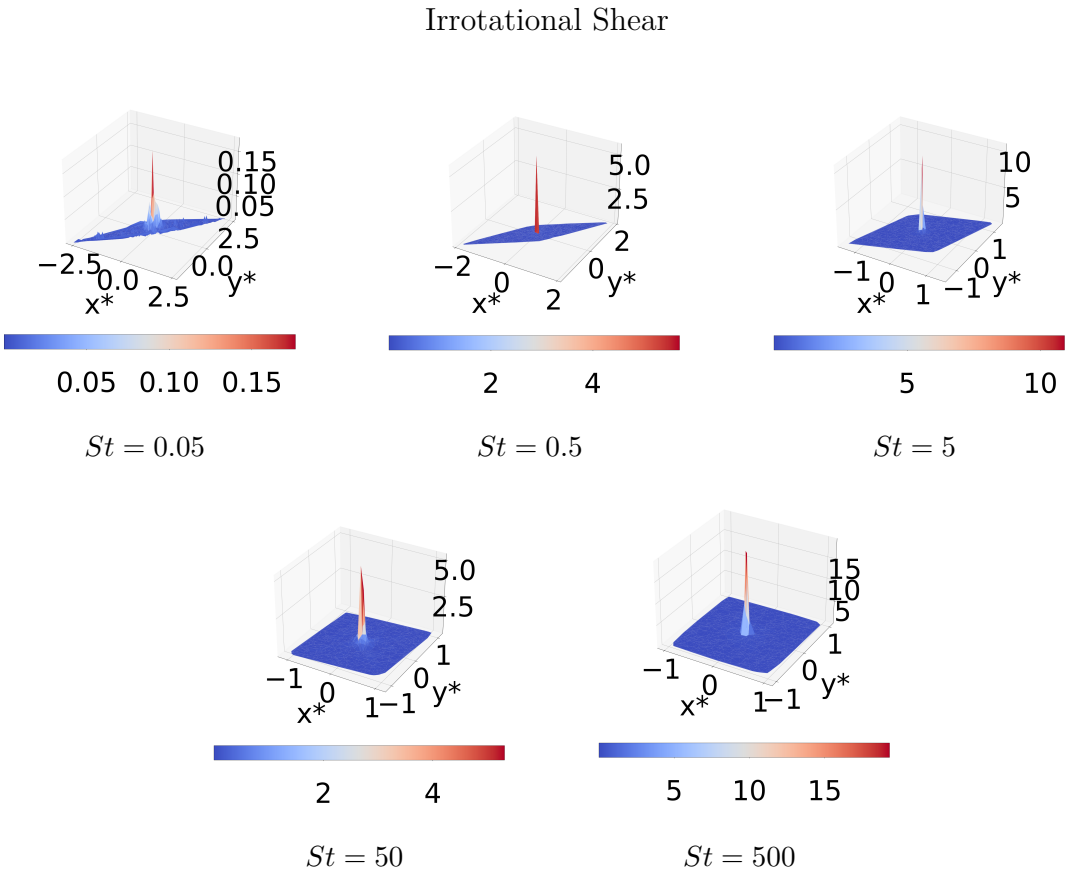


Figure A.4: ϵ position dependence.

Appendix B

Particle Trajectory Validation Initial Conditions

Table B.1: Particle initial conditions.

Particle No.	$\mathbf{x}_{p1}$	$\mathbf{x}_{p2}$	v_1	v_2
1	-0.4291	0.8439	0.5691	0.5221
2	-0.1935	-0.8574	-0.267	-0.5935
3	0.1412	-0.1105	-0.1747	-0.1573
4	0.5905	0.5948	0.0793	0.4736
5	0.2104	-0.676	-0.5699	-0.5983
6	-0.8633	0.5032	-0.3184	0.3097
7	-0.7844	-0.7602	-0.651	0.6103
8	0.9500	-0.1414	-0.1757	0.5658
9	0.2141	0.5535	0.0670	-0.9972
10	-0.9319	-0.1569	-0.4772	-0.7012

Appendix C

Additional Flow Tests with Master ANN for St=0.05

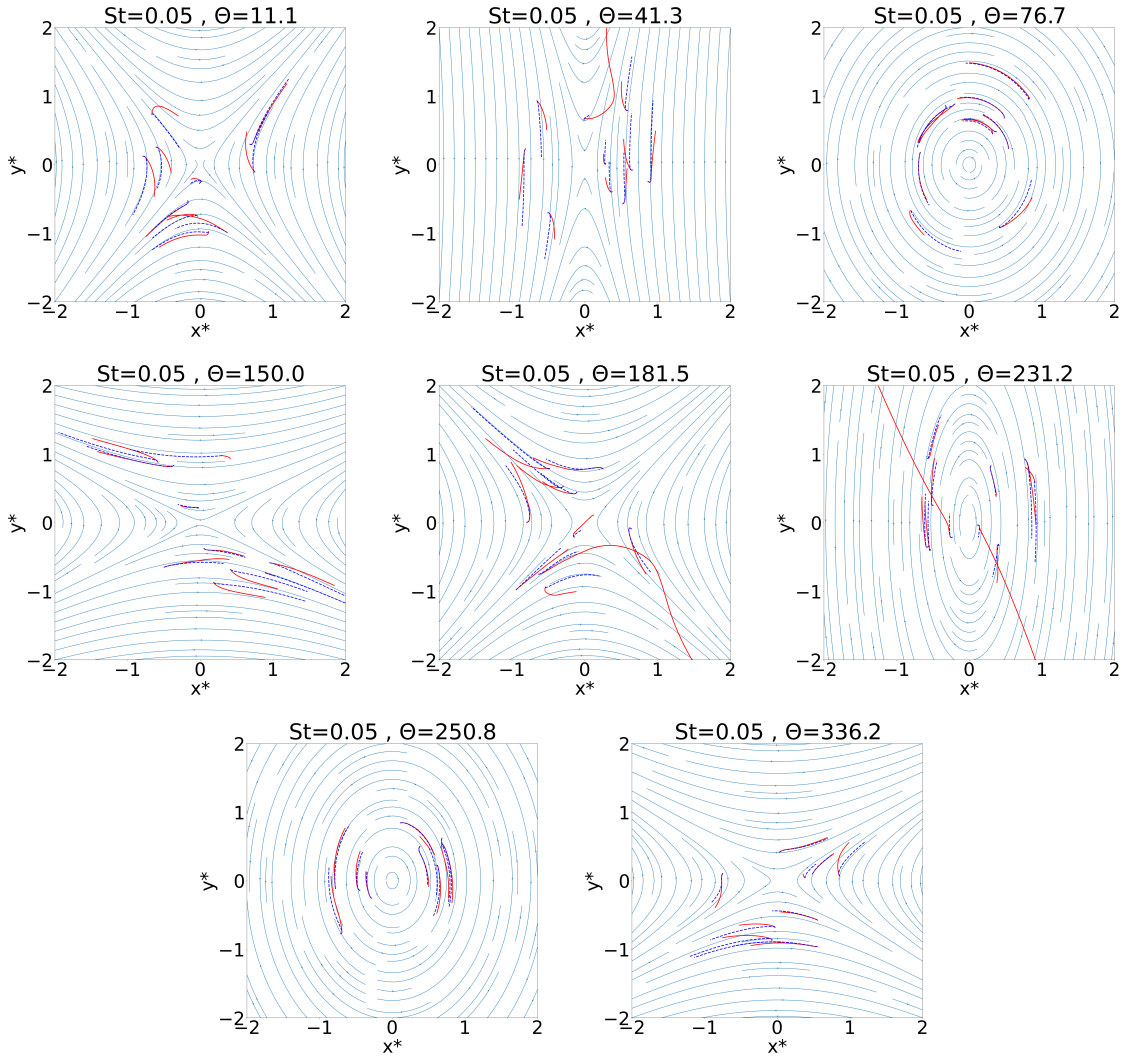


Figure C.1: Master ANN particle trajectory validation for $St = 0.05$, using various values of θ. The flow field is indicated by the blue streamlines. The particle trajectories of the governing equation solution and ANN approximation are shown by thedashed blue and solid red lines, respectively. Each figure shows particles that were initialized at a random position and with a random velocity with x and y components in range $[-1, 1]$. All plots show the particle trajectory for $t^* = [0, 1]$.