Deep learning-based prediction of Reynolds-averaged Navier-Stokes solutions for vertical-axis turbines

by

Chloë Dorge

A Thesis Submitted to the Faculty of Graduate Studies of
the University of Manitoba
in Partial Fulfilment of the Requirements of the Degree of

MASTER OF SCIENCE

Department of Mechanical Engineering

The University of Manitoba

Winnipeg

Abstract

The optimization of turbine array layouts for maximized power generation is an important and challenging problem in the fields of wind and hydrokinetic energy. Since multi-turbine simulations with resolved rotor geometries are beyond the capacity of modern computers, turbine arrays are generally modelled by replacing each rotor with a permeable, zero-thickness momentum sink, known as an actuator. Although actuator models are computationally inexpensive, they do not easily replicate the wake structures observed in experiments and foil-resolved simulations. With the aim of developing an alternative approach for modelling turbine interactions, the following study explores the potential of a deep learning-based turbine modelling technique. In the proposed method, a Convolutional Neural Network (CNN) is trained to predict the solutions of a foil-resolved, two-dimensional (2-D) Reynolds-Averaged Navier-Stokes (RANS) model, for a vertical-axis hydrokinetic turbine operating in free-stream velocities between 1 and 3 m/s. Based on the boundary conditions of free-stream velocity and rotor position, the flow gradients of x-velocity, y-velocity, pressure, and turbulent viscosity are predicted, in addition to the angular velocity of the rotor. Training and testing data are generated from the solutions of five RANS simulations, with free-stream velocities of 1, 1.5, 2, 2.5 and 3 m/s. Three trained CNN models are produced to evaluate the effects of (1) the dimensions of the training data, and (2) the number of simulations that are used as training cases. Smaller data sizes were found to improve prediction accuracy overall, while diminishing the computational cost associated with the generation of data from RANS solutions. Dramatic improvements in prediction accuracy were achieved by increasing the number of training cases from two to three. For the best achieved CNN model, the variables of x-velocity, y-velocity, pressure, turbulent viscosity, and angular velocity were predicted with mean relative errors of 5.97%, 8.98%, 12.68%, 7.37%, and 0.88%, respectively.

Acknowledgements

This research was supported by the NSERC Discovery and NRCan Clean Growth grants.

I would like to extend my sincere thanks to my advisor Dr. Eric Bibeau for his support, and for the freedom he granted me in the development of my research topic and methodology. I would also like to thank my committee members Dr. Nan Wu and Dr. Shawn Clark for their time and feedback. Many thanks to Dr. Ali Kerrache for his invaluable assistance in using the Grex high performance computer. Thanks also to Zeev Kapitanker and Derek Neufeld for their technical support, and to Michael Bear from New Energy Corporation for providing details on the EnviroGen 005 Series vertical-axis turbine.

I would also like to express my deepest appreciation to Rahmat Ali for his encouragement and guidance. His knowledge in the field of deep learning was indispensable.

Lastly, special thanks to my family, without whom this endeavour would not have been possible.

Contents

List of Tables

List of Figures

Abbreviations

2-D	Two-Dimensional
3-D	Three-Dimensional
6DOF	Six Degrees of Freedom
ANN	Artificial Neural Network
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
LES	Large Eddy Simulation
MPPT	Maximum Power Point Tracking
RANS	Reynolds-Averaged Navier-Stokes
UDF	User-Defined Function

1. Introduction

In May 2021, the Mauna Loa Atmospheric Baseline Observatory of the National Oceanic and Atmospheric Administration recorded an average atmospheric CO_2 mole fraction of 419 ppm in dry air, the highest monthly average concentration recorded at the weather station since it began monitoring CO_2 levels in 1958 (National Oceanic and Atmospehric Administration, 2021) (Tans & Thoning, 2018). Geological records indicate that modern atmospheric CO_2 levels have not occurred on Earth since the mid-Pliocene warm period, a relatively stable climatic period spanning 3.264 to 3.025 million years ago, during which the atmospheric CO_2 concentration ranged from approximately 350 to 450 ppm (Haywood, Dowsett et al., 2016). Paleoclimatic reconstructions of the mid-Pliocene warm period indicate that it featured a peak sea level that was 22±10 m above today's, as well as an intensified hydrological cycle, diminished global areas of sea ice and arid desert, and forest cover over what is now the Arctic tundra (Haywood, Hill et al., 2013). It is estimated that the average annual surface temperature during this period was 2.7 to 4ºC higher than it was at the cusp of the Industrial Revolution, at which point the average atmospheric CO_2 concentration was measured at 280 ppm (Haywood, Dowsett et al., 2016). Notwithstanding the uncertainty that is inherent to climate modelling and emission forecasting, current trends in Greenhouse Gas (GHG) emissions suggest that this century will witness a momentous climatic shift.

By 2050, a mean global atmospheric CO_2 concentration exceeding 500 ppm is not only plausible, but likely. In 2007, the expert meeting report prepared by the Intergovernmental Panel on Climate Change (IPCC) presented four plausible GHG concentration trajectories for the 21st century, referred to as Representative Concentration Pathways (RCPs) (Moss et al., 2008). In order of fossil fuel intensity, these scenarios include RCP2.6, RCP4.5, RCP6, and RCP8.5

(Schwalm et al., 2020). These RCPs were not developed to predict society's response to climate change, but rather to serve as a basis for climate models that would predict the outcomes and risks associated with different climate-related policies (Moss et al., 2008). The worst-case emission pathway, RCP8.5, proposes a global radiative forcing of 8.5 W/m^2 in 2100, with an atmospheric CO_2-eq concentration accelerating throughout the century to reach a maximum concentration of about 1370 ppm (Moss et al., 2008). This scenario is characterized as "a relatively conservative business as usual case with low income, high population and high energy demand" (Schwalm et al., 2020). Although RCP8.5 has often been regarded as an unlikely and even alarmist scenario since its development in 2007, a recent study posits that RCP8.5 may be the most reasonable basis for policy development (Schwalm et al., 2020). Of the four RCPs, the concentration pathway of RCP8.5 tracks most accurately with the total cumulative CO_2-eq emission observed between 2005 and 2020, within 1% (Schwalm et al., 2020). Moreover, when evaluated against projections of total cumulative emission for a "business as usual" policy scenario developed by the International Energy Agency (IEA), RCP8.5 was again found to be a sensible basis for risk assessment and policy development (Schwalm et al., 2020). As historical emission data begins to demonstrate the utility of a "worst-case" emission trajectory that has often been considered misleading, the importance of reducing global GHG emissions is becoming increasingly apparent.

The reduction of GHG emissions is achieved through three primary levers: the reduction of energy demand, the improvement of energy efficiency, and the expansion of renewable energy capacity. Although global energy demand is still climbing and significant improvements in energy efficiency are elusive in most fields, global renewable energy production is accelerating. According to a 2021 press release from the IEA, renewable energy is forecasted to comprise nearly 95% of the increase in global power capacity through 2026, and the increase in renewable capacity

between 2021 and 2026 is anticipated to be 50% higher that it was from 2015 to 2020 (International Energy Agency, 2021).

1.1. Hydrokinetic Energy

Hydrokinetic turbines are an important and growing branch of renewable energy technology. In a 2021 assessment of over 2.94 million river reaches around the world, the theoretical global riverine hydrokinetic resource was estimated as 6.66 TW, excluding Greenland (Ridgill et al., 2021). The average global tidal power resource is estimated as 3 TW, with about one third of this resource existing in shallow waters (Dhanak et al., 2016). In Canada, the hydrokinetic energy resource is estimated to be in the vicinity of 300 to 700 GW (d'Auteuil et al., 2019). Although there is substantial debate over what fraction of these hydrokinetic resources is pragmatically accessible, small-scale hydrokinetic turbines are bound to play a critical role in the electrification of remote communities around the world.

Hydrokinetic energy resources are particularly valuable in remote communities, as many of these populations are completely reliant on diesel generators for heating and electricity. In Canada alone, there are approximately 300 remote communities, the majority of which are powered by diesel generators (Karimi & Kazerani, 2017). In 2020, the annual diesel consumption in Canada's remote communities reached 682 million liters, of which two thirds was used for heating, and one third for electricity generation (Lovekin et al., 2020). This quantity of diesel corresponds to approximately 1.79 Mt of annual CO_2 emissions. While these emissions only constitute approximately 0.33% of the 536 Mt of CO_2 generated across Canada in the same year (Statista, 2021), the environmental and economic burden of diesel-based heating and electricity generation in Canada should not be overlooked. Diesel-powered communities in Canada are

especially vulnerable to air, soil, and water contamination by fuel leaks, since diesel lines and tanks are frequently damaged with the annual freezing and thawing of expansive soils. In addition, these communities must contend with an elevated and volatile cost of electricity generation relative to Canada's grid-tied population. The cost of diesel-based electricity generation forces many remote communities to impose load restrictions, which are an additional economic hindrance (Karimi & Kazerani, 2017). In the Kasabonika Lake First Nation community in Northern Ontario, load restrictions are estimated to have produced economic losses exceeding $9 million between 2010 and 2017 (Karimi & Kazerani, 2017). For many remote communities in Canada and around the world, small-scale hydrokinetic turbines can be a source of clean, continuous power, as well as a path to increased autonomy and improved living conditions.

1.2. Optimization of Turbine Array Configurations

In the fields of wind and hydrokinetic energy, the spatial optimization of turbine arrays for maximized power production is a challenging numerical problem with significant economic implications. Computational Fluid Dynamics (CFD) models of turbine interactions are exceedingly complex and computationally expensive, as they must resolve the performance of each turbine and the flow field of the entire array. Although models with fully resolved boundary layers are the most accurate method for capturing both the performance and wake of a turbine, these types of simulations are prohibitively time consuming without access to a High-Performance Computer (HPC), even for a single-turbine scenario. For those with access to an HPC, dual-turbine Reynolds-Averaged Navier Stokes (RANS) simulations with resolved boundary layers are feasible, but may nevertheless take months each to complete. The computational cost of performing such an intensive simulation for numerous dual-turbine array layouts quickly renders the process of optimization unwieldy.

To approach turbine array optimization within a more practical time frame, foil-based turbines are typically modelled by replacing complex rotor geometries with permeable, zero-thickness momentum sinks, commonly referred to as actuators, which approximate the forces and wake induced by the turbine rotors without the resolution of boundary layers. Many actuator models have been developed for both horizontal and vertical-axis turbines, with varying degrees of complexity and precision. In general, turbine actuators can be classified as either non-rotating or rotating, i.e., exerting either a static or dynamic load on the fluid domain, respectively. Non-rotating actuators take the form of the swept area of the foils, i.e., a cylinder for straight-bladed vertical-axis turbines, and a disc for horizontal-axis turbines. These actuators offer the greatest reduction in computational cost and are the most straight-forward to deploy, but deliver relatively crude results. Rotating actuators, on the other hand, are comprised of lines or surfaces that track with the position of each foil, coinciding with the aerodynamic center or chord line, respectively. Given their transient nature, rotating actuator models are better able to account for the effects of foil tip vortices and swirl on the evolution of the wake and on power generation, while still dramatically reducing computational cost relative to foil-resolved simulations. Rotating surface actuators offer the most refined results, but are also the most complex to develop.

High-quality actuator models are typically governed by advanced empirical corrections and carefully calibrated foil data, which require extensive know-how to develop, validate, and finesse. These complex adjustments are used to prescribe important flow phenomena, such as the projection of calculated body forces from the actuator elements onto the fluid domain, and the injection of leading and trailing edge vortices (Bachant et al., 2016). To account for the unsteady effects of dynamic stall and added mass, static foil characteristics must be corrected at each time step, based on the angle of attack and relative velocity inferred from the flow field (Bachant et al.,

2016). For vertical-axis turbines, lift and drag coefficients must also be corrected to account for the phenomenon of flow curvature, wherein the circular path of the foils results in a non-uniform angle of attack along the chord line (Bachant et al., 2016). Customized turbine features such as shrouds and winglets can also exacerbate the complexity of actuator models, as they may necessitate additional corrective measures. With expertly executed corrections, actuator models can approximate important flow features of turbine interactions, while having a sufficiently small computational cost to be executed on a personal computer. However, the complexity of such advanced models is not a trivial obstacle, and even the most rigorous correction techniques do not easily replicate the vortical wake structures and wake recovery observed in experiments and in foil-resolved simulations.

With the goal of developing an alternative approach for performing turbine array optimization, the following study investigates the effectiveness of a deep learning-based turbine modelling method, in which computational cost is diminished not by circumventing the resolution of boundary layers, but rather by minimizing the number of foil-resolved simulations that must be performed. More specifically, the following research uses a Convolutional Neural Network (CNN) to approximate foil-resolved RANS simulations of a vertical-axis turbine operating in previously unseen free-stream velocities, using the RANS solutions for bounding free-stream velocities as training data. As a preliminary foray into deep learning-based turbine modelling, this study does not address turbine interactions or the optimization of turbine spacing, but instead focuses on the viability of using a CNN to predict both the performance and flow field of a single turbine. The methodology and insights developed through this research are meant to serve as a steppingstone to dual-turbine interaction studies, in which a CNN will be applied to predict the power generation and flow field for previously unseen rotor spacings, in lieu of, or perhaps in

addition to, previously unseen flow velocities. Although the optimization of turbine spacing would ultimately be based on the predicted power output of the array, the prediction of flow gradients could provide additional numerical and graphical information by which to evaluate and improve the performance of the CNN, both as an array optimization tool and as a flow modelling technique in general.

1.3. Research Objectives

With the aim of developing an array optimization method that is both robust and computationally practical, the following research investigates the effectiveness of a deep learning-based turbine modelling technique. In the proposed method, a CNN is trained to infer the solutions of a transient, 2-D RANS model, for a vertical-axis hydrokinetic turbine operating in free-stream velocities between 1 and 3 m/s. For the boundary conditions of free-stream velocity and rotor position, the angular velocity of the turbine is predicted, as well as the flow field variables of x-velocity, y-velocity, pressure, and turbulent viscosity, i.e., the four unknowns of the 2-D RANS equation, with Reynolds stresses defined by means of the Boussinesq approximation. Training and testing data are produced from the solutions of five RANS simulations, with free-stream velocities of 1, 1.5, 2, 2.5 and 3 m/s. Three trained CNN models are generated to assess the effects of (1) the dimensions of the training data, and (2) the number of simulations that are used as training cases.

The work outlined herein can be condensed into the following core steps:

i) A transient, 2-D turbine model with fully resolved boundary layers is prepared in ANSYS Fluent®. Simulations are executed on the Grex HPC at the University of Manitoba, for free-stream velocities of 1, 1.5, 2, 2.5, and 3 m/s. Flow field and rotor

motion data is exported from each simulation, at each time step of the 21st turbine revolution.

ii) Simulation data exports are processed to produce the boundary conditions and ground truths by which the CNN will be trained and tested. For each CNN model developed in this work, the data generated from each individual simulation is restricted to either training or testing, in order that the trained CNN model only be tested against the solutions of previously unseen free-stream velocities. In order to investigate the effects of data size on prediction accuracy, two training datasets are produced: one containing data for free-stream velocities of 1 and 3 m/s, and the other containing data for free-stream velocities of 1, 1.5 and 3 m/s.

iii) CNNs are developed using the PyTorch machine learning library. Two CNN architectures are developed for different sizes of input data, in order to investigate the impact of training data dimensions on prediction accuracy. In addition, the smaller of the two architectures is trained using the two training datasets produced in the previous step, in order to evaluate how prediction accuracy is influenced by the number of simulations that are used as training cases.

1.4. Thesis Organization

The contents of this thesis are as follows: Chapter 2 provides a literature review on common actuator modelling techniques and deep learning-based modelling of flow around foils. Chapter 3 outlines the methodology of this work, as applied in the execution of turbine simulations in ANSYS Fluent®, in the generation of training and testing data from simulation exports, and in the development of a CNN. In Chapter 4, the results of the CFD simulations and trained CNN models are presented and discussed. Conclusions and recommendations are provided in Chapter 5.

2. Literature Review

The following section provides an overview of the various actuator methods that are commonly deployed in turbine interaction models. In addition, recent applications of deep learning in the modelling of foil flows are discussed.

2.1. Actuator Methods

To avoid the extreme computational expense of foil-resolved turbine simulations, turbine rotors are often modelled as permeable, zero-thickness momentum sinks, called actuators. Combined with Navier-Stokes or Euler equations, actuators can approximate the loading, power output, and three-dimensional (3-D) wake field of turbine rotors, without the resolution of boundary layers. For the greatest reduction in computational cost, turbines can be modelled in steady-state simulations by representing rotors as non-rotating actuators, which exert an invariable load on the flow domain. Alternatively, to achieve a higher quality transient simulation with moderate computational cost, turbine rotors can be modelled as rotating actuators, which exert a dynamic load on the flow domain that tracks with the motion of the rotor. The following section outlines the governing principles and limitations of non-rotating and rotating actuators, for both horizontal and vertical-axis turbines.

2.1.1. Non-Rotating Actuators

Non-rotating actuators approximate the time-averaged torque and power generation of turbines by applying constant momentum sink terms across the swept area of the foils. The forces projected by actuator elements onto the fluid domain are calculated based on prescribed or inferred velocity values. For horizontal and vertical-axis turbines, non-rotating actuators take the shape of

a disc and a cylinder, respectively. In the following section, an overview of actuator discs and cylinders is provided.

2.1.1.1. Actuator Discs

In the most simplistic actuator disc model, the forces occurring over the turbine foils are uniformly distributed over the swept area of the rotor, and the flow through the rotor is assumed to be ideal, i.e., steady, inviscid, incompressible, and irrotational. This model, which is commonly referred to as the 'classical' actuator disc model, is the foundation of Betz's law, which states that the maximum theoretical coefficient of performance of a rotor in ideal flow is 16/27. Based on classical Rankine-Froude theory, the classical actuator disc model relates the thrust and kinetic power of the turbine to the loss of axial momentum between the flow upstream and downstream of the actuator (Mikkelsen, 2003). As depicted in Figure 1, the flow traversing the actuator is modelled as an isolated streamtube, with a uniform cross-sectional velocity profile at the inlet and outlet, and at the actuator disc. In accordance with the conservation of energy and mass, the pressure discontinuity across the actuator disc causes the streamtube to slow and expand. The velocity at the disc is taken to be the mean of the inlet and outlet velocities.

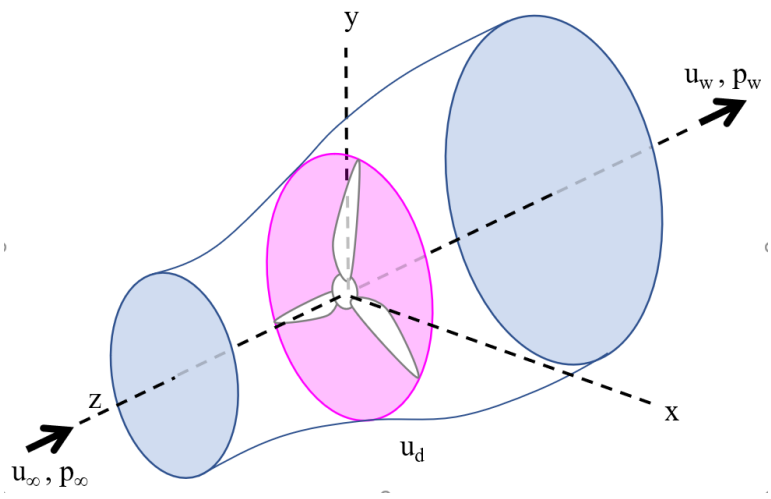


Figure 1: Classical actuator disc model

A more refined extension of the classical actuator disc is the Blade Element Momentum (BEM) actuator, which accounts for the force variation along the span of the foils by dividing the flow field into independent annular streamtubes, as shown in Figure 2. In this model, the change in axial and tangential momentum across each annular streamtube is balanced by the axial and tangential forces generated by the intersected foil elements (Mikkelsen, 2003).

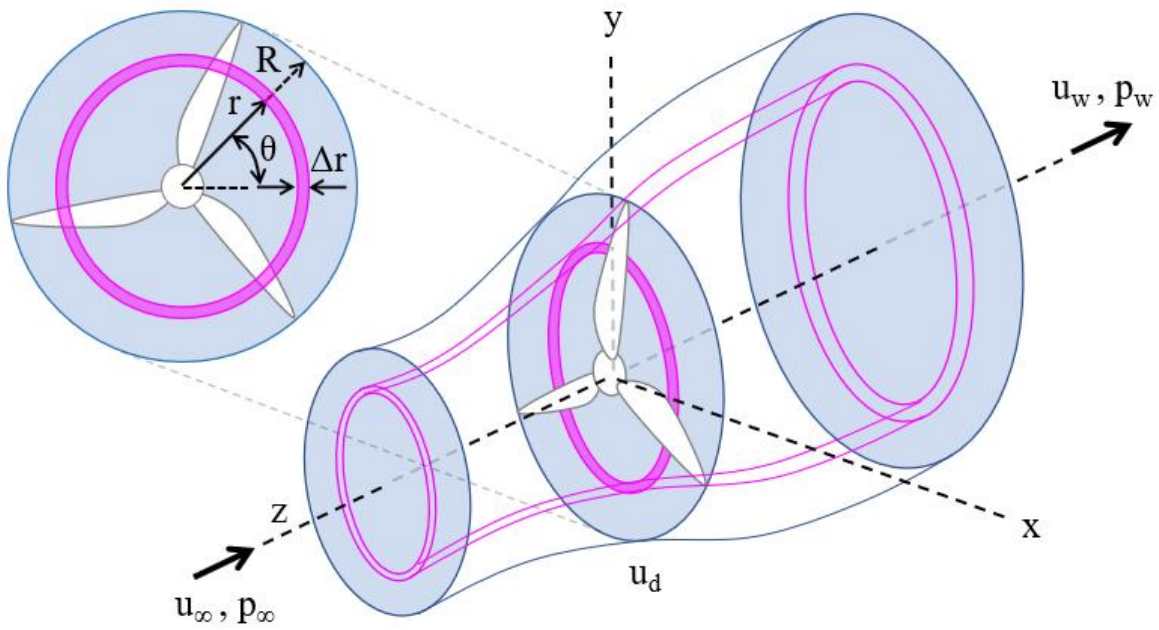


Figure 2: BEM actuator model

To develop this type of actuator model, the foils are discretized along their span into cross-sections that coincide with the annular streamtubes, as depicted in Figure 3. For each foil cross-section, BEM theory is used to calculate the forces in the axial and tangential directions, based on 2-D foil data. The calculated forces are then multiplied by the number of foils on the rotor, and are distributed over the corresponding actuator annulus.

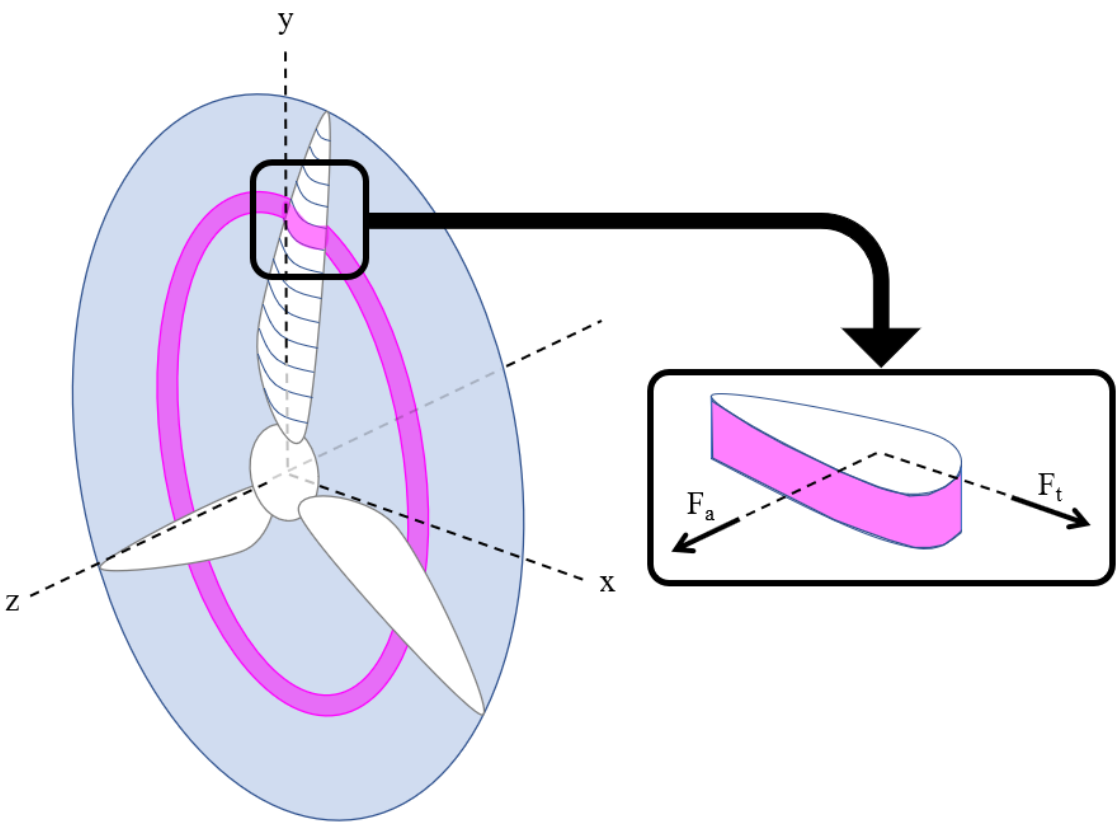


Figure 3: Foil discretization in the BEM actuator model

The BEM actuator model assumes steady, inviscid, and axisymmetric flow, and neglects the conservation of circulation (Mikkelsen, 2003). The cross-sectional velocity profile is assumed to be uniform at the inlet and outlet of the streamtube, and at the actuator disc. The flow velocity at the actuator is taken to be the mean of the free-stream and far-wake velocities. Due to the assumed independence of the annular streamtubes, the effects of pressure in the radial direction are neglected. Like the classical actuator disc, the BEM actuator model does not account for foil tip vortices, and therefore underpredicts the kinetic energy transferred to the wake by the rotor and overpredicts the kinetic energy contributing to power generation.

The BEM actuator model is a simple method for gaining insight into the behaviour of horizontal-axis turbines, and is frequently applied in industry (Mikkelsen, 2003). However, since

the BEM model is simply the relation of the rotor loading to the axial momentum loss, any application of the model requires that either the loading on the rotor or the axial velocity deficit be prescribed. To achieve an actuator disc model that can capture the turbine wake in three dimensions without prescribing the velocity at the rotor, the BEM actuator can be deployed in a discretized flow domain and coupled with Navier-Stokes or Euler equations, creating what is known as the general actuator disc model. In lieu of calculating the rotor forces based on a prescribed flow velocity, the general actuator method computes them based on velocities inferred over the actuator surface. Although the generalized actuator can capture some important features of the turbine wake, it shares some drawbacks with the BEM actuator model. Namely, both models are inviscid and axis-symmetric, and require high-quality foil data to reliably replicate rotor loading. In order to replicate the vorticity and wake decay observed in viscous models, the generalized actuator model must be enhanced with finely tuned empirical corrections.

2.1.1.2. Actuator Cylinders

A variety of actuator models have been developed for vertical-axis turbines. The double-multiple streamtube model, an early actuator model developed for Darrieus turbines, adapted the generalized actuator model to vertical-axis turbines by approximating the upstream and downstream portions of the rotor as two actuator plates in tandem, and by dividing the flow traversing the two actuator plates into multiple adjacent streamtubes (Paraschivoiu, 1988). While the double-multiple streamtube model is straightforward and computationally inexpensive, it does not accurately predict turbine loading and power generation (Ferreira et al., 2014). To eschew the need for streamtube theory, vertical-axis turbines are more frequently modelled using actuator cylinders. Similar to the generalized actuator disc, the actuators cylinder is produced by discretizing the swept area of the foils, and computing the normal and tangential forces occurring

over each element based on inferred local velocities and angles of attack (Cheng et al., 2016). A 2-D actuator cylinder model is depicted in Figure 4, in plan view.

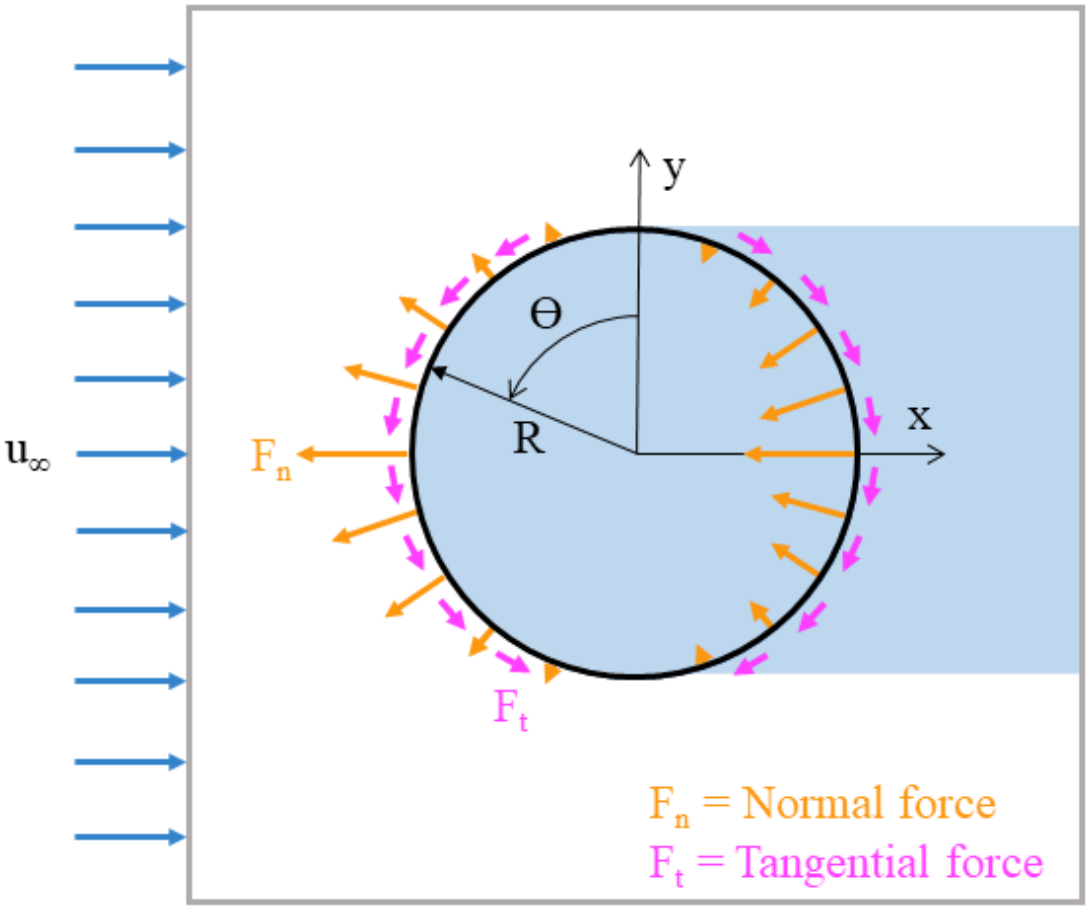


Figure 4: Two-dimensional actuator cylinder

Actuator cylinder models can be deployed to solve for the loading and power generation of multiple interacting turbines, and can be adapted to turbines with curved or swept foils (Ning, 2016). Actuator-in-actuator models consisting of two concentric actuator cylinders have also been developed, to account for the effects of the rotor shaft and struts (De Tavernier & Ferreira, 2019). Although actuator cylinder models can yield accurate predictions of turbine loading and power production, they are inviscid models, and therefore do not replicate wake decay and spread without empirical corrections (Ning, 2016).

2.1.2. Rotating Actuators

For both horizontal and vertical-axis turbines, transient RANS or Large Eddy Simulation (LES) models with moderate computational cost can be achieved by replacing foils with actuator lines or surfaces, as depicted in Figure 5.

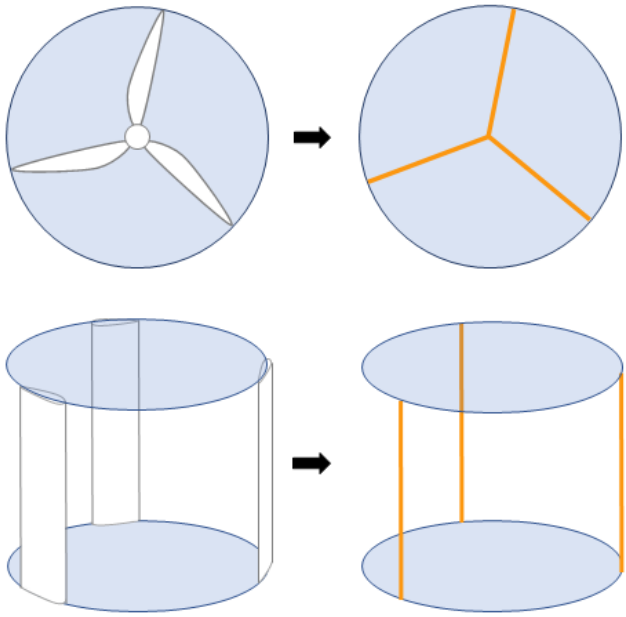


Figure 5: Actuator lines for horizontal and vertical-axis turbines

In rotating actuator models, the static lift, drag, and pitching moment coefficients of the actuator elements are interpolated from coefficient lookup tables at each time step, based on the velocity and angle of attack inferred from the flow domain. Using the computed coefficients, the forces incident on each actuator element are determined, and projected back onto the fluid domain as a momentum source term (Bachant et al., 2016). Advanced rotating actuators are currently the most viable method for modelling turbine arrays, as they can provide reasonably accurate loading and power predictions and are able to capture some important wake features. However, the

development of high-fidelity actuator models is far from easy, and remains a challenging problem in the fields of wind and hydrokinetic energy.

2.2. Deep Learning-Based Modelling of Foil Flow

Deep learning refers to a category of machine learning that employs Artificial Neural Networks (ANNs), which process data through numerous layers to extract non-linear patterns. Given their great capacity to extract general patterns from highly complicated data, ANNs are a powerful method for reducing the computational cost associated with the modelling of physical field phenomena, including fluid flow. In the field of CFD, ANNs can be trained with simulation data to infer solutions for previously unseen boundary conditions. In addition, they can be deployed in tandem with solvers to expedite solution convergence. In a recent study, a CNN was trained to predict velocity and pressure gradients around previously unseen airfoil shapes, using 2-D RANS solutions as training data (Thuerey et al., 2020). The velocity and pressure gradients were predicted with a relative error of less than 3%. In a similar work, a CNN was used as an intermediate step in RANS-based airfoil simulations, in order to expedite convergence (Obiols-Sales et al., 2020). The application of the CNN was found to reduce computational cost by 1.9 to 7.4 times for both steady laminar and turbulent flows, while satisfying the convergence conditions of the solver.

3. Methodology

The following chapter outlines the methodology of the work herein, as deployed in the preparation of numerical turbine models, in the generation and processing of data, and in the development of a CNN.

3.1. Numerical Simulations

To generate training and testing data for the CNN, 2-D transient simulations of a vertical-axis turbine are performed in ANSYS Fluent®, for free-stream velocities of 1, 1.5, 2, 2.5 and 3 m/s. Flow-driven rotation of the rotor is achieved by deploying the Dynamic Mesh method with the Six Degree of Freedom (6DOF) solver. A User-Defined Function (UDF) is used to specify the mass, moment of inertia, and rotational axis of the turbine. The turbine is modelled as free-wheeling, i.e., with no counter-torque applied to the rotor shaft, in order to achieve higher rotational velocities and steeper gradients in the flow domain. The geometry, mass, and moment of inertia of the turbine modelled in this study are based on those of a 5-kW, four-bladed, H-rotor hydrokinetic turbine, with a 60-inch rotor diameter, and NACA 0018 foils with a chord length of 4 inches and a pitching angle of 4°.

Simulations are performed using the Transition SST turbulence model, owing to its proven accuracy in the modelling of vertical-axis turbines (Marsh et al., 2017). The turbulence intensity of the flow domain is assumed to be 5%, and the turbulent length scale is specified as the diameter of the rotor. To ensure that the network is trained and tested based on stable simulation solutions, data is exported at each time step of the 21st turbine revolution (Rezaeiha et al., 2017). All simulations are performed using second-order spatial and temporal discretization, with a residual

criterion of 1×10^{-6}. To complete this study within a practical time frame, simulations are executed on the Grex HPC at the University of Manitoba.

As depicted in Figure 6, the flow domain is divided into three regions: a rectangular outer domain, a square middle domain, and a circular rotor domain.

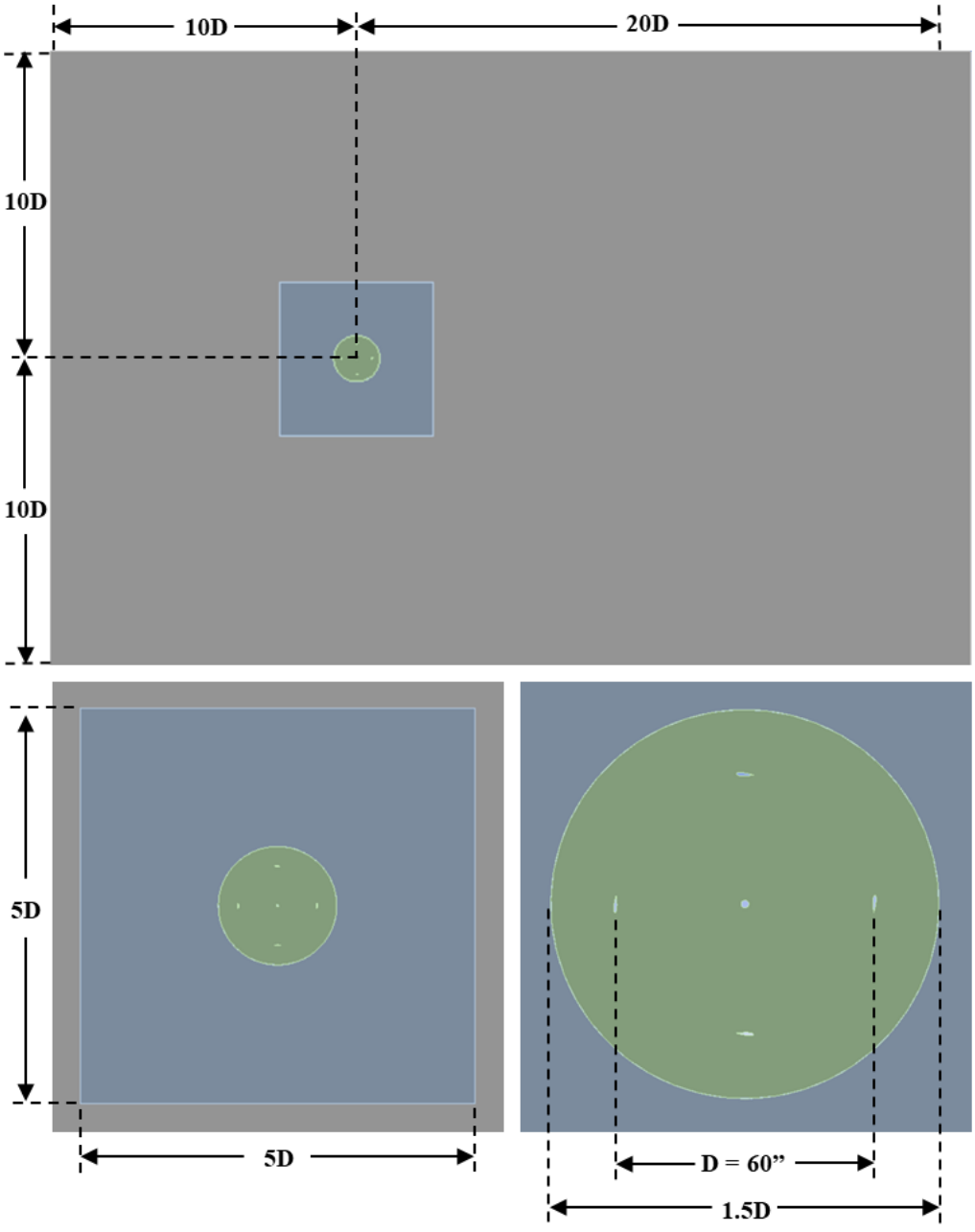


Figure 6: Flow domain geometry

The region over which solutions are inferred by the CNN encompasses the middle and rotor domains, as shown in Figure 7.

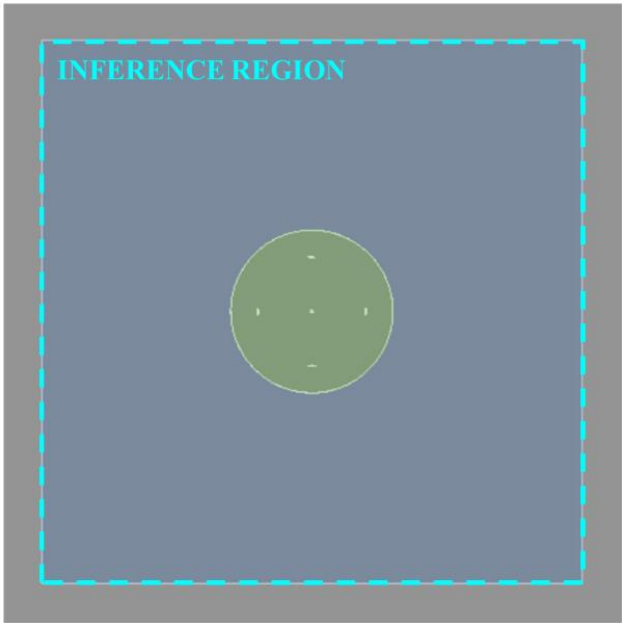


Figure 7: Inference region

The division of the flow domain into discrete zones serves three important purposes. Firstly, the separate zones allow for greater control over the mesh resolution around the turbine. By adjusting the resolution of each zone independently, grid independence can be achieved with fewer grid elements. Secondly, the zones are also used to limit solution data exports to the desired CNN inference region, thereby minimizing the memory required to store solution data, and reducing the computational expense of processing solution exports into training and testing data. As can be seen in Figure 7, the border of the inference region coincides with the outer boundary of the middle flow domain. Lastly, the division of the flow domain into discrete zones is required in the deployment of the 6DOF solver, which governs the flow-driven rotation of the rotor. In the configuration of the dynamic mesh, the dynamic rotor domain is defined as rigid, i.e., having an invariant mesh structure. To maintain mesh continuity between the dynamic rotor domain and the

rest of the flow field, the middle domain is defined as deforming, i.e., with a mesh structure that is reconstructed at each time step, in accordance with the motion of the rigid rotor domain. The outer domain is defined as stationary, i.e., with a fixed mesh. The configuration of the dynamic mesh for the different fluid zones is depicted in Figure 8.

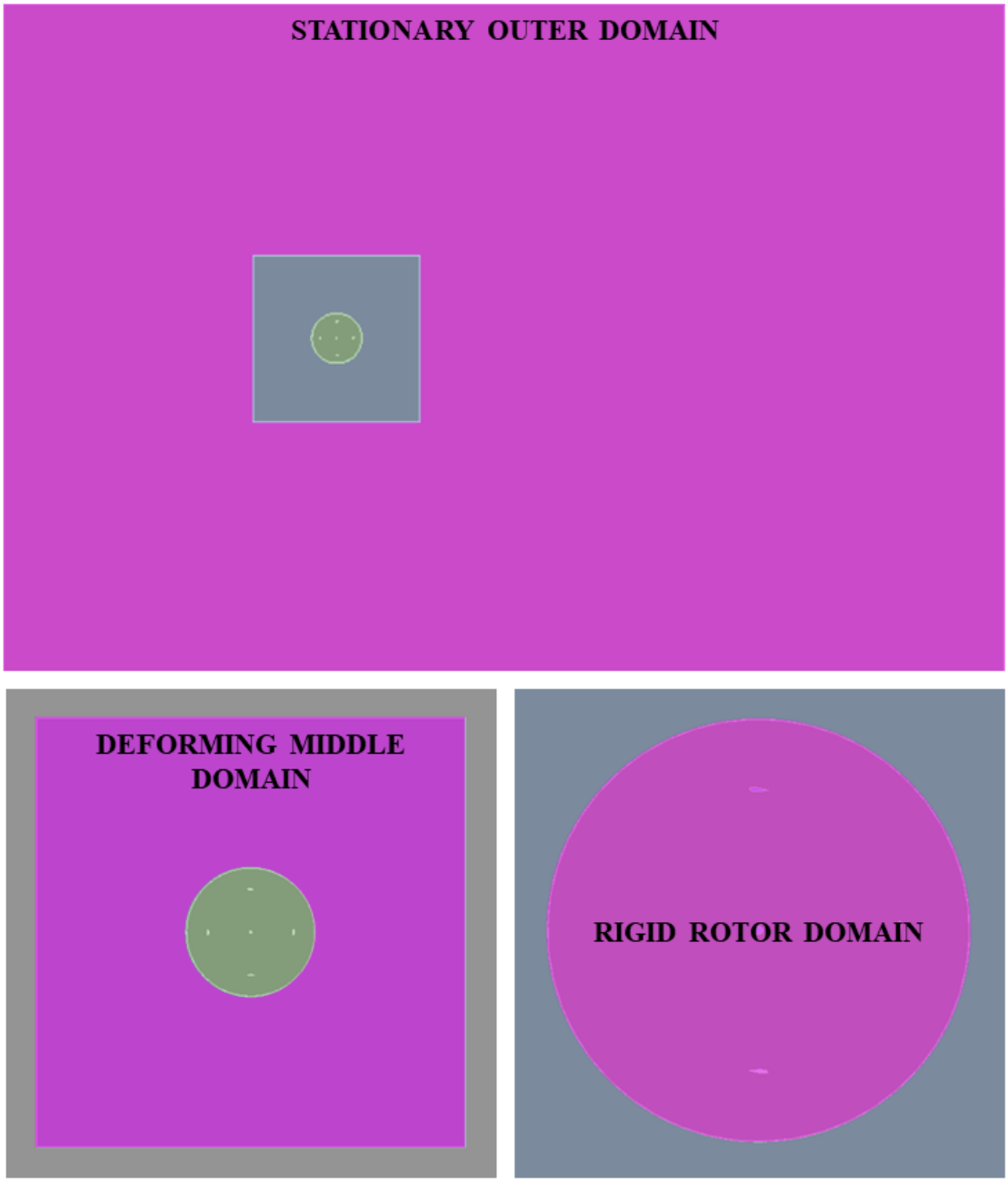


Figure 8: Flow domain regions as used in the configuration of the dynamic mesh

Triangular meshing is applied throughout the flow domain, as required by the dynamic mesh method. To resolve boundary layers, inflation layer meshing is applied around the foils and the rotor shaft, as shown in Figure 9.

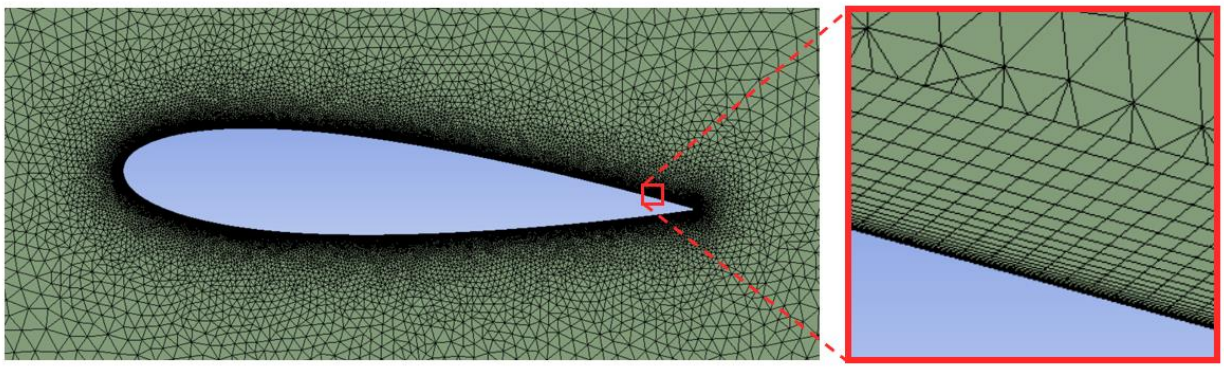


Figure 9: Resolution of the boundary layer region via inflation layer meshing

To resolve the viscous sublayer, a non-dimensional first-cell height of $y^+ < 1$ is applied along all surfaces. A first-cell width of $x^+ < 30$ is applied along surfaces in the tangential direction. To calculate the values of first-cell height and width, the maximum relative velocity of the fluid at the surface of the foils is assumed to be the sum of the maximum free-stream velocity (3 m/s) and the magnitude of the tangential velocity of the foil, assuming a maximum rotational velocity of 90 RPM. The applied surface mesh settings are summarized in Table 1.

Table 1: Surface Mesh Settings

Maximum first-cell width [m]	6.00E-05
Non-dimensional first-cell width, x^+	< 30
Maximum first-cell height [m]	2.00E-06
Non-dimensional first-cell height, y^+	< 1
Inflation growth rate	1.2
Number of inflation layers	20

A grid independence study is performed by varying the cell size in the outer, middle, and rotor domains. The properties of the meshes evaluated in the grid independence study are summarized in Table 2. For each mesh, a time step size of 0.0001 s is applied.

Table 2: Properties of the Meshes Evaluated in the Grid Independence Study

Mesh No.	**Time step size, T [s]**	**Non-dimensional first cell width, x^+**	**Non-dimensional first cell height, y^+**	**Grid size in rotor domain [m]**	**Grid size in middle domain [m]**	**Grid size in outer domain [m]**	**Number of nodes**
1	0.0001	< 30	< 1	0.005	0.025	0.1	863894
2	0.0001	< 30	< 1	0.01	0.05	0.1	657687
3	0.0001	< 30	< 1	0.01	0.05	0.2	554085
4	0.0001	< 30	< 1	0.02	0.1	0.2	500988

To ensure that grid independence is achieved without forfeiting computational efficiency, the four meshes are compared in terms of both their solutions and computational expense after 5000 time steps, at t = 0.5 s. The compared metrics include rotor angle, rotor drag and lift coefficients, and simulation running time. The results of the grid independence study are summarized in Table 3.

Table 3: Results of the Grid Independence Study at t = 0.5 s (5000th Time Step)

Mesh No.	**Rotor angle, θ [°]**	**Difference in θ relative to Mesh 1 [%]**	**Rotor drag coefficient, C_D**	**Difference in C_D relative to Mesh 1 [%]**	**Rotor lift coefficient, C_L**	**Difference in C_L relative to Mesh 1 [%]**	**Running time [hours]**	**Difference in running time relative to Mesh 1 [%]**
1	30.85	-	0.5503	-	-0.1131	-	56.41	-
2	31.29	1.41	0.5384	-2.15	-0.1141	0.89	34.95	-38.04
3	31.51	2.14	0.5345	-2.86	-0.1097	-3.00	30.43	-46.06
4	25.52	-17.30	0.4894	-11.06	-0.1292	14.29	25.07	-55.55

As can be seen in Table 3, meshes 2 and 3 both produce small discrepancies relative to the results for the finest mesh. Since grid and time step independence are interrelated, further investigation is required to ascertain whether grid independence is maintained with time step sizes larger than the one deployed throughout the first grid independence test (T = 0.0001s). Although meshes 2 and 3 were both found to be viable in the grid independence study, their sensitivity to time step size could differ. For this reason, meshes 2 and 3 are both evaluated in a time step size independence study. For four different time step sizes, the rotor angle produced by the two meshes

at t = 0.5 s is determined. The findings of the time step size independence study are summarized in Table 4.

Table 4: Results of the Time Step Size Independence Study at t = 0.5 s for Meshes 2 and 3

Time step size, T [s]	**Mesh 2**		**Mesh 3**	
	Rotor angle, θ, at t = 0.5 s [°]	**Difference in θ relative to result for T = 0.0001 s [%]**	**Rotor angle, θ, at t = 0.5 s [°]**	**Difference in θ relative to result for T = 0.0001 s [%]**
0.0001	31.2879	-	31.5141	-
0.0002	30.7781	-1.6294	32.1426	1.9943
0.0004	33.8644	8.2348	-	-
0.0005	34.6214	10.6543	-	-

For both meshes 2 and 3, the absolute difference in rotor angle between time steps of 0.0001 s and 0.0002 s was less than 2%. Time step sizes of 0.0004 s and 0.0005 s were found to produce excessive discrepancy in the rotor angle when deployed with mesh 2, and were therefore not tested with mesh 3. Based on the results obtained in the grid and time step size independence studies, simulations are conducted using mesh 3, with a time step size of 0.0002 s. Mesh 3 is depicted in Figure 10.

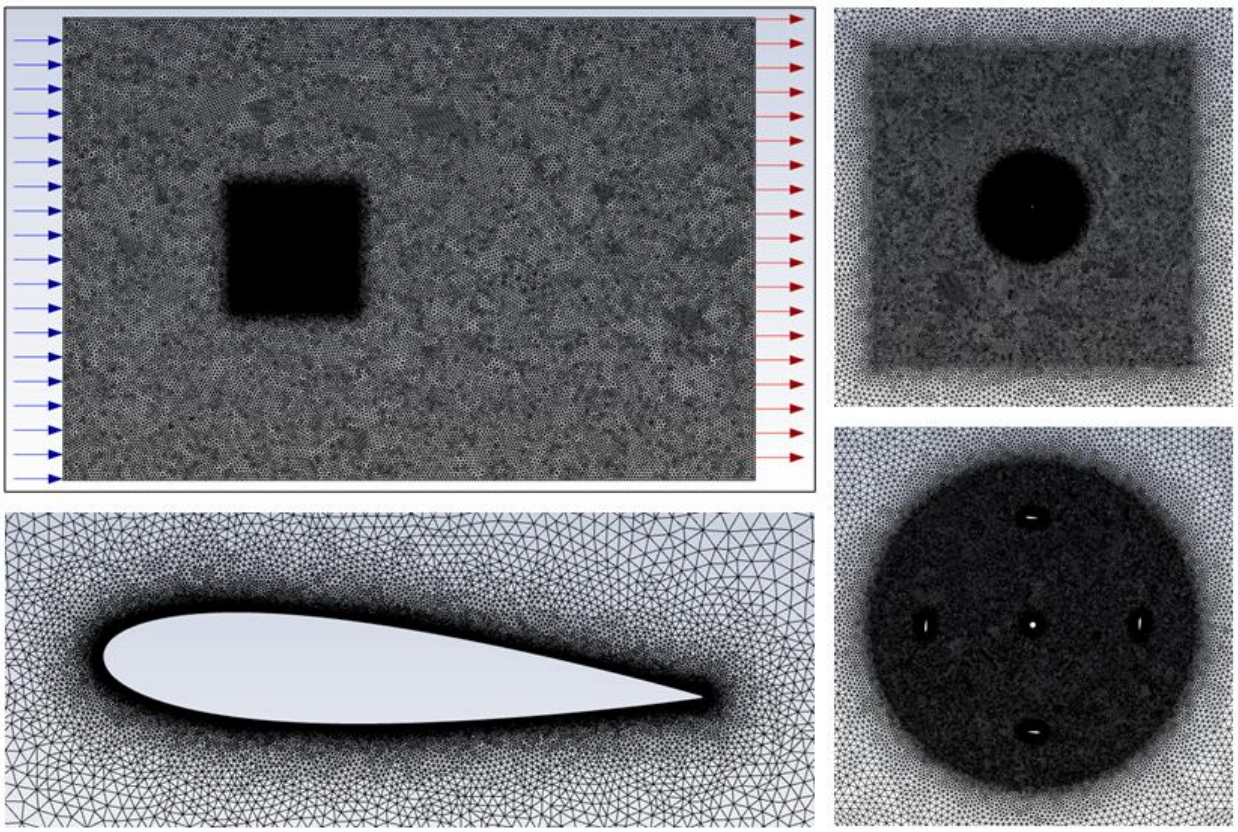


Figure 10: Mesh 3

The individual and combined computation times of the five performed simulations are shown in Table 5.

Table 5: Computational Cost of Each Simulation

Simulated free-stream velocity [m/s]	**Number of CPUs used in simulation**	**Computation time for 21 turbine revolutions**	
		[Days]	**[CPU hours]**
1	12	24.52	7061
1.5	12	15.43	4444
2	12	12.85	3699
2.5	12	9.27	2671
3	12	7.97	2296
TOTAL		**70.04**	**20171**

3.2. Data Generation

To generate training and testing data for the CNN, flow field data is exported in ASCII format at each time step of the 21st turbine revolution. These exports contain the x-velocity, y-velocity, relative total pressure, turbulent viscosity, and wall shear stress at each node within the specified export region, as well as the coordinates of each node. To minimize the quantity of data that must be stored and processed, flow field exports are limited to the region over which the CNN will be trained to infer solutions, i.e., the middle and rotor domains, as depicted in Figure 7. In addition to the flow field data, a record of the angular position of the rotor domain at each time step is also exported.

3.3. Data Pre-Processing

The following section outlines the steps taken to process the raw simulation exports into training and testing data for the CNN. Each unit of data that is used to train or test the CNN is comprised of concatenated inputs and targets, which are often referred to as boundary conditions and ground truths. Boundary conditions and ground truths may consist of one or more matrices each, depending on the number of information items that are fed into and predicted by the network. The CNN applied in the following work uses five boundary conditions and five ground truths, for a total of ten concatenated matrices. As there is an equal number of boundary conditions and ground truths, the network performs a one-to-one mapping of inputs to outputs. The contents of each boundary condition and ground truth are shown in Figure 11.

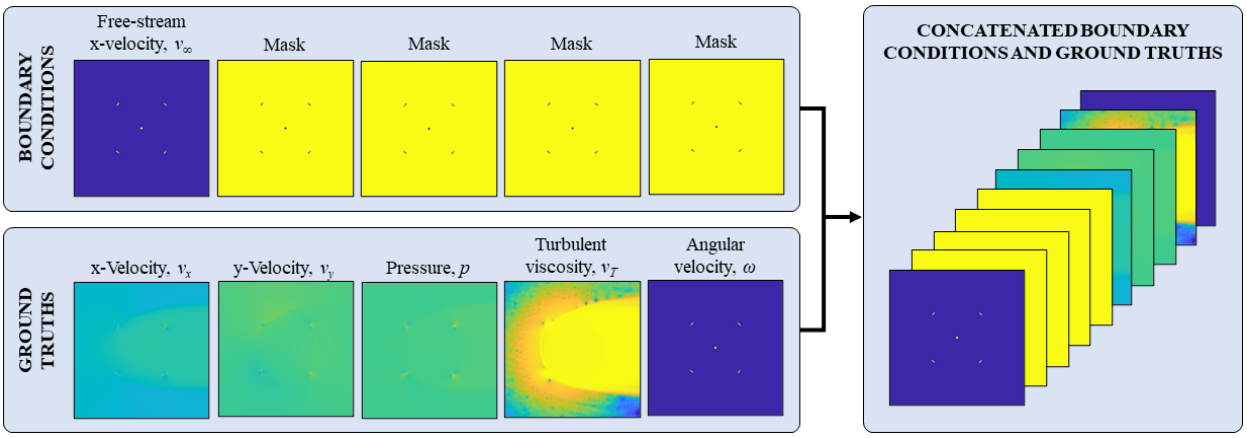


Figure 11: Boundary conditions and ground truths

The five boundary condition channels contain only two pieces of information: the free-stream velocity of the flow, which is always applied in the x-direction, and the location of solid bodies within the flow domain. The location of the solid bodies is redundantly imbedded in each input and ground truth. In the four boundary condition channels that contain only a mask, the fluid and solid regions are delineated as one and zero, respectively. For all other channels with non-

zero values in the fluid domain, solid regions are set to zero. The ground truth channels contain the flow field and rotor motion values that are to be predicted by the CNN; namely, x-velocity, y-velocity, pressure, turbulent viscosity, and the angular velocity of the rotor.

The steps taken to generate the boundary conditions and ground truths shown in Figure 11 are outlined below. For conciseness, the steps are not presented in their programmed order of execution.

i) Since the flow field data exported from the turbine simulations is produced from a non-uniform mesh, the data is interpolated onto a uniform grid, as shown in Figure 12. Interpolated variables include x-velocity v_x, y-velocity v_y, relative total pressure p, turbulent viscosity ν_T, and wall shear stress τ_{wall}. For all channels, a grid resolution of 1024 × 1024 is applied.

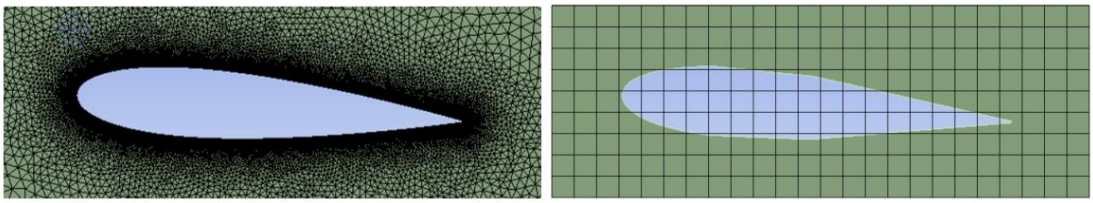


Figure 12: Interpolation of the non-uniform mesh onto a cartesian grid

The wall shear stress data is a convenient way to delineate between solid and fluid regions, since non-zero values only occur along the edges of solid bodies and form closed loops. Prior to interpolating the wall shear stress data, the data is binarized by setting all non-zero values to one. This is done to enhance the definition of solid edges. By interpolating the binarized wall shear stress data, solid and fluid regions are tidily set to one and zero, respectively, to produce the mask shown in Figure 13.

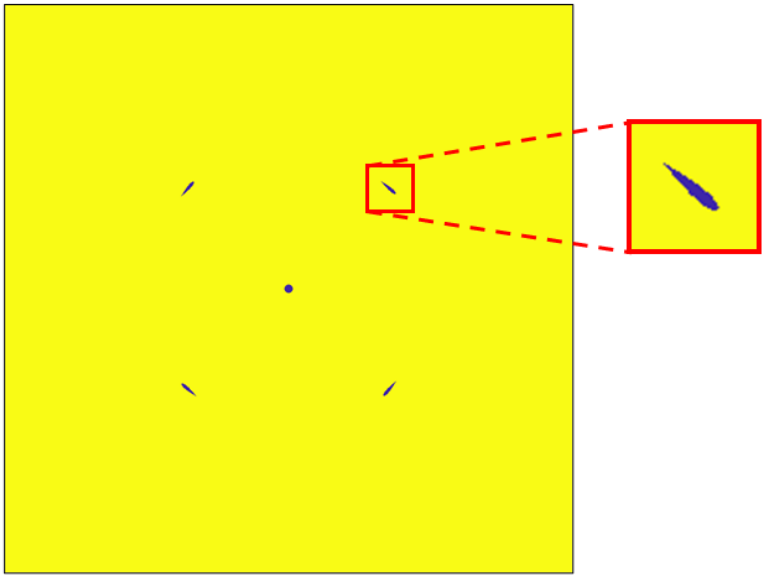


Figure 13: Delineation of the solid and fluid regions to create a [0, 1] mask

ii) The [0, 1] mask depicted in Figure 13 is used to generate the five boundary condition matrices. To produce the boundary condition channel that that will be mapped to the x-velocity ground truth, the mask is inverted and multiplied by the free-stream velocity v_∞. The remaining four boundary condition channels, which map onto the channels of y-velocity, pressure, turbulent viscosity, and angular velocity, all contain the mask.

iii) In a recent study using a CNN for the prediction of airfoil flows, the normalization of the velocity and pressure channels with respect to free-stream velocity was shown to improve the performance of the network (Thuerey et al., 2020). In this work, the interpolated x-velocity, y-velocity, and pressure matrices are normalized with respect to the magnitude of the free-stream velocity v_∞ as follows:

$$\tilde{v}_x = v_x/|v_\infty|$$

$$\tilde{v}_y = v_y/|v_\infty|$$

$$\tilde{p} = p/|v_\infty|^2$$

iv) To encode the Reynolds number in both the boundary conditions and ground truths, the free-stream velocity, x-velocity, and y-velocity matrices are scaled in the following manner: Firstly, the free-stream velocity is scaled linearly so that the minimum and maximum velocities of 1 and 3 m/s are made 0.1 and 1 m/s, respectively. In order that this scaling be reflected in the ground truths, the x-velocity and y-velocity matrices are scaled by an equivalent factor as the free-stream velocity matrix.

v) In a recent work applying a CNN to the prediction of airfoil flows, the removal of the offset from the pressure channel was shown to improve results by reducing the leaning task of the CNN (Thuerey et al., 2020). To perform this step, the mean of the pressure channel $\bar{p}$ is subtracted from each element *i* of the matrix, as follows. For a normalized pressure matrix $\tilde{p}$ with *n* elements, a zero-offset pressure matrix $\hat{p}$ is obtained as follows:

$$p_{mean} = \sum_{i} \tilde{p}_i \ /n$$

$$\hat{p} = \tilde{p} - p_{mean}$$

vi) Using the exported record of the rotor's angular position, the angular velocity of the turbine rotor is calculated at each time step. The velocity is assumed to be constant over the time step of 0.0002 s. To produce the angular velocity channel, the [0, 1] mask is inverted and multiplied by the computed angular velocity.

vii) As explained in step (i), the ground truth matrices for x-velocity, y-velocity, pressure, and turbulent viscosity were produced by interpolating the exported flow field data onto a uniform grid. Since the exported data only corresponds to points within the fluid domain, the boundaries of solid regions are lost in the interpolation process, and solid regions are filled with non-uniform values. To zero all elements within solid regions,

the ground truth matrices are multiplied by the inverse of the mask matrix, as depicted in Figure 14.

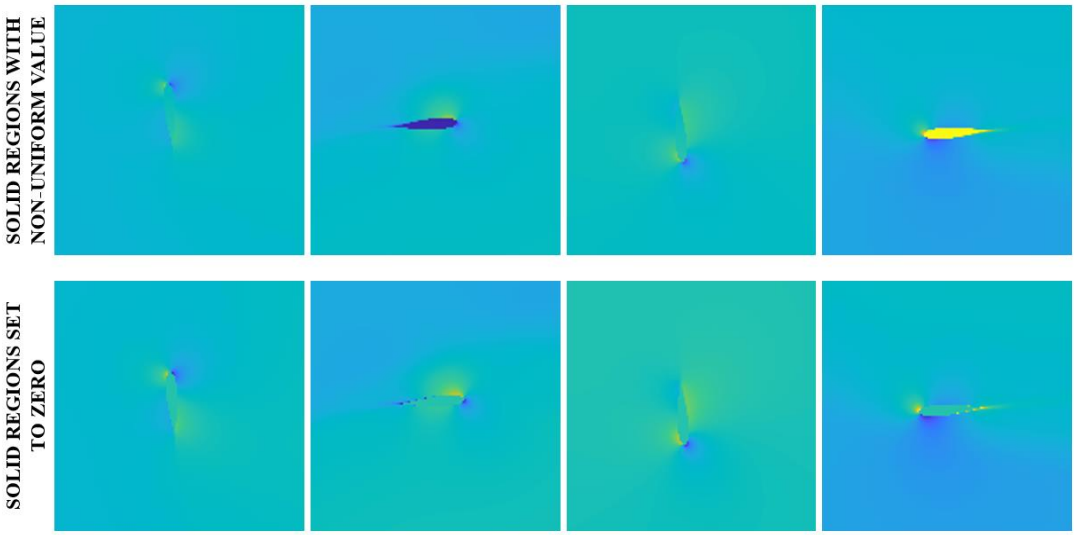


Figure 14: Zeroing of all data points within solid regions

viii) The five boundary condition matrices and five ground truth matrices are concatenated into a single array.

ix) Each of the five ground truth channels are independently scaled to the [-1, 1] range, in order to simplify the learning task of the CNN. To achieve this, the maximum absolute value for each channel is determined across all training and testing data. Each channel is then divided by its respective absolute maximum. The boundary condition channel containing the free-stream velocity is scaled by the same factor as the x-velocity ground truth, so that they correlate. The other boundary condition channels are not scaled, as they only contain a [0, 1] mask.

Due to the large dimensions of the produced arrays (10 × 1024 × 1024) and the large quantity of time steps within a single turbine revolution, training and testing data is only prepared for one quarter revolution of each simulation.

3.4. Data Post-Processing

Since the data used to train the and test the CNN were normalized and scaled, the outputs of the CNN are processed in the inverse way, as follows:

i) First, the [-1, 1] scaling that was applied in step (ix) of the pre-processing is removed from all channels. This step is applied to the CNN outputs as well as to the ground truths, in order that the discrepancy between them be calculated. To achieve this, each channel is multiplied by its respective scaling factor, i.e., the absolute maximum that was calculated in step (ix) of the pre-processing.

ii) The scaling that was applied to encode the Reynolds number in the channels, as described in step (iv) of the pre-processing, is removed. To perform this step, the free-stream velocity channel is scaled so that 0.1 m/s becomes 1 m/s, and 1 m/s becomes 3 m/s. The x-velocity and y-velocity channels are then scaled by the same factor.

iii) Lastly, to undo step (iii) of the pre-processing, the x-velocity, y-velocity, and pressure channels are denormalized with respect to the unscaled magnitude of the free-stream velocity, as shown below.

$$v_x = \tilde{v}_x \times |v_\infty|$$

$$v_y = \tilde{v}_y \times |v_\infty|$$

$$p = \tilde{p} \times |v_\infty|^2$$

3.5. Convolutional Neural Networks

CNNs are a class of ANNs that process array-based inputs through a series of layers. CNNs are frequently used for image analysis, but are also well suited to problems involving physical field phenomena. The CNNs deployed in this study are comprised of two key sections: an encoder, which compresses inputs through the process of convolution, and a decoder, which expands inputs through the process of transposed convolution, also known as deconvolution. The following section provides a basic overview of the key features of CNNs, including convolution, transposed convolution, pooling, and activation functions.

3.5.1. Convolutional Layers

Convolutional layers are comprised of filters and biases. Filters, commonly known as kernels, are matrices containing trainable weights. Each filter is accompanied by a bias, which is a trainable value. In the training of a CNN, weights and biases are fine-tuned through an iterative process of forward and backward propagation. In forward propagation, inputs are processed through the layers of the network to produce outputs, which are then compared to a pre-computed ground truth. In the process of backward propagation, the weights and biases in each layer of the network are adjusted with respect to the gradient of error between them and the final output.

In the process of convolution, a filter convolves across the elements of an input matrix with a specified stride. At each location, the coinciding elements of the input matrix and filter are multiplied and summed to produce a single element output, which is then summed with the bias and stored in an output matrix, known as a compressed feature map. The transformation of inputs to outputs in the convolutional layer is illustrated in Figure 15.

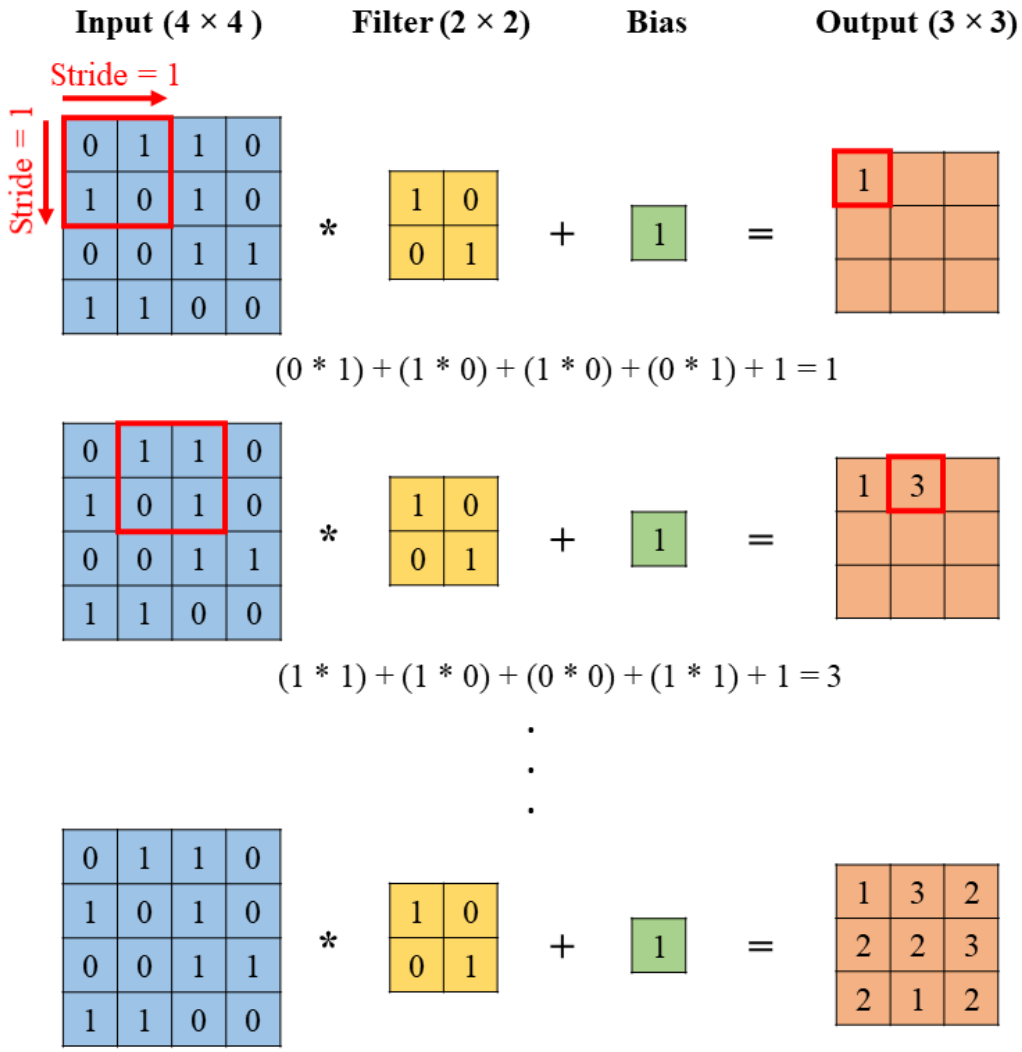


Figure 15: Convolution

Convolutional layers can also process an input with multiple channels, i.e., an array, into an output with a single channel. In this type of convolutional layer, the number of filters and biases matches the number of channels in the input array. Using the convolution method illustrated in Figure 15, an output matrix is generated for each channel of the input array. The output matrices for each channel are then summed to produce the final output matrix of the convolutional layer. The processing of a multi-channel input into a single-channel output is illustrated in Figure 16.

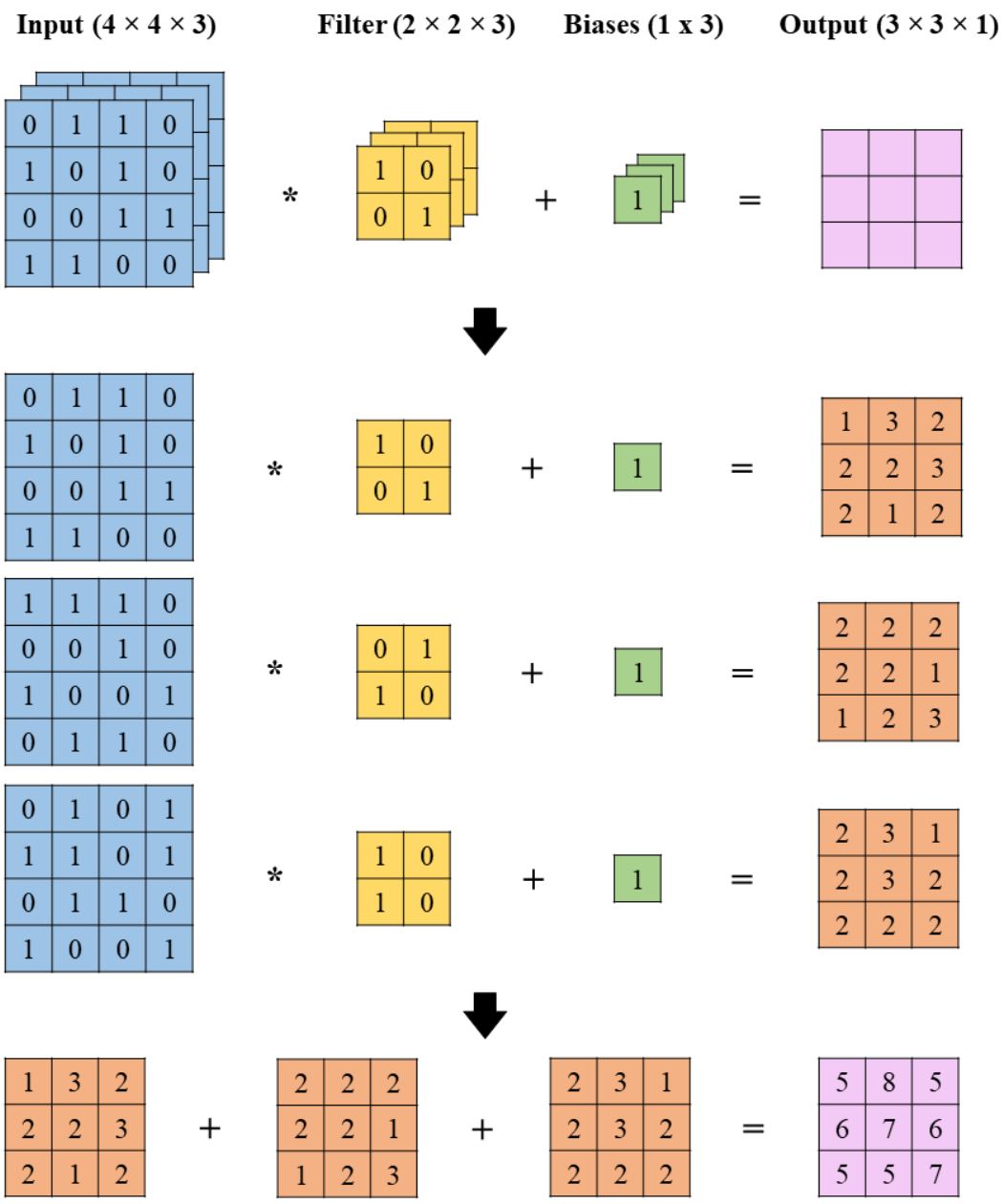


Figure 16: Convolution for multi-channel inputs

Another common technique applied in convolutional layers is the padding of input matrices with zeros. This technique is applied to improve the extraction of features around the edges of inputs. The application of a convolutional layer to a zero-padded input is demonstrated in Figure 17.

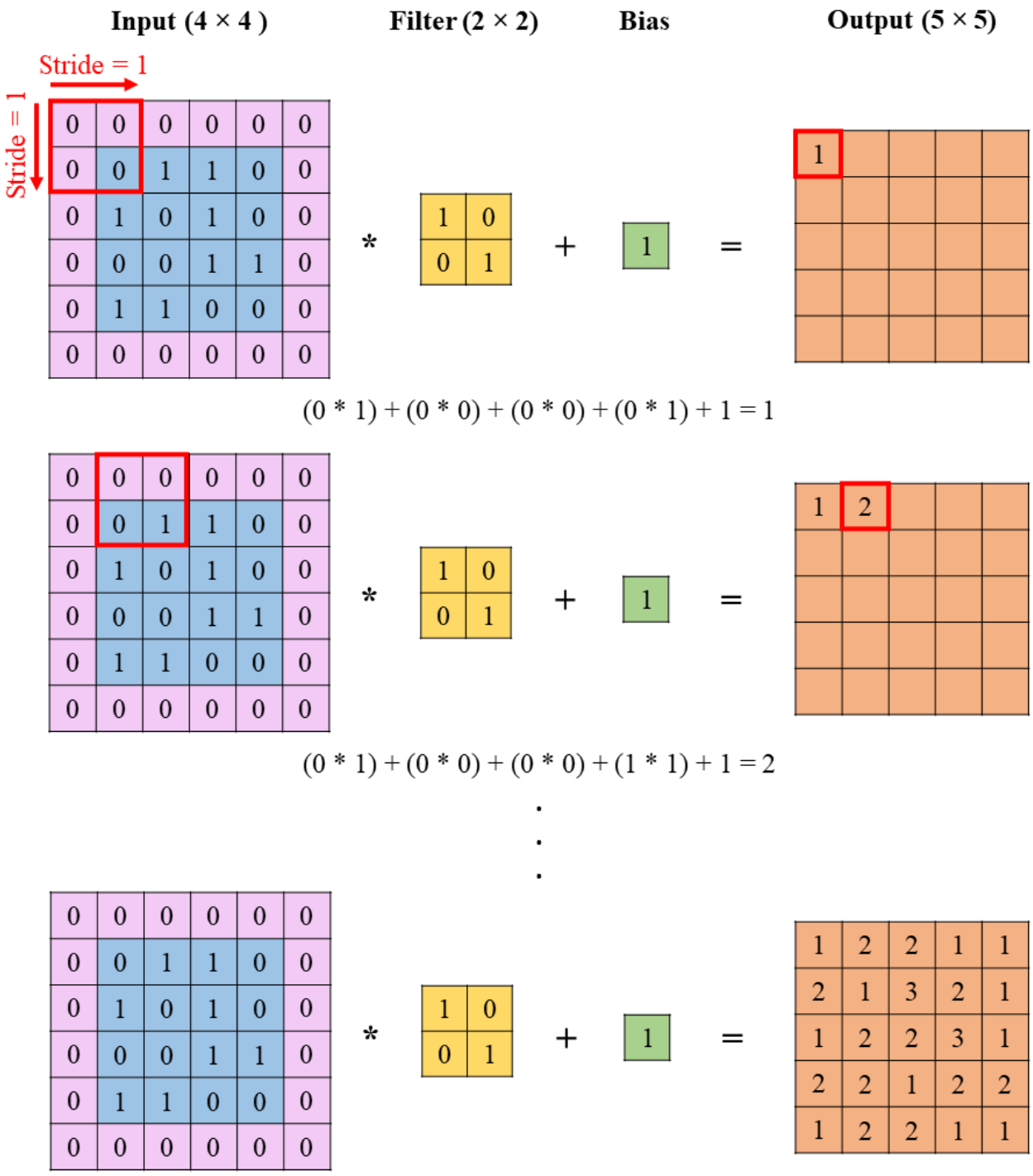


Figure 17: Convolution with zero padding

The dimensions of the output matrix can be calculated based on the dimensions of the input matrix, the thickness of the padding applied to the input, the dimensions of the filter, and the convolution stride, as shown here:

$$H_{in} = Input\ height, \qquad W_{in} = Input\ width$$

$$P = Padding\ thickness$$

$$H_f = Filter\ height, \qquad W_f = Filter\ width$$

$$S = Stride$$

$$H_{out} = Output\ height, \qquad W_{out} = Output\ width$$

$$H_{out} = \frac{H_{in} - H_f + 2P}{S} + 1$$

$$W_{out} = \frac{W_{in} - W_f + 2P}{S} + 1$$

3.5.2. Transposed Convolutional Layers

Transposed convolutional layers, or deconvolutional layers, serve the inverse function of convolutional layers. Assuming that the output of a convolutional layer is inputted to a deconvolutional layer with the same filter dimensions and stride, the output of the deconvolutional layer will have the same dimensions as the input to the convolutional layer. In the process of deconvolution, the filter strides over an empty output matrix, and is multiplied by the corresponding element of the input matrix at each location. The process is repeated until the filter is multiplied by each element in the input matrix, and the resultant matrices are summed together. The process of transposed convolution is depicted in Figure 18.

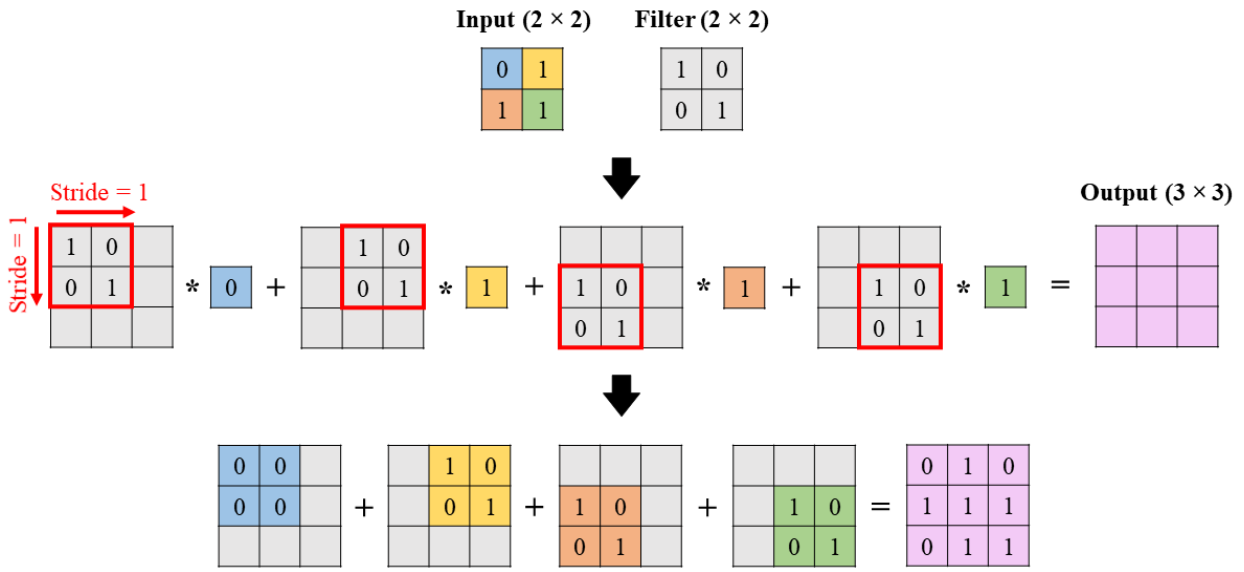


Figure 18: Transposed convolution

3.5.3. Pooling Layers

Pooling layers compress inputs by applying maximum pooling, minimum pooling, or average pooling. In pooling layers, a filter convolves across an input matrix with a specified stride, and extracts either the maximum, the minimum, or the average value of the cells it contains. Examples of maximum, minimum and average pooling are provided in Figure 19.

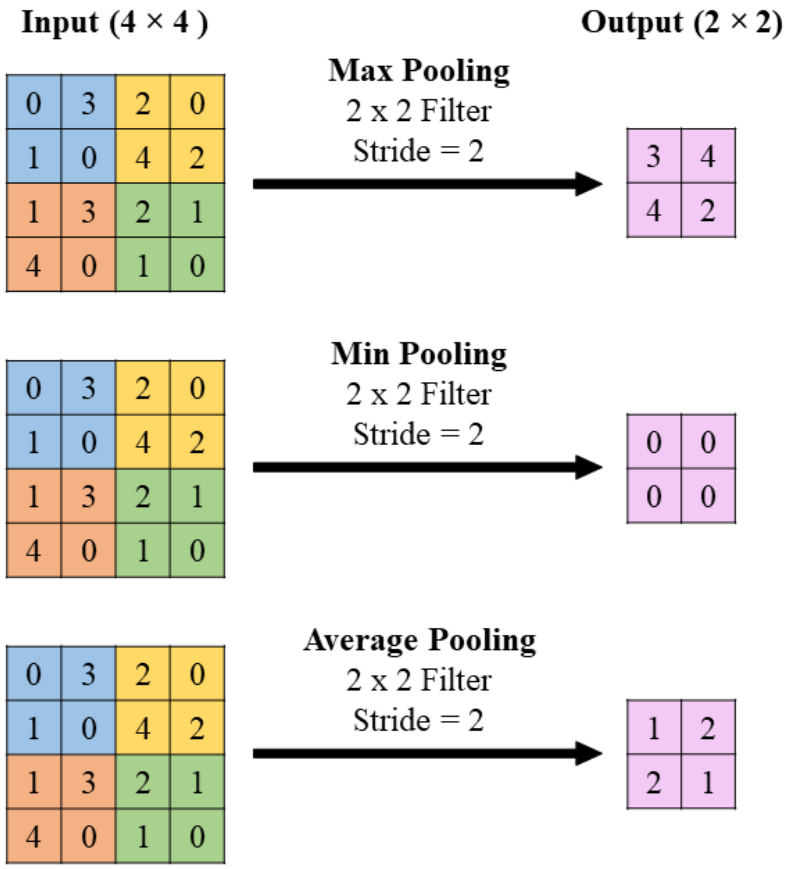


Figure 19: Maximum, minimum, and average pooling

3.5.4. Activation Functions

Activation functions are used in ANNs to facilitate the learning of complex non-linear patterns (Sharma et al., 2017). Activation functions are used between the nodes of successive network layers to process the outputs of one layer before they are used as inputs in the next layer, as depicted in Figure 20. By modulating the output of each node through a non-linear activation function, the relative importance of each node output in the mapping of inputs to outputs is encoded within the network.

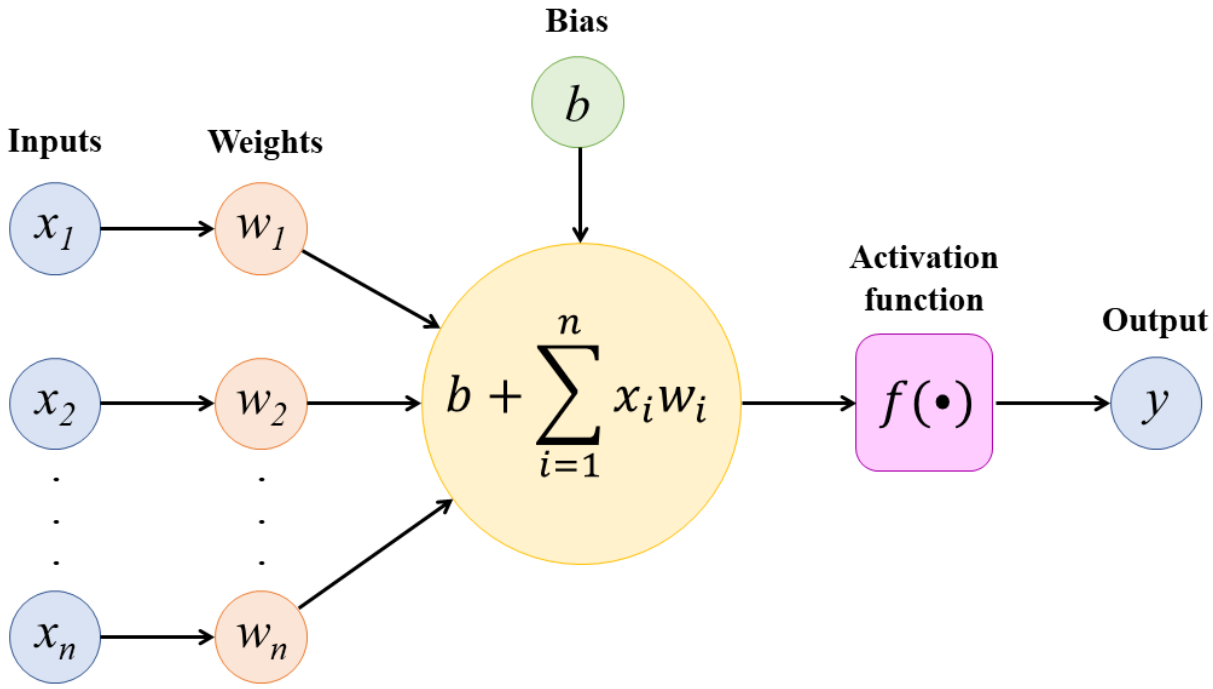


Figure 20: ANN neuron

A variety of activation functions are used in ANNs to facilitate the learning of different types of information. Some of the activation functions that are commonly deployed in ANNs are shown in Figure 21. The Rectified Linear Unit (ReLU) activation function is used throughout this work, due to its established compatibility with CNNs.

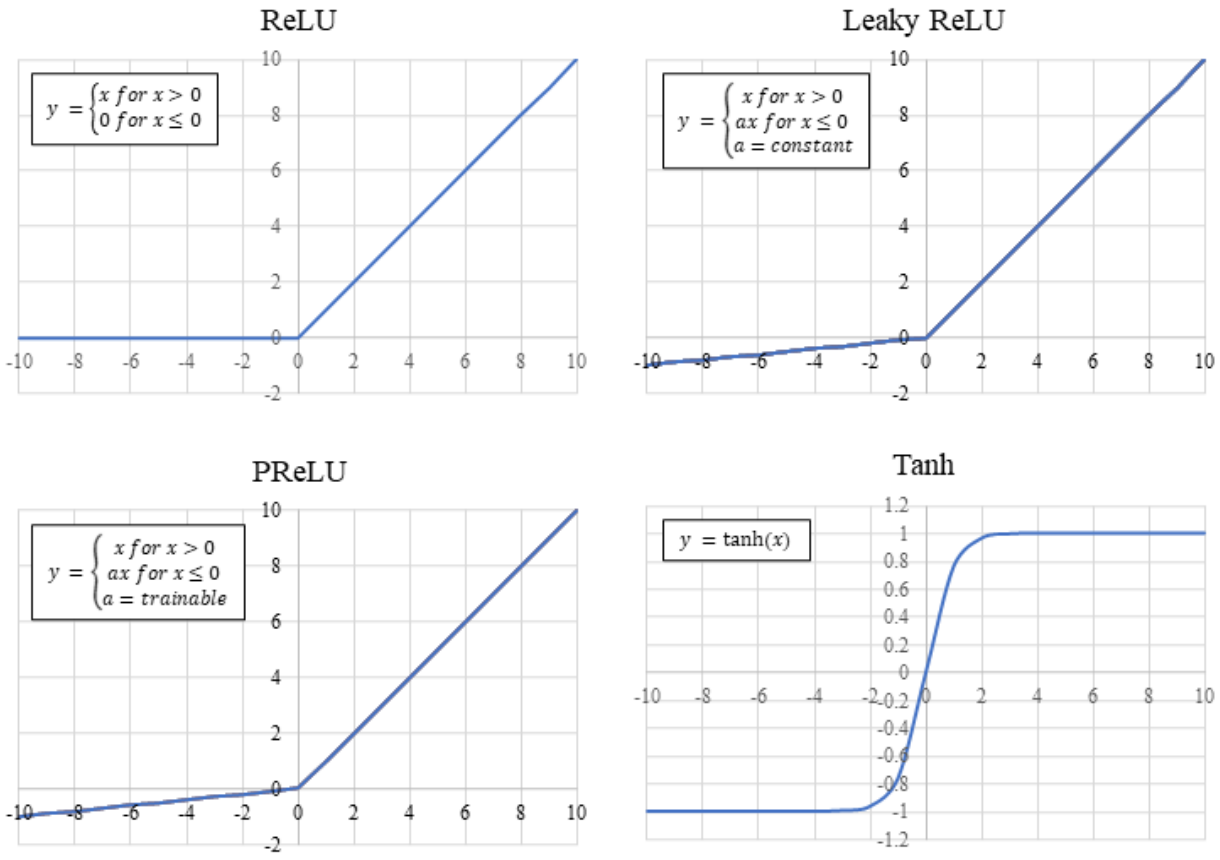


Figure 21: Common activation functions

3.6. Deep Learning Architecture

The following section provides details on the architecture and parameters of the networks applied in this research. Two different CNN architectures are developed, to suit two different input sizes. Both network architectures use convolutional and deconvolutional layers, with the ReLU activation function. Batch normalization, which is used in ANNs to augment the accuracy and speed of training (Bjorck et al., 2018), is also deployed between deconvolutional layers in both architectures. Both networks also deploy feature-wise concatenation, which consists of stacking equally sized outputs from the encoder and decoder sections of the network before they are inputted to the next deconvolutional layer. Since the two networks are used to predict the same

variables, i.e., x-velocity, y-velocity, pressure, turbulent viscosity, and rotor angular velocity, they both contain five input channels and five output channels. The larger of the two architectures, depicted in Figure 22, is designed for inputs of size 1024 × 1024.

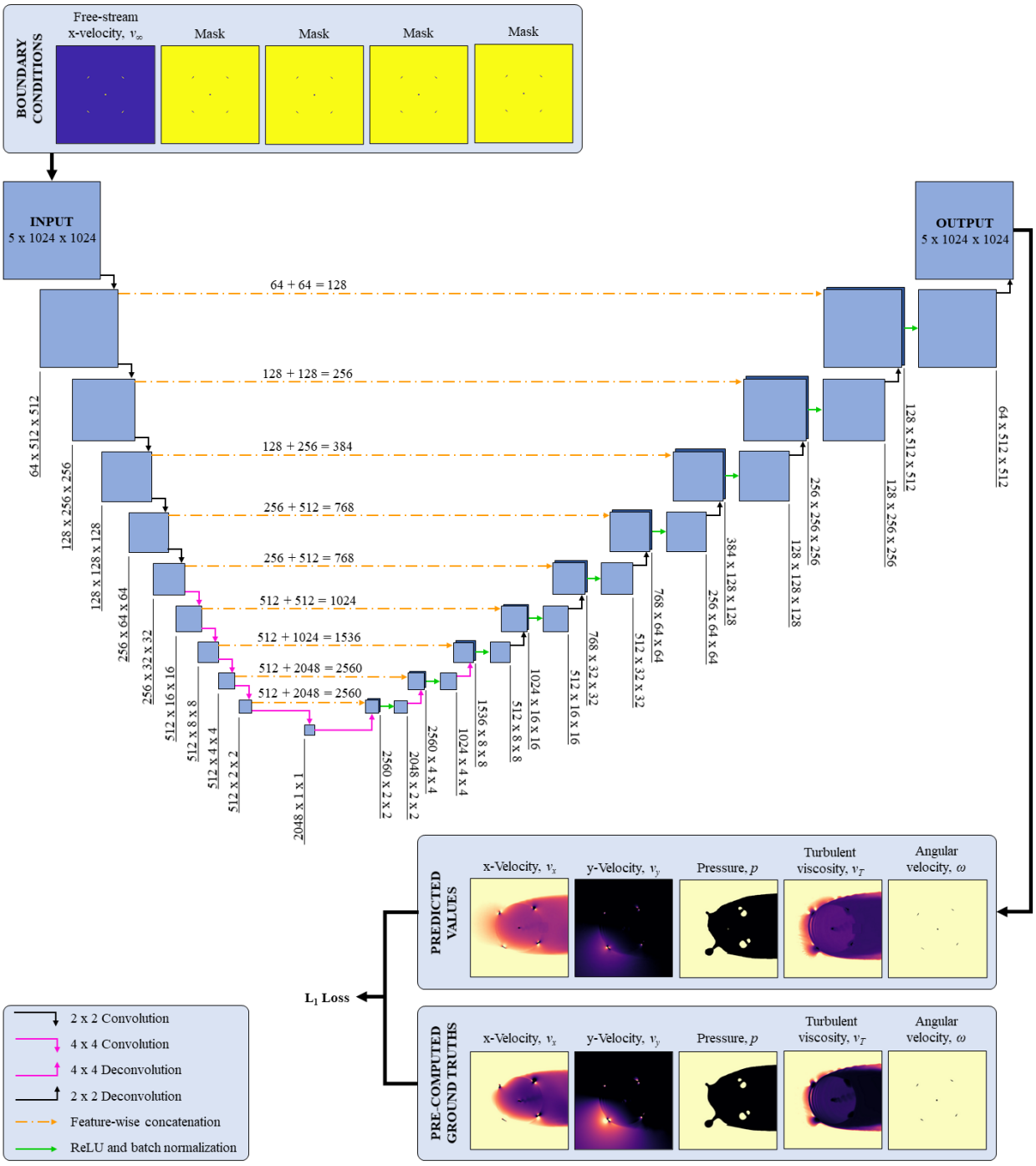


Figure 22: Large network architecture

The smaller architecture, designed for inputs of size 128 × 128, is shown in Figure 23.

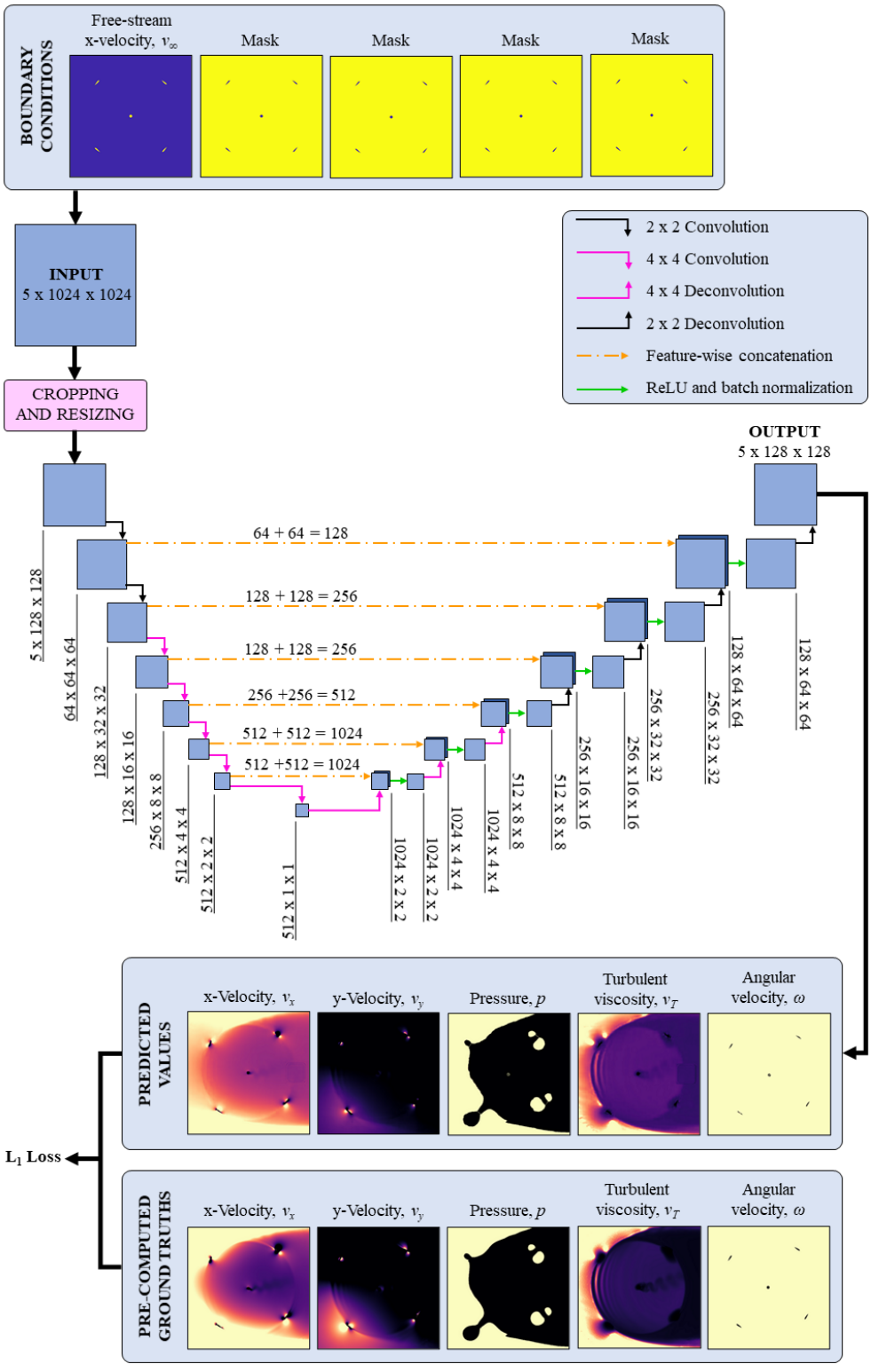


Figure 23: Small network architecture

3.7. Supervised Training

The CNNs developed in this study rely on supervised learning, which is the induction of a model from training data (Cunningham et al., 2008). For networks performing classification, training data consists of labelled data with one or more categorical variables. For networks performing prediction, such as the ones applied in this work, training data consists of pre-computed ground truths with numerical values in a continuous range. For both the CNN architectures applied in this work, training is performed using the Adam optimization algorithm (Kingma & Ba, 2014).

3.7.1. Loss Function

In the training of the CNN, the weights and biases of the network are optimized to minimize the mean absolute error between the pre-computed ground truth and the output of the CNN. The mean absolute error, commonly known as the L_1 loss, is calculated by summing the absolute error between the corresponding elements of the ground truth and output matrices, and then dividing the total by the number of matrix elements. For a ground truth matrix y and an output matrix a, each having n elements, the L_1 loss is calculated as follows:

$$L_1 = \frac{1}{n}\sum_{i=1}^{n}|y_i - a_i|$$

3.7.2. Training Parameters

Several parameters are adjusted to fine-tune the training of the CNN, including the learning rate, the dropout threshold, the batch size, and the number of epochs. The learning rate is used within the training optimization algorithm to specify the increment by which weights and biases are adjusted at each iteration of the training process. The dropout threshold is applied at each node of the network to determine whether the output of the node is inputted to the next layer, or

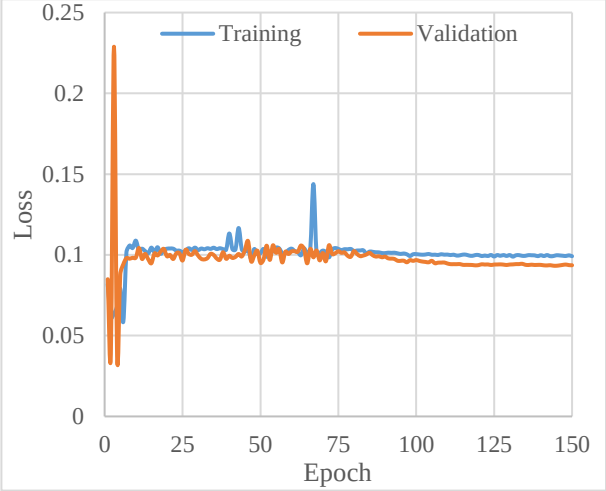
disregarded, i.e., “dropped out”. The batch size specifies the number of training inputs that are submitted to the CNN at one time. An epoch is the processing of the entire training data set, through one cycle of forward and backward propagation. Through numerous epochs, the weights and biases of the network are fine-tuned by the training optimization function.

Learning rates of 6×10^{-4} and 1×10^{-5} are used for the large and small CNNs, respectively. A dropout threshold of 0.01 and a batch size of two are applied for all networks presented herein. The large and small networks are trained using 518 and 1500 epochs, respectively. These training parameters are selected based on the metrics of training loss and validation loss. Training loss, which is calculated after each cycle of forward and backward propagation, is a measure of how well the network is fitting to the training data. Validation loss, which is calculated at the end of each epoch, is a measure of how well the network is fitting to previously unseen data, i.e., data which did not influence the weights and biases of the network. A portion of the training data is reserved for the purpose of validation; in this case, 20% of the dataset. Both the training and validation losses are calculated using the loss function described in the previous section.

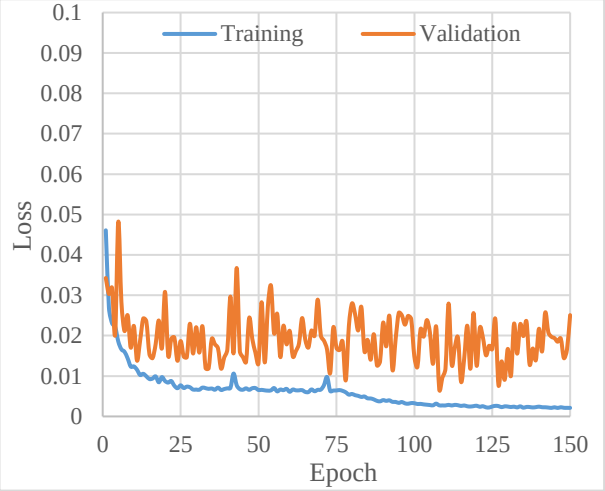
While training parameters do not typically need to be optimized with great precision, they should be adjusted so that training and validation losses are sufficiently small and stable to satisfy the objectives of the application. Of all the training parameters, learning rate is generally the most critical to tune. To start the tuning process, the learning rate, which is always a positive value below one, can be varied by orders of magnitude, generally from 10^{-1} to 10^{-6}. Having established which order of magnitude yields the lowest and most stable losses over a specified number of epochs, the coefficient of the learning rate can be further modulated using integer values between 1 and 9.

To illustrate the rationale by which a learning rate was selected for the large CNN architecture, the partial results of a parametric study are presented in Figure 24. Training and validation losses are shown for six learning rates ranging from 4×10^{-2} to 4×10^{-6}, over 150 epochs. Of the six presented cases, a learning rate of 4×10^{-3} (Case 2) is shown to produce the smallest training loss, stabilizing around 0.002. Although a learning rate of 4×10^{-4} (Case 3) yields a slightly higher training loss of about 0.003, the training loss in this case is more stable over the 150 epochs. The training losses produced by learning rates of 4×10^{-2} (Case 1), 4×10^{-5} (Case 4), and 4×10^{-6} (Case 5) stabilize around 0.1, 0.004, and 0.01, respectively. Based on these findings, the learning rate is varied within the range of 4×10^{-3} and 4×10^{-4}. A learning rate of 6×10^{-4} (Case 6) is found to produce a training loss that is comparable to that achieved in Case 2, roughly 0.002, while maintaining good loss stability over the course of the 150 epochs.

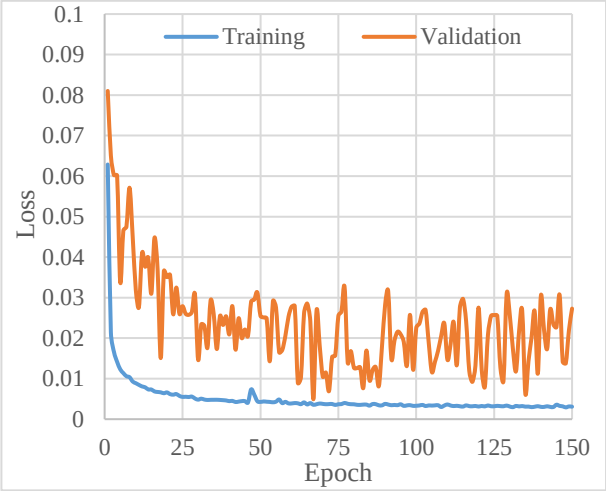
Case 1: Dropout = 0.01, Learning rate = 0.04



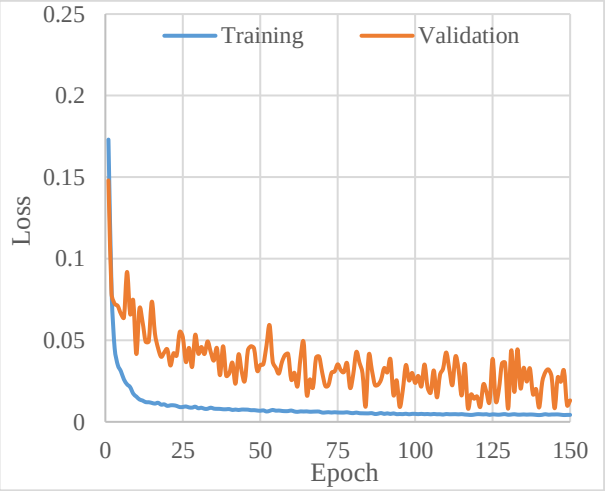
Case 2: Dropout = 0.01, Learning rate = 0.004



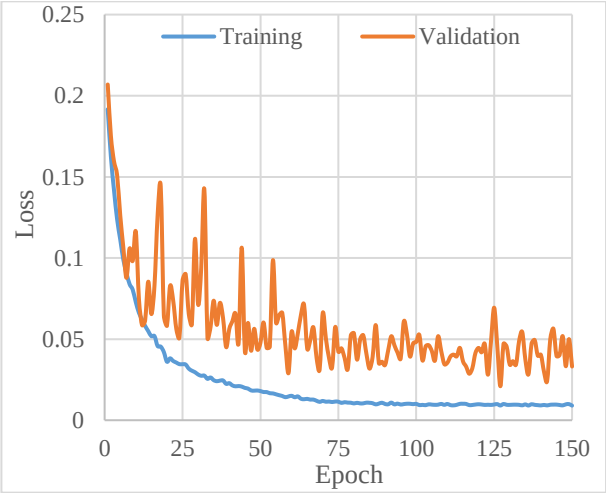
Case 3: Dropout = 0.01, Learning rate = 0.0004



Case 4: Dropout = 0.01, Learning rate = 0.00004



Case 5: Dropout = 0.01, Learning rate = 0.000004



Case 6: Dropout = 0.01, Learning rate = 0.0006

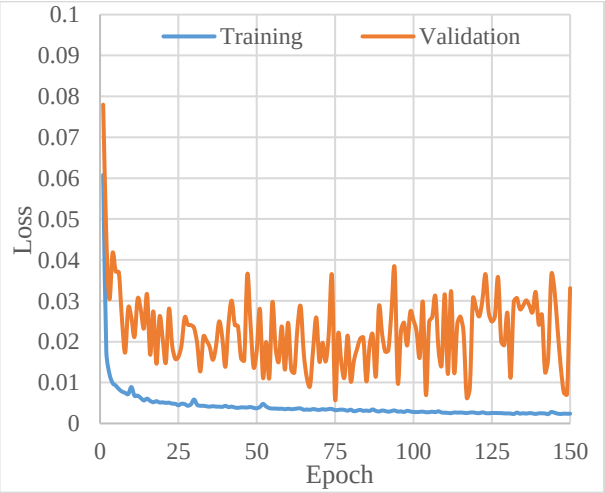


Figure 24: Parametric study for the large CNN architecture

3.7.3. Evaluation Metrics

The performance of the trained CNN models is evaluated with respect to each of the predicted unknowns, which include the field variables of x-velocity, y-velocity, pressure, and turbulent viscosity, as well as the angular velocity of the turbine. For each network channel, relative error is calculated as the sum of the absolute difference between the elements of the ground truth and output matrices, divided by the sum of the elements in the ground truth matrix. For a ground truth matrix *y* and an output matrix *a*, each having *n* elements, the relative error *RE* is calculated as follows:

$$RE = \frac{\sum_{i=1}^{n} |y_i - a_i|}{\sum_{i=1}^{n} y_i}$$

The minimum, maximum, and average relative errors are determined across all testing samples, as well as the standard deviation of the relative error. In order to evaluate the performance of the trained models with respect to different training scenarios, i.e., different free-stream velocities, the maximum, minimum, and average relative errors are also calculated across the testing samples of each individual testing case, in addition to the standard deviation of relative error.

4. Results and Discussion

The following section presents the results of the CFD simulations, as well as the results of three trained CNN models. Prior to discussing the significance of the findings, two important caveats should be reiterated. Firstly, this study is based on 2-D simulations. Since 2-D simulations constrain vorticity to the x-y plane, they underestimate the amount of turbulent kinetic energy added to the wake by the turbine, and overestimate the amount of energy transmitted to the turbine shaft. Moreover, since 2-D simulations necessarily omit the geometry of the rotor struts, the drag force produced by the struts is not accounted for, and the angular velocity of the turbine is further exaggerated. Secondly, throughout this study, the turbine is modelled as free-wheeling, i.e., with zero counter-torque applied to the turbine shaft. Since the instantaneous power of a rotating body is the product of the applied torque and the angular velocity of the body, a free-wheeling turbine generates no power. Although the simulations performed in this study neglect power generation and are not capable of accurately replicating the rotor motion and wake characteristics observed in field experiments, the insights gleaned from this work are expected to facilitate the adaptation of the research methodology to high-fidelity, 3-D simulations with Maximum Power Point Tracking (MPPT).

4.1. CFD Results

To generate training and testing data for the CNN, five 2-D RANS simulations were performed in ANSYS Fluent®, for free-stream velocities of 1, 1.5, 2, 2.5, and 3 m/s. Contour plots are provided below for the free-stream velocity of 3 m/s. The gradients of x-velocity, y-velocity, relative total pressure, and turbulent viscosity are shown in Figure 25, Figure 26, Figure 27, and Figure 28, respectively.

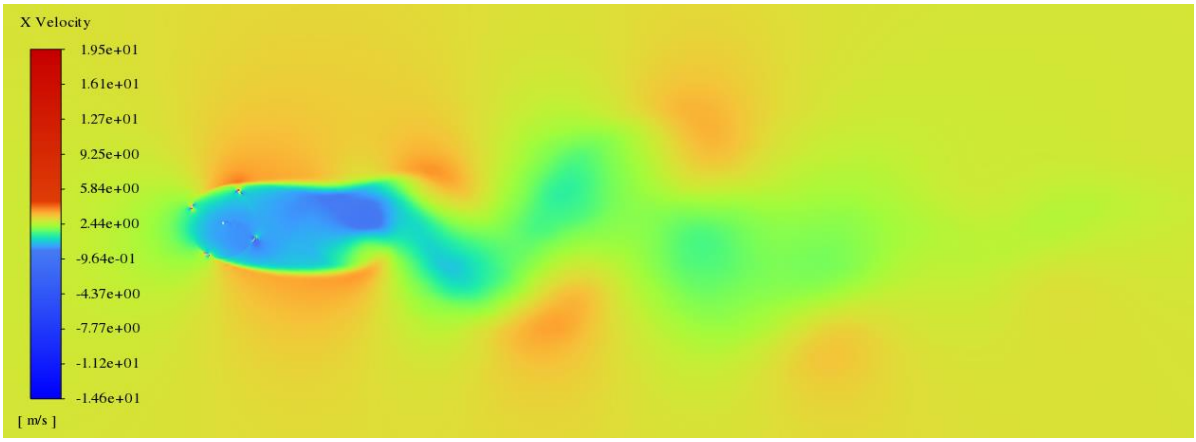


Figure 25: Gradient of x-velocity in the turbine wake, for v_∞ = 3 m/s

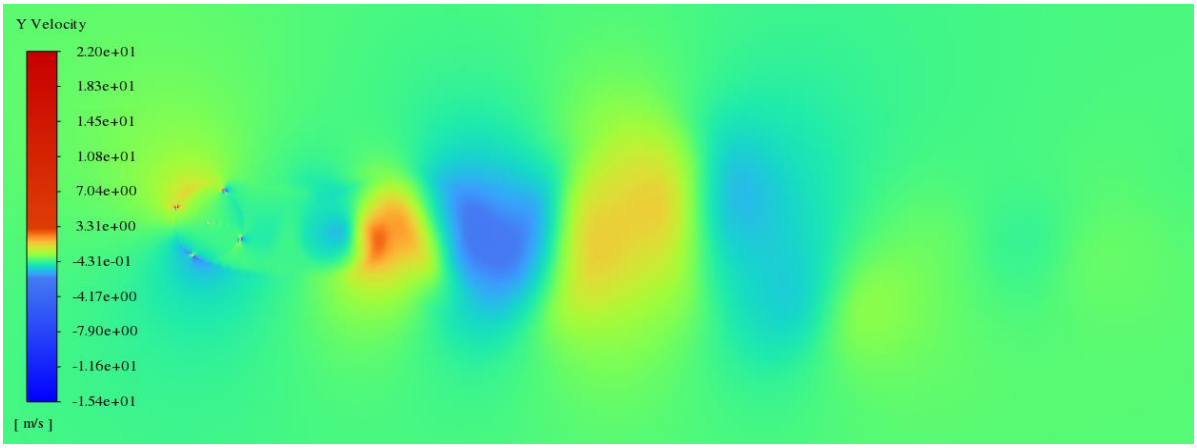


Figure 26: Gradient of y-velocity in the turbine wake, for v_∞ = 3 m/s

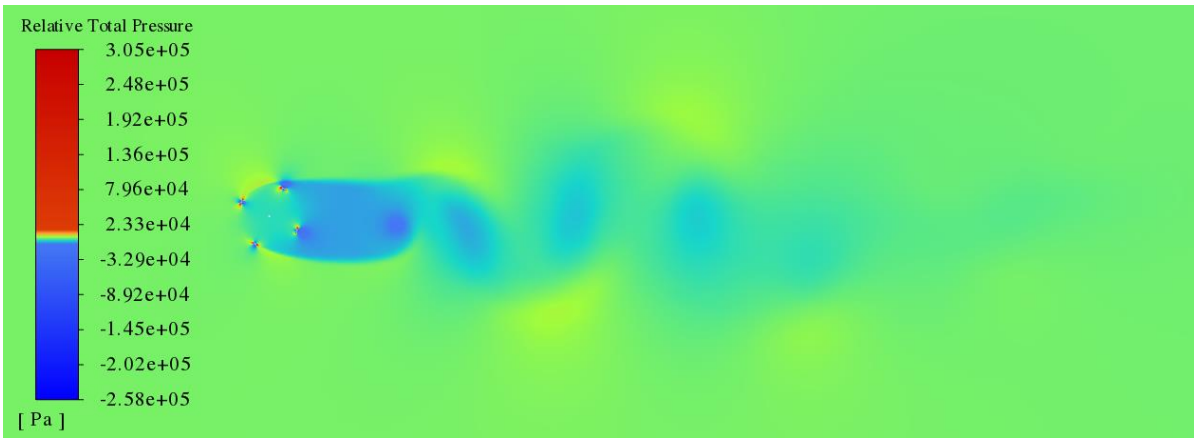


Figure 27: Gradient of relative total pressure in the turbine wake, for v_∞ = 3 m/s

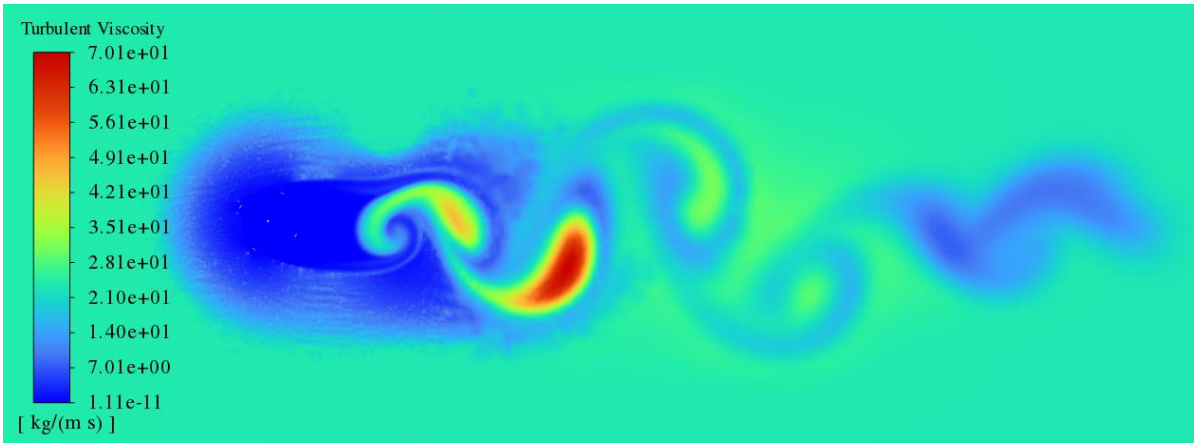


Figure 28: Gradient of turbulent viscosity in the turbine wake, for v_∞ = 3 m/s

For a free-stream velocity of 3 m/s, the wake recovery distance is approximately 23 m, or roughly 15 turbine diameters, measured from the center of the rotor. The gradients of x-velocity, y-velocity, relative total pressure, and turbulent viscosity are shown in greater detail around the rotor, in Figure 29, Figure 30, Figure 31, and Figure 32, respectively.

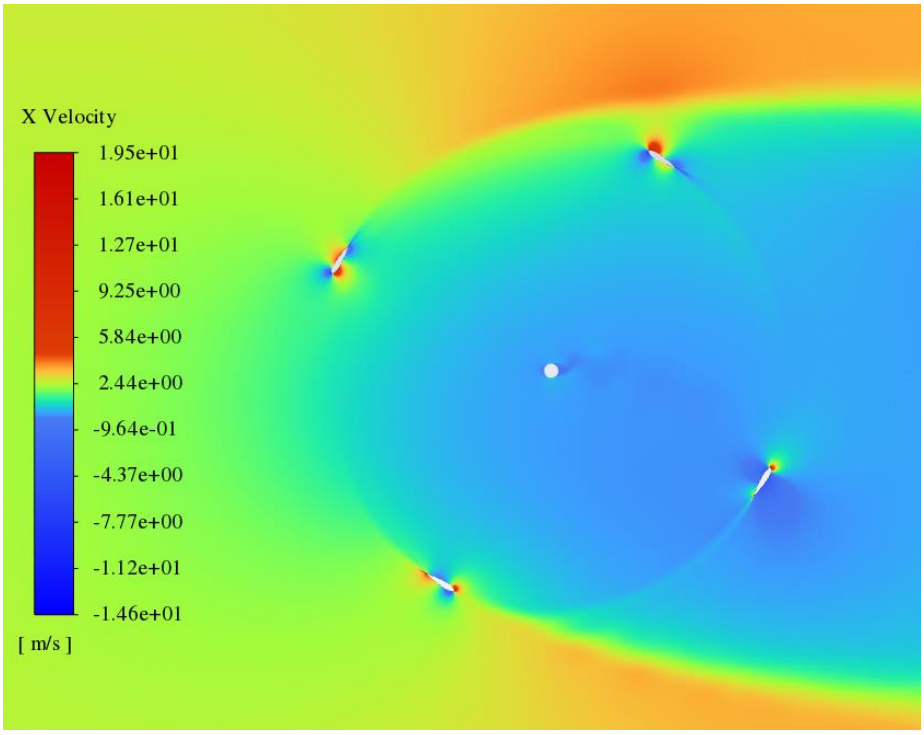


Figure 29: Gradient of x-velocity around the rotor, for v_∞ = 3 m/s

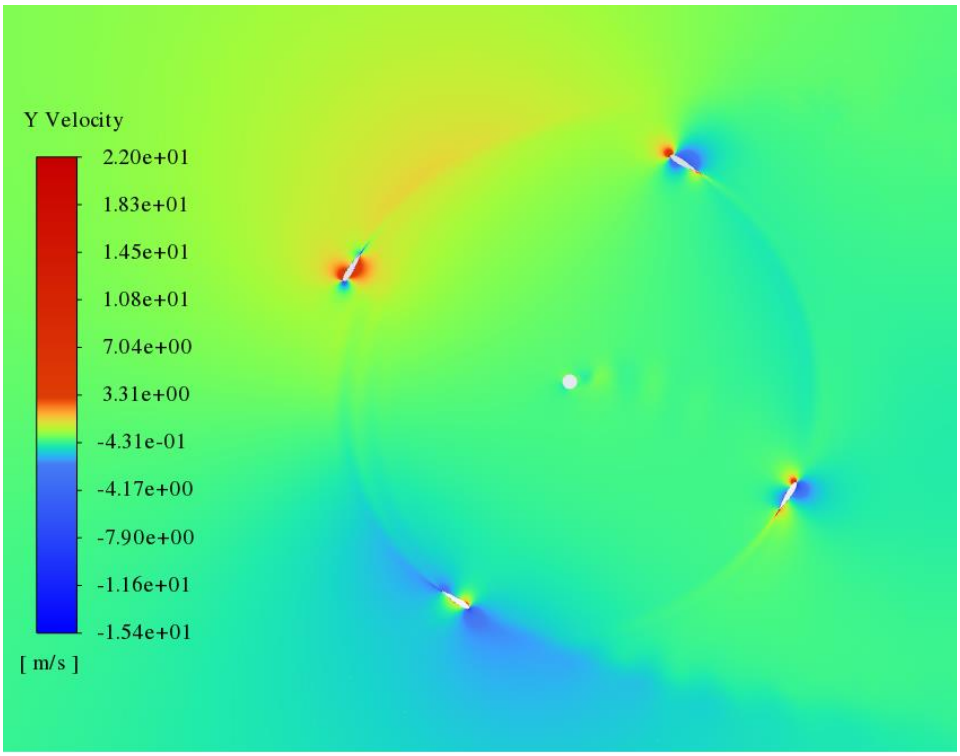


Figure 30: Gradient of y-velocity around the rotor, for v_∞ = 3 m/s

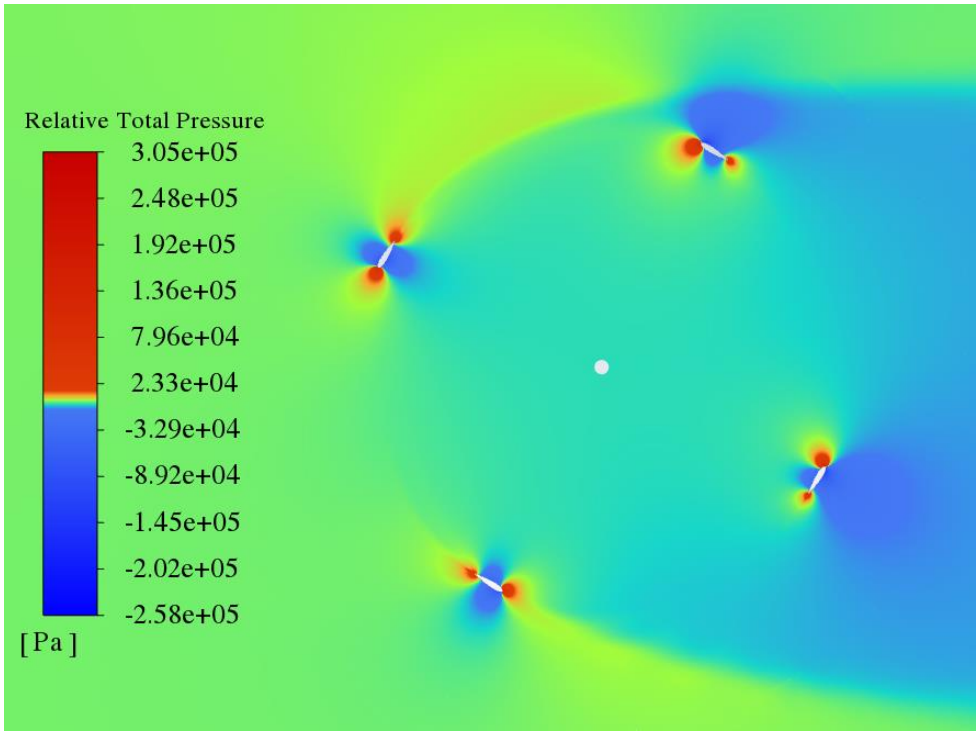


Figure 31: Gradient of relative total pressure around the rotor, for v_∞ = 3 m/s

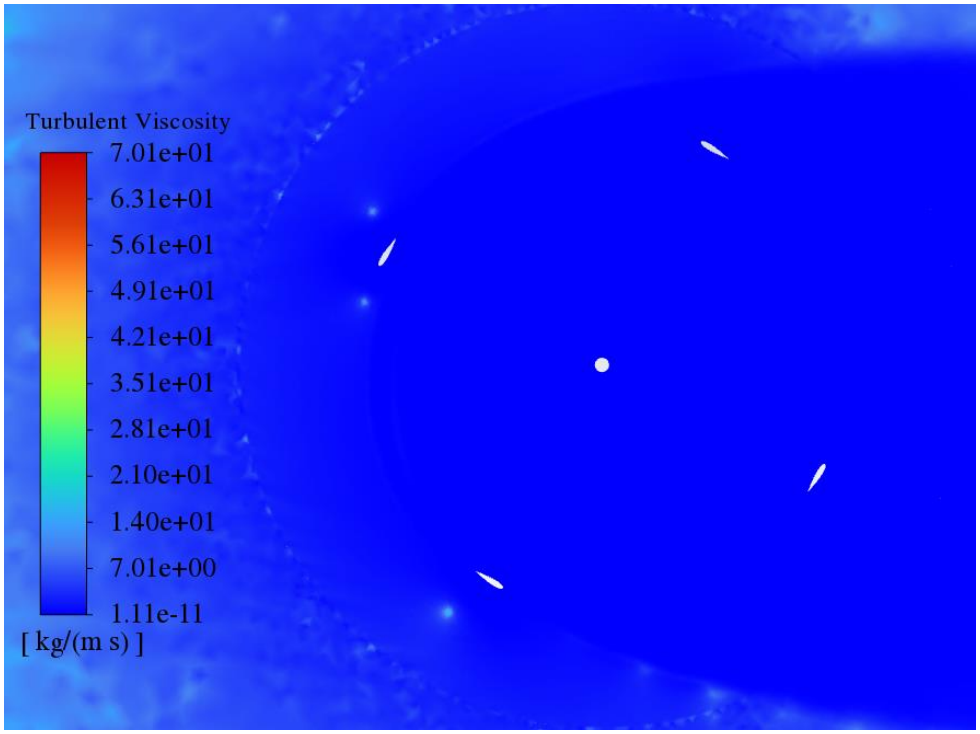


Figure 32: Gradient of turbulent viscosity around the rotor, for v_∞ = 3 m/s

As can be seen in Figure 32, there is a slight discontinuity in the turbulent viscosity gradient at the outer boundary of the rotor domain, in the region where the dynamic mesh is applied. To address this issue, it is recommended that the dynamic mesh parameters be adjusted so that a smoother grid is maintained as the rotor domain spins.

The gradients of x-velocity, y-velocity, relative total pressure and turbulent viscosity are shown around one of the foils in Figure 33, Figure 34, Figure 35, and Figure 36, respectively, for a free-stream velocity of 3 m/s.

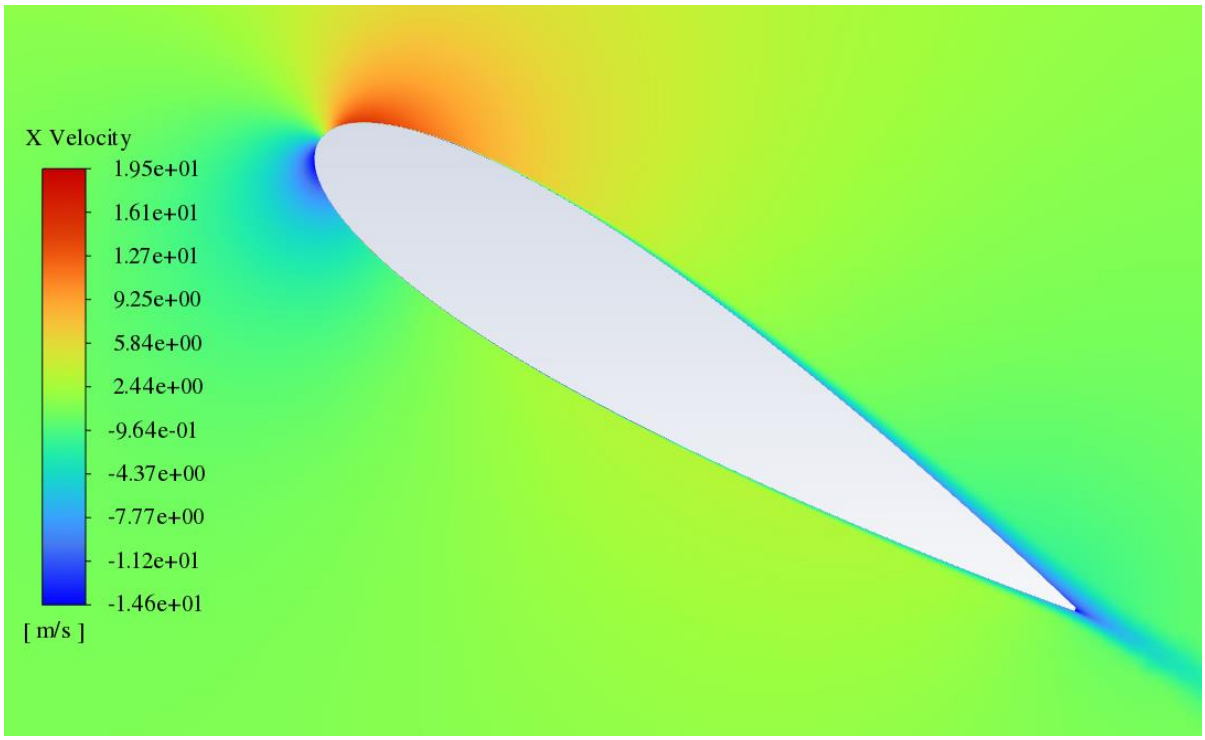


Figure 33: Gradient of x-velocity around one of the foils, for v_∞ = 3 m/s

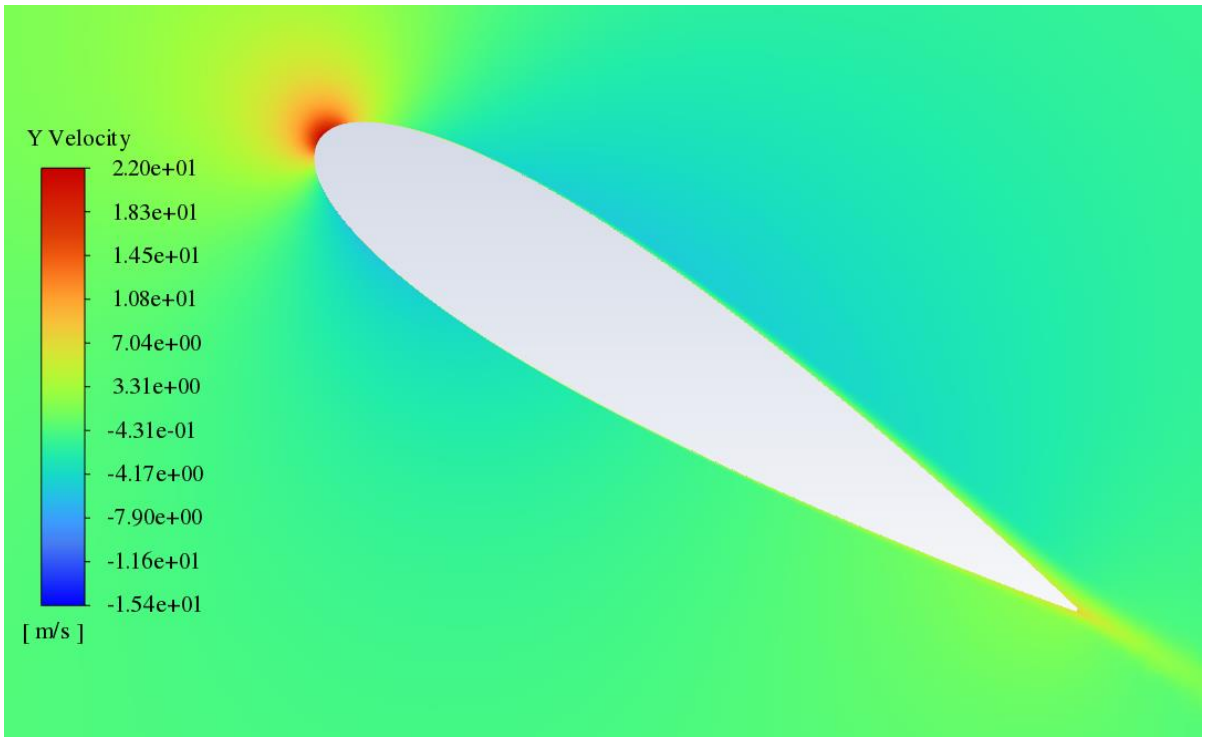


Figure 34: Gradient of y-velocity around one of the foils, for v_∞ = 3 m/s

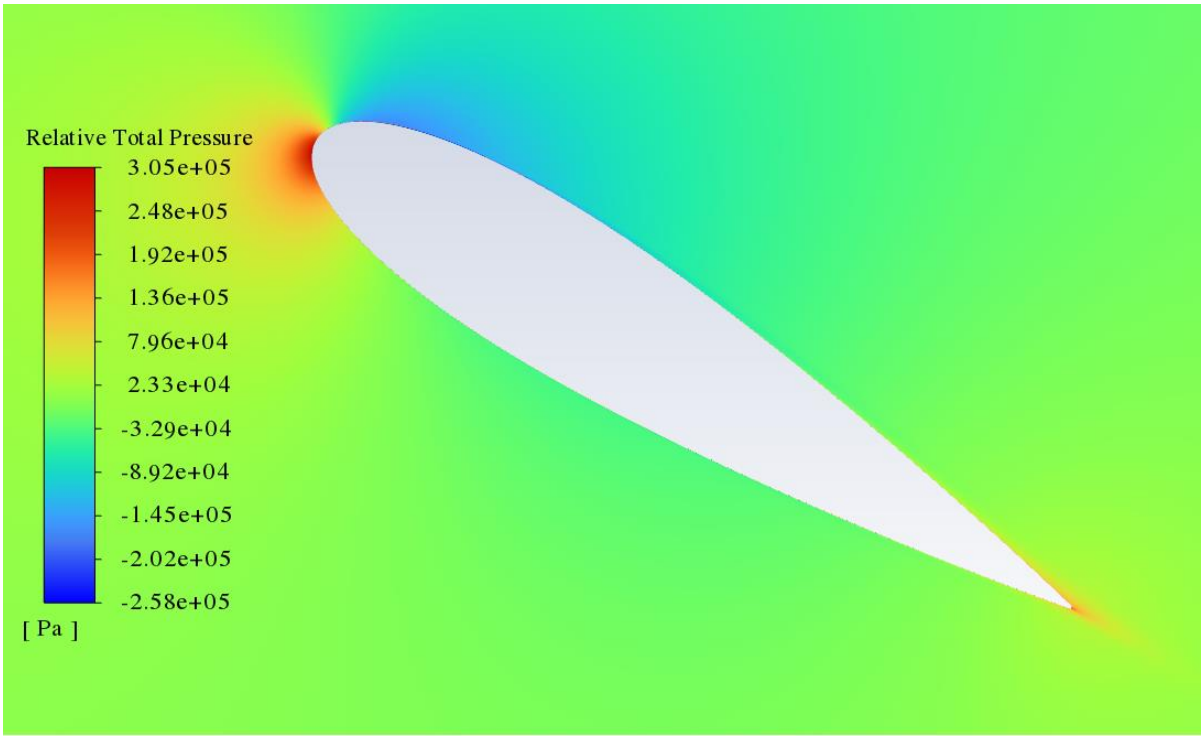


Figure 35: Gradient of relative total pressure around one of the foils, for v_∞ = 3 m/s

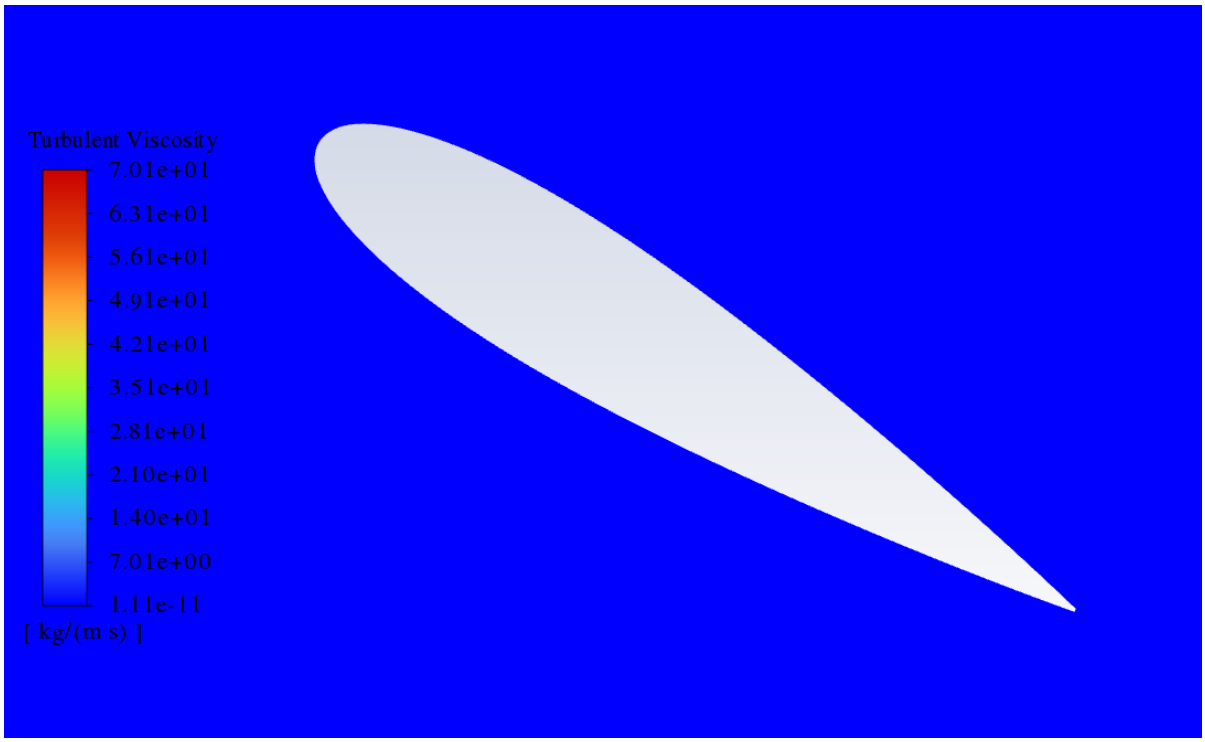


Figure 36: Gradient of turbulent viscosity around one of the foils, for v_∞ = 3 m/s

The average angular velocity of the rotor in the 21st revolution is shown in Table 6, for each of the five simulated free-stream velocities.

Table 6: Average Angular Velocity of the Rotor for Each Simulated Free-Stream Velocity

Free-stream velocity [m/s]	**Average angular velocity of rotor [RPM]**
1	64
1.5	98
2	131
2.5	167
3	198

To calculate the first-cell width and height of the grid within the boundary layers, the maximum rotational velocity of the rotor was assumed to be 90 RPM, for a free-stream velocity of 3 m/s. As can be seen from Table 6, the rotational velocity of the turbine at a free-stream velocity of 3 m/s exceeds the assumed value by a factor of more than two. In order to achieve x^+ and y^+ values below 30 and 1, respectively, for a free-stream velocity of 3 m/s and a rotational velocity of 200 RPM, it is recommended that the first-cell height and width be reduced from 6×10^{-5} and 2×10^{-6} to 3×10^{-5} and 1×10^{-6}, respectively. While the rotational velocity of the rotor is expected to be less for a 3-D simulation with MPPT, an assumed rotational velocity of 200 RPM could serve as a conservative estimate.

4.2. Model 1: Large Architecture with Two Training Cases

Model 1 was developed using the large CNN architecture depicted in Figure 22, and was trained with simulation results for the free-stream velocities of 1 and 3 m/s. The network was trained using 386 samples, each containing an array with dimensions of $10 \times 1024 \times 1024$. Training was performed over 518 epochs, with a learning rate of 6×10^{-4}, a dropout threshold of 0.01, and a batch size of two. The training and validation losses throughout the training process are shown in Figure 37. Over the course of the training process, the validation loss stabilized at approximately 0.02. Training took roughly seven hours using an NVIDIA GeForce RTX 3060 GPU.

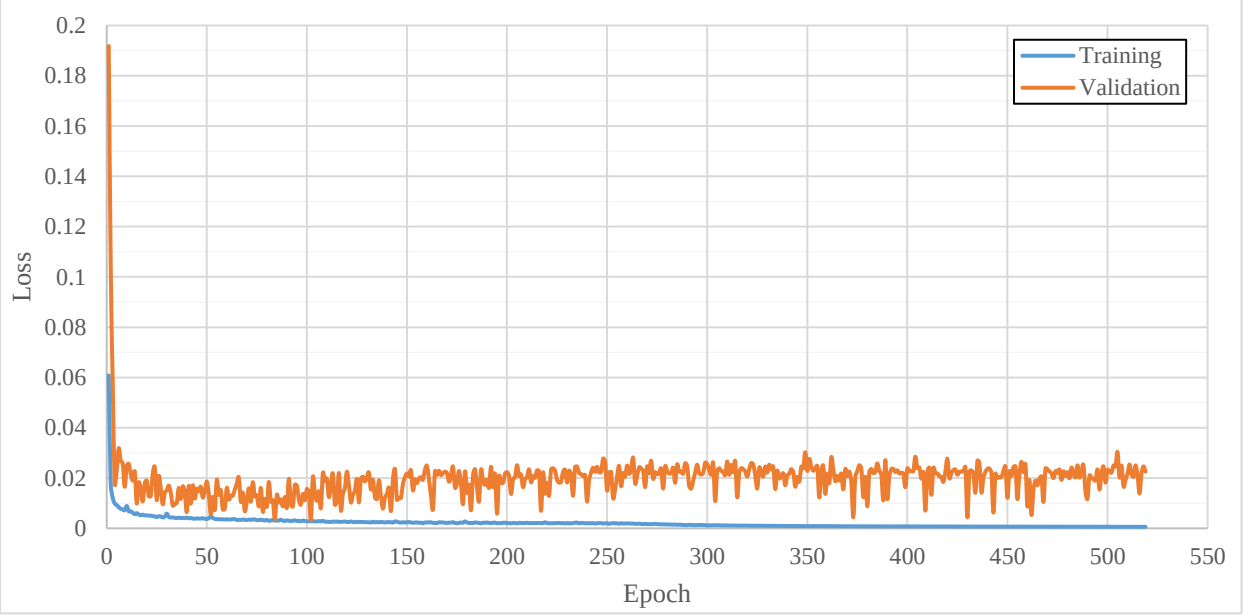


Figure 37: Training and validation losses over 518 epochs (Model 1)

Model 1 was tested using simulation results for free-stream velocities of 1.5, 2, and 2.5 m/s. Thirty-eight testing samples were selected for each free-stream velocity, to obtain a total of 114 testing samples. The minimum, maximum, and average relative errors across all testing samples are shown in Table 7, as well as the standard deviation of the relative error.

Table 7: Relative Error of Each Channel, for the Combined Testing Cases of v_∞ = 1.5, 2, and 2.5 m/s (Model 1)

Channel variable	Relative error			
	Minimum [%]	**Maximum [%]**	**Average [%]**	**Standard deviation**
x-Velocity, v_x	13.52	33.41	24.58	0.0822
y-Velocity, v_y	11.15	53.37	35.92	0.178
Pressure, p	4.51	43.25	8.37	0.0415
Turbulent viscosity, v_T	19.12	31.50	26.73	0.0509
Angular velocity, ω	0.08	1.63	0.63	0.00589

As can be seen in Table 7, Model 1 produces relatively high relative errors for the variables of x-velocity, y-velocity, and turbulent viscosity. In comparison, the relative error associated with the predicted angular velocity is small. This is to be expected, since the prediction of a scalar is less complex than the prediction of a gradient.

In order to evaluate the performance of Model 1 with respect to different free-stream velocities, the minimum, maximum, and average relative errors are also calculated across the samples from each individual testing case, as well as the standard deviation of the relative error. The relative errors with respect to the free-stream velocities of 1.5, 2, and 2.5 m/s are provided in Table 8, Table 9, and Table 10, respectively.

Table 8: Relative Error of Each Channel, for the Testing Case of v_∞ = 1.5 m/s (Model 1)

Channel variable	Relative error			
	Minimum [%]	**Maximum [%]**	**Average [%]**	**Standard deviation**
x-Velocity, v_x	13.54	33.41	32.73	0.0320
y-Velocity, v_y	12.01	53.37	52.07	0.0667
Pressure, p	4.51	15.88	5.69	0.0182
Turbulent viscosity, v_T	20.38	30.15	29.37	0.0153
Angular velocity, ω	0.08	1.45	0.25	0.00218

Table 9: Relative Error of Each Channel, for the Testing Case of v_∞ = 2 m/s (Model 1)

Channel variable	Relative error			
	Minimum [%]	**Maximum [%]**	**Average [%]**	**Standard deviation**
x-Velocity, v_x	26.97	27.25	27.08	0.000641
y-Velocity, v_y	42.88	44.09	43.35	0.00282
Pressure, p	9.11	43.25	11.62	0.0546
Turbulent viscosity, v_T	30.56	31.50	30.92	0.00222
Angular velocity, ω	0.08	0.48	0.23	0.00123

Table 10: Relative Error of Each Channel, for the Testing Case of v_∞ = 2.5m/s (Model 1)

Channel variable	Relative error			
	Minimum [%]	**Maximum [%]**	**Average [%]**	**Standard deviation**
x-Velocity, v_x	13.52	27.07	13.93	0.0219
y-Velocity, v_y	11.15	43.93	12.34	0.0527
Pressure, p	6.77	12.47	7.80	0.00999
Turbulent viscosity, v_T	19.12	31.28	19.91	0.0192
Angular velocity, ω	0.17	1.63	1.41	0.00249

From the results for the different training velocities, it can be observed that the predicted flow field variables have significant error in all testing cases. The relative errors in the x-velocity and y-velocity channels are highest for the free-stream velocity of 1.5 m/s, and lowest for the free-stream velocity of 2.5 m/s. The error in the pressure channel is relatively stable across all three training cases, although a high maximum relative error is observed for the free-stream velocity of 2 m/s.

The pre-computed ground truths and network predictions for a free-stream velocity of 1.5 m/s are juxtaposed in Figure 38, for two different rotor angles. For both testing samples, the steepness of the gradients in the x-velocity, y-velocity, and turbulent viscosity channels is noticeably underpredicted. A small square region of discontinuity is visible in the x-velocity and turbulent viscosity predictions, around the farthest downstream point of the foil path.

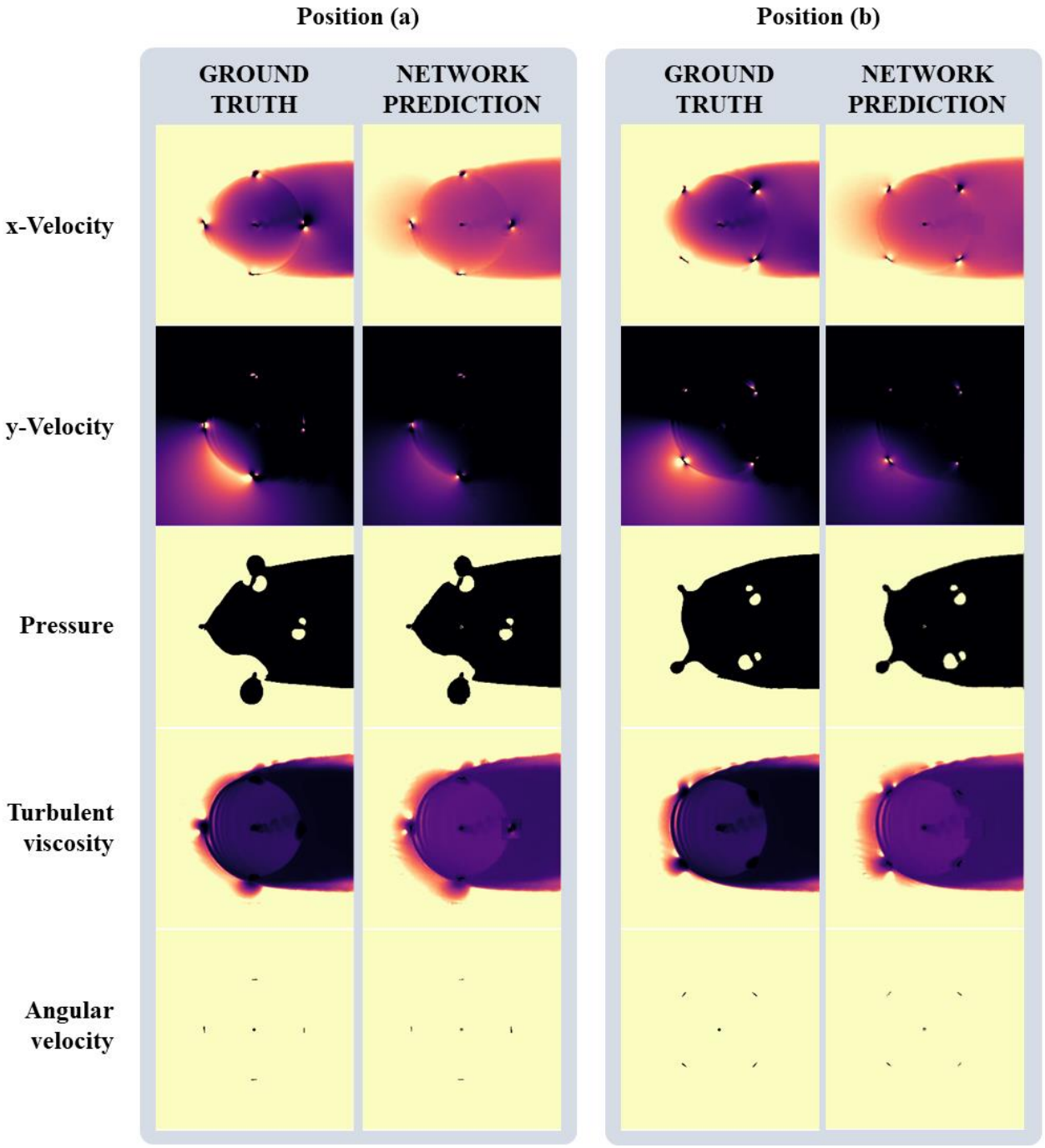


Figure 38: Ground truth and network predictions at two rotor positions, for v_∞ = 1.5 m/s (Model 1)

The pre-computed ground truths and network predictions for a free-stream velocity of 2 m/s are shown side by side in Figure 39, for two different rotor angles. Again, the steepness of

the gradients in the x-velocity, y-velocity, and turbulent viscosity channels is noticeably underpredicted for both testing samples.

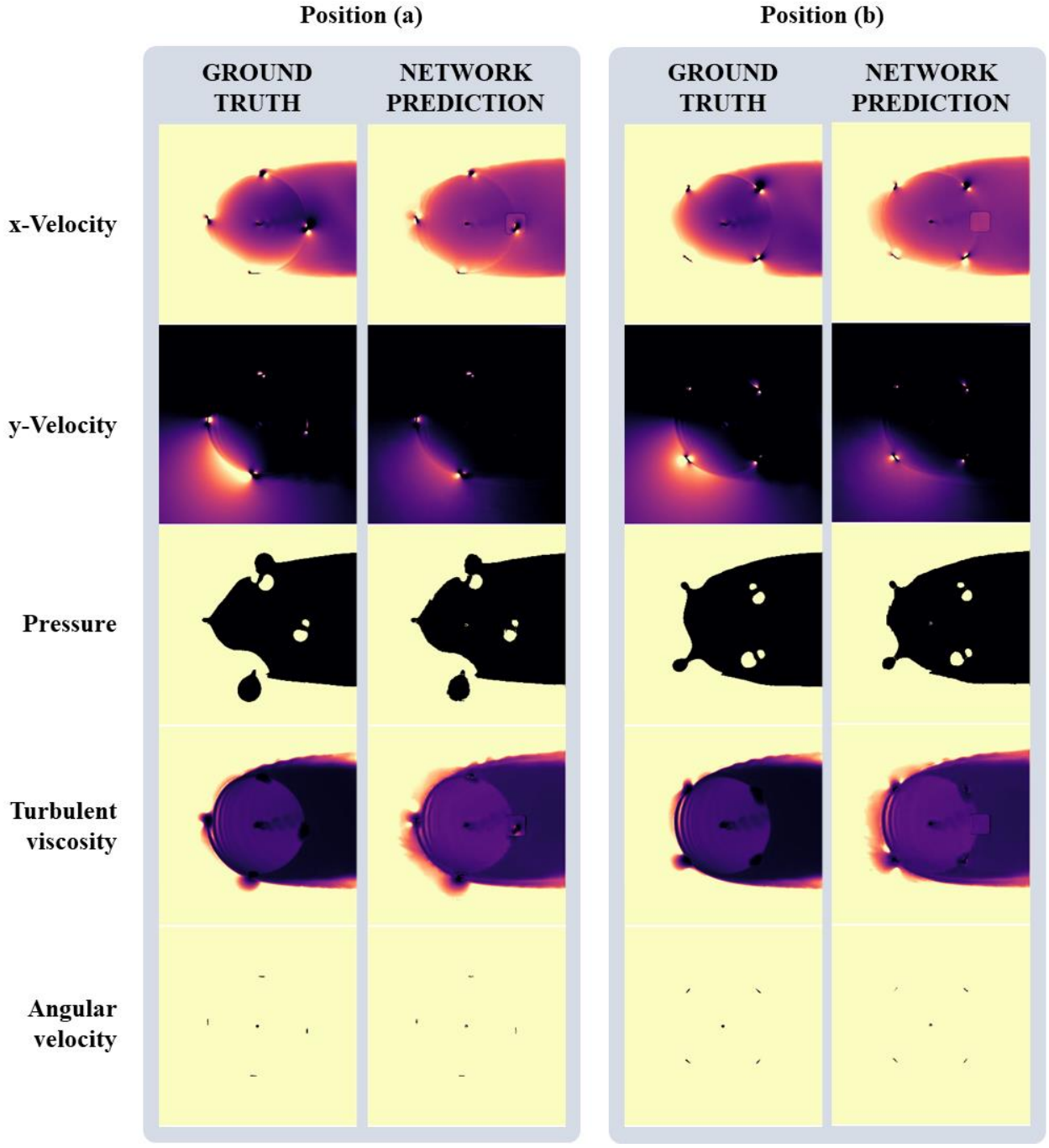


Figure 39: Ground truth and network predictions at two rotor positions, for v_∞ = 2 m/s (Model 1)

The pre-computed ground truths and network predictions for a free-stream velocity of 2.5 m/s are shown side by side in Figure 40, for two different rotor angles. For both testing samples, the steepness of the gradients in the x-velocity, y-velocity, and turbulent viscosity channels is overpredicted, contrary to the results obtained for the free-stream velocities of 1.5 and 2 m/s.

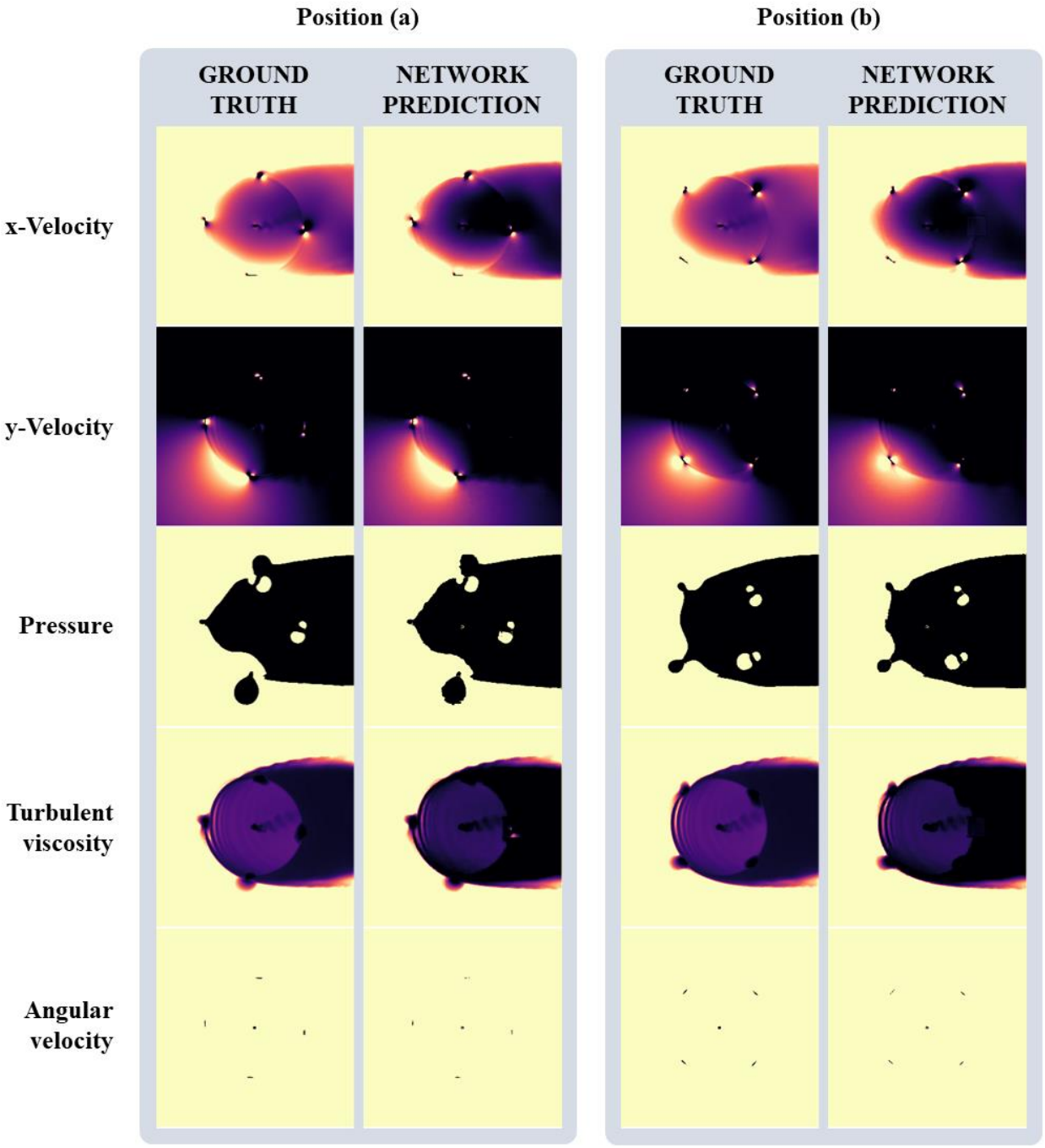


Figure 40: Ground truth and network predictions at two rotor positions, for v_∞ = 2.5 m/s (Model 1)

4.3. Model 2: Small Architecture with Two Training Cases

Model 2 was developed using the small CNN architecture depicted in Figure 23, and was trained with simulation results for the free-stream velocities of 1 and 3 m/s. The network was trained using 386 samples, each containing an array with dimensions of 10 × 128 × 128. Training was performed over 1500 epochs, with a learning rate of 1×10^{-5}, a dropout threshold of 0.01, and a batch size of two. The training and validation losses throughout the training process are shown in Figure 41. Over the course of the training process, the validation loss stabilized at approximately 0.0008. Training took roughly three hours using an NVIDIA GeForce RTX 3060 GPU.

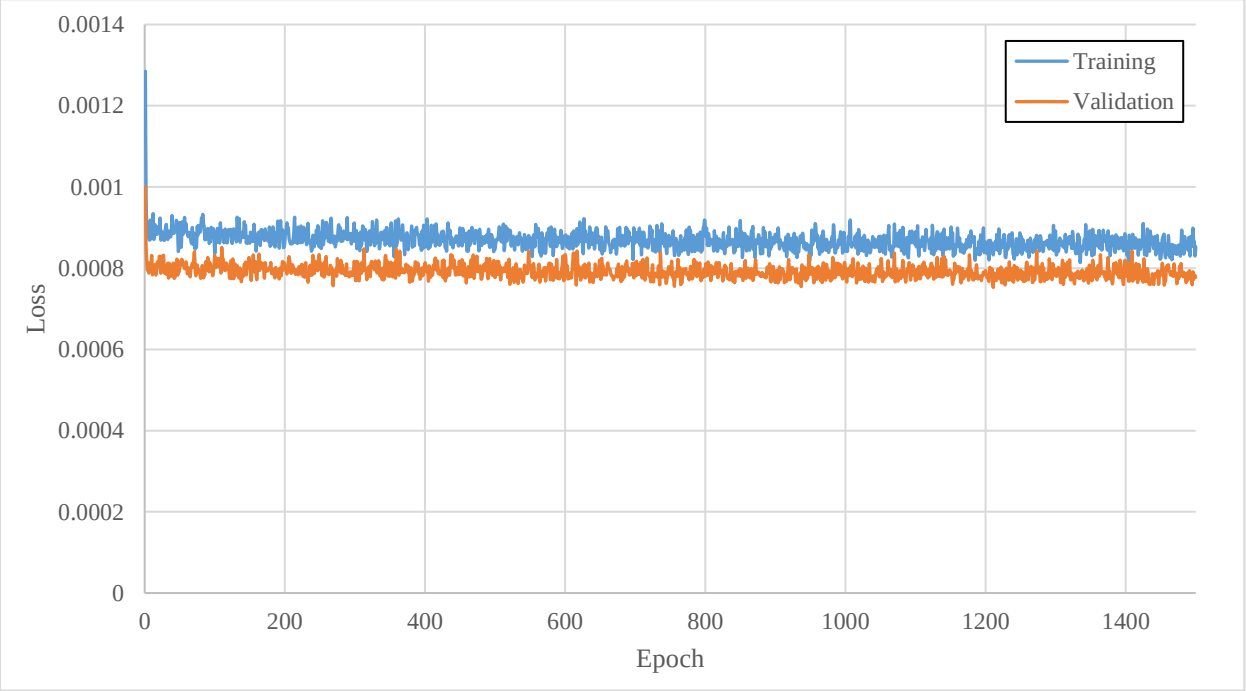


Figure 41: Training and validation losses over 1500 epochs (Model 2)

Model 2 was tested using simulation results for free-stream velocities of 1.5, 2, and 2.5 m/s. Thirty-eight testing samples were selected for each free-stream velocity, to obtain a total of 114 testing samples. The minimum, maximum, and average relative errors calculated across all testing samples are shown in Table 11, as well as the standard deviation of the relative error.

Table 11: Relative Error of Each Channel, for the Combined Testing Cases of v_∞ = 1.5, 2, and 2.5 m/s (Model 2)

Channel variable	Relative error			
	Minimum [%]	**Maximum [%]**	**Average [%]**	**Standard deviation**
x-Velocity, v_x	13.30	26.66	20.01	0.0514
y-Velocity, v_y	12.77	24.85	18.34	0.0432
Pressure, p	10.43	25.05	17.60	0.0385
Turbulent viscosity, v_T	6.98	10.48	8.77	0.00972
Angular velocity, ω	0.32	1.97	1.33	0.00530

As demonstrated by the relative errors shown in Table 7 and Table 11, Model 2 achieved significantly better results than Model 1 for the channels of x-velocity, y-velocity, and turbulent viscosity. Interestingly, in Model 2, the average relative error of the pressure channel is comparable to relative errors in the x-velocity and y-velocity channels. This contrasts the results obtained by Model 1, in which the average relative error in the pressure channel is markedly less than the average relative errors in the velocity channels. A slight increase in the relative error was observed for the angular velocity channel, as compared to Model 1. Overall, the reduction of the training data dimensions improved results.

In order to evaluate the performance of Model 2 with respect to different free-stream velocities, the minimum, maximum, and average relative errors are also calculated across the samples from each individual testing case, as well as the standard deviation of the relative error.

The relative errors with respect to the free-stream velocities of 1.5, 2, and 2.5 m/s are provided in Table 12, Table 13, and Table 14, respectively.

Table 12: Relative Error of Each Channel, for the Testing Case of v_∞ = 1.5 m/s (Model 2)

	Relative error			
Channel variable	**Minimum [%]**	**Maximum [%]**	**Average [%]**	**Standard deviation**
x-Velocity, v_x	13.43	26.66	25.76	0.0207
y-Velocity, v_y	13.01	24.85	23.71	0.0185
Pressure, p	15.15	25.05	21.51	0.0200
Turbulent viscosity, v_T	7.27	10.48	9.44	0.00485
Angular velocity, ω	1.32	1.94	1.65	0.00190

Table 13: Relative Error of Each Channel, for the Testing Case of v_∞ = 2 m/s (Model 2)

	Relative error			
Channel variable	**Minimum [%]**	**Maximum [%]**	**Average [%]**	**Standard deviation**
x-Velocity, v_x	20.16	21.05	20.56	0.00235
y-Velocity, v_y	17.05	18.50	17.72	0.00369
Pressure, p	15.64	23.95	17.97	0.0186
Turbulent viscosity, v_T	8.56	9.71	9.34	0.00310
Angular velocity, ω	0.32	0.96	0.65	0.00196

Table 14: Relative Error of Each Channel, for the Testing Case of v_∞ = 2.5 m/s (Model 2)

Channel variable	Relative error			
	Minimum [%]	**Maximum [%]**	**Average [%]**	**Standard deviation**
x-Velocity, v_x	13.30	20.18	13.70	0.0109
y-Velocity, v_y	12.77	17.22	13.60	0.00687
Pressure, p	10.43	19.60	13.33	0.0179
Turbulent viscosity, v_T	6.98	9.57	7.53	0.00428
Angular velocity, ω	0.61	1.97	1.69	0.00259

As shown in the above results, the average relative error was lowest in all flow gradient channels for the free-stream velocity of 2.5 m/s. In Model 1, the average relative errors in the channels of x-velocity, y-velocity, and turbulent viscosity were also found to be lowest at the highest free-stream velocity. This finding suggests that network learning is enhanced by the stronger gradients produced by larger free-stream velocities.

The pre-computed ground truths and network predictions for a free-stream velocity of 1.5 m/s are juxtaposed in Figure 42, for two different rotor angles. The small square region of discontinuity that was produced by Model 1 in the x-velocity and turbulent viscosity channels is not present.

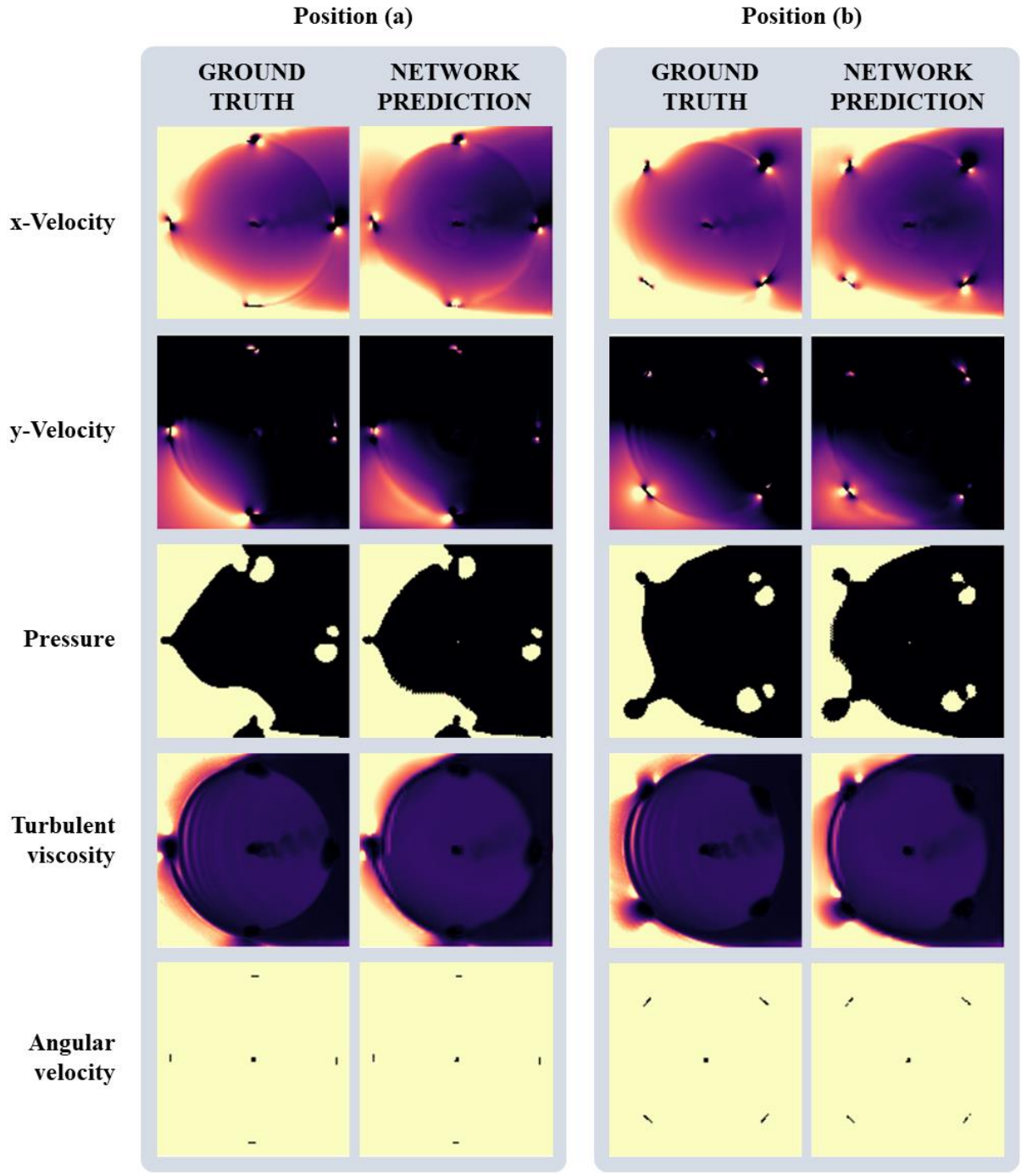


Figure 42: Ground truth and network predictions at two rotor positions, for v_∞ = 1.5 m/s (Model 2)

The pre-computed ground truths and network predictions for a free-stream velocity of 2 m/s are juxtaposed in Figure 43, for two different rotor angles. Slight discrepancies can be

observed in the wake of the rotor shaft, and in the smoothness of the pressure gradient. The rippled wake pattern generated by the foil tips, which can be observed in the y-velocity and turbulent viscosity ground truths, is only partially replicated in the predictions.

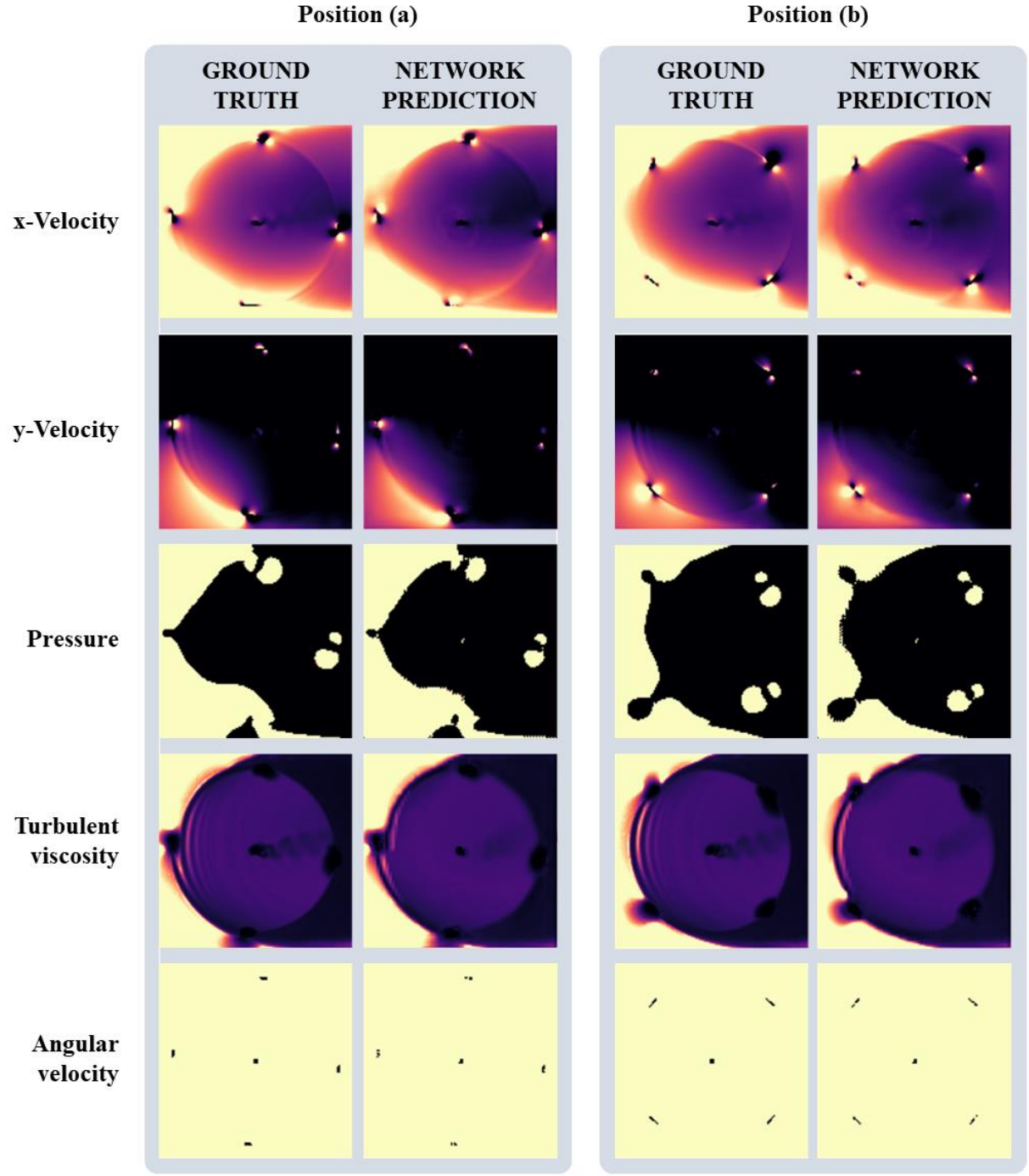


Figure 43: Ground truth and network predictions at two rotor positions, for v_∞ = 2 m/s (Model 2)

The pre-computed ground truths and network predictions for a free-stream velocity of 2.5 m/s are juxtaposed in Figure 44, for two different rotor angles.

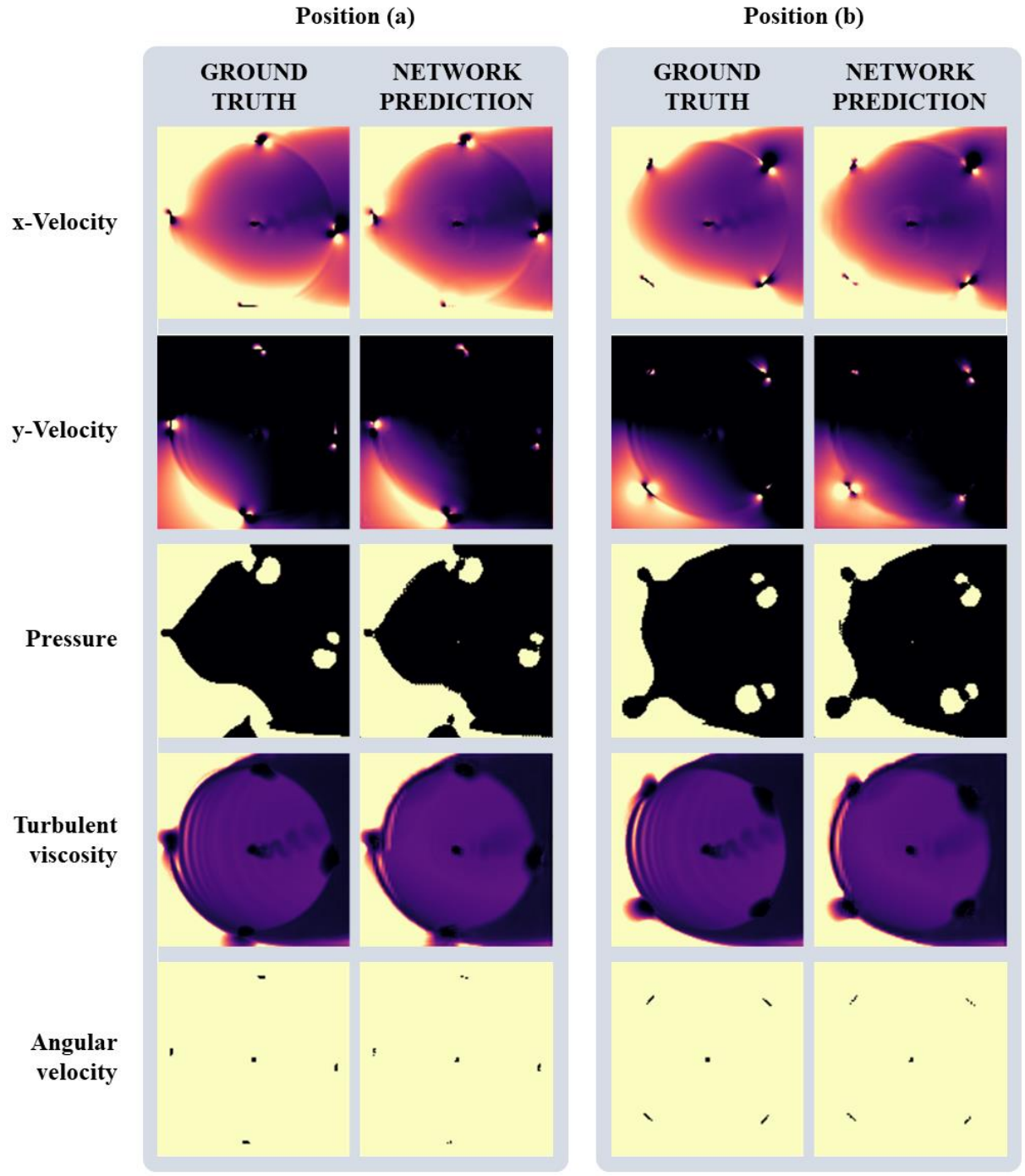


Figure 44: Ground truth and network predictions at two rotor positions, for v_∞ = 2.5 m/s (Model 2)

4.4. Model 3: Small Architecture with Three Training Cases

Model 3 was developed using the small CNN architecture depicted in Figure 23, and was trained with simulation results for the free-stream velocities of 1, 1.5 and 3 m/s. The network was trained using 415 samples, each containing an array with dimensions of 10 × 128 × 128. Training was performed over 1500 epochs, with a learning rate of 1×10^{-5}, a dropout threshold of 0.01, and a batch size of two. The training and validation losses during the training process are shown in Figure 45. Over the course of training, the validation loss stabilized around 0.001. Training took roughly three hours using an NVIDIA GeForce RTX 3060 GPU.

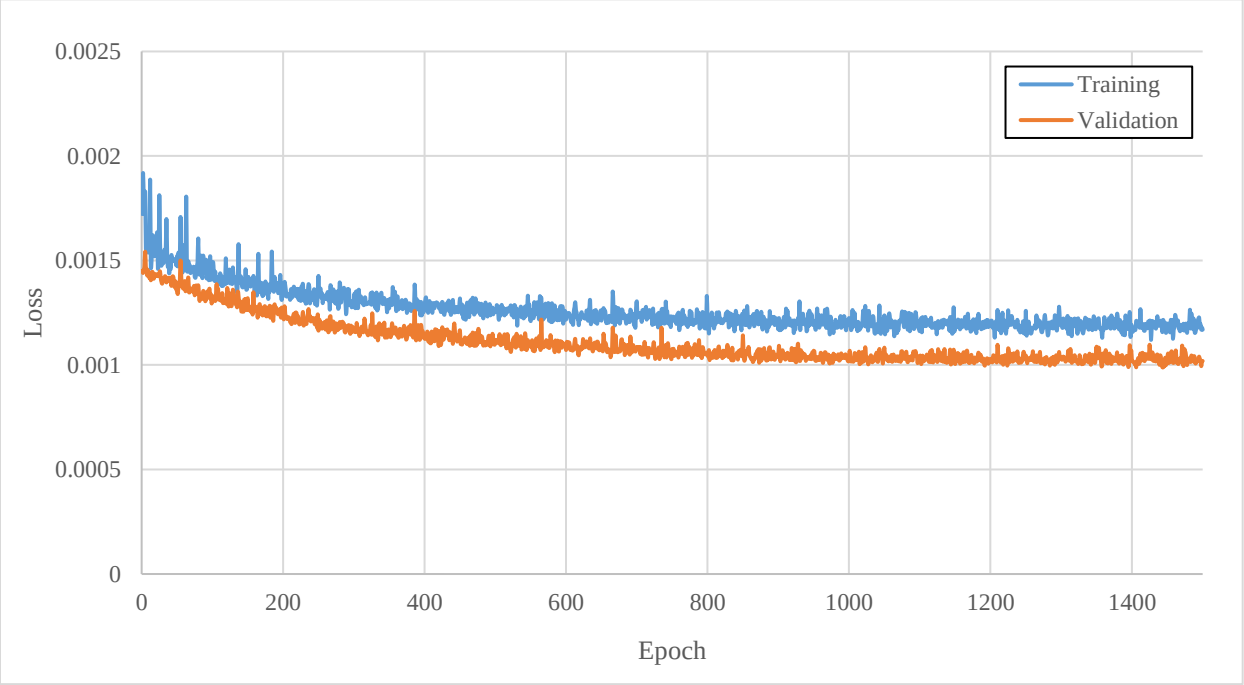


Figure 45: Training and validation losses over 1500 epochs (Model 3)

Model 3 was tested using simulation results for velocities of 2 and 2.5 m/s. Thirty-eight testing samples were selected for each free-stream velocity, to obtain a total of 76 testing samples. The minimum, maximum, and average relative errors calculated across all testing samples are shown in Table 15, as well as the standard deviation of the relative error.

Table 15: Relative Error of Each Channel, for the Combined Testing Cases of v_∞ = 2 and 2.5 m/s (Model 3)

Channel variable	Relative error			
	Minimum [%]	**Maximum [%]**	**Average [%]**	**Standard deviation**
x-Velocity, v_x	5.02	6.69	5.97	0.00579
y-Velocity, v_y	7.47	10.62	8.98	0.0106
Pressure, p	10.59	21.21	12.68	0.0158
Turbulent viscosity, v_T	6.56	13.12	7.37	0.00767
Angular velocity, ω	0.33	1.47	0.88	0.00383

As demonstrated by the relative errors shown in Table 7, Table 11, and Table 15, Model 3 achieved significantly better results than both previous models, with the highest average relative error occurring in the pressure channel. In both other models, the average relative error in the pressure channel was less than that of the x-velocity and y-velocity channels.

In order to evaluate the performance of Model 3 with respect to different free-stream velocities, the minimum, maximum, and average relative errors are also calculated across the samples from each individual testing case, in addition to the standard deviation of the relative error. The relative errors with respect to the free-stream velocities of 2 and 2.5 m/s are provided in Table 16 and Table 17, respectively.

Table 16: Relative Error of Each Channel, for the Testing Case of v_∞ = 2 m/s (Model 3)

	Relative error			
Channel variable	**Minimum [%]**	**Maximum [%]**	**Average [%]**	**Standard deviation**
x-Velocity, v_x	5.02	5.92	5.43	0.00205
y-Velocity, v_y	7.47	8.61	8.01	0.00302
Pressure, p	10.59	21.21	12.11	0.0180
Turbulent viscosity, v_T	6.74	13.12	7.61	0.00984
Angular velocity, ω	0.33	0.82	0.55	0.00159

Table 17: Relative Error of Each Channel, for the Testing Case of v_∞ = 2.5 m/s (Model 3)

	Relative error			
Channel variable	**Minimum [%]**	**Maximum [%]**	**Average [%]**	**Standard deviation**
x-Velocity, v_x	5.59	6.69	6.51	0.00191
y-Velocity, v_y	8.04	10.62	9.96	0.00462
Pressure, p	12.01	17.77	13.25	0.0108
Turbulent viscosity, v_T	6.56	8.17	7.14	0.00331
Angular velocity, ω	0.54	1.47	1.22	0.00205

As shown in the above results, comparable relative errors were achieved for both free-stream velocities. The pre-computed ground truths and network predictions for a free-stream

velocity of 2 m/s are juxtaposed in Figure 46, for two different rotor angles. Vortex shedding behind the rotor shaft is again not well predicted, and some roughness is still present in the predicted pressure gradient. The rippled wake pattern generated by the foil tips, which is visible in the y-velocity and turbulent viscosity channels, is again only partially inferred by the network.

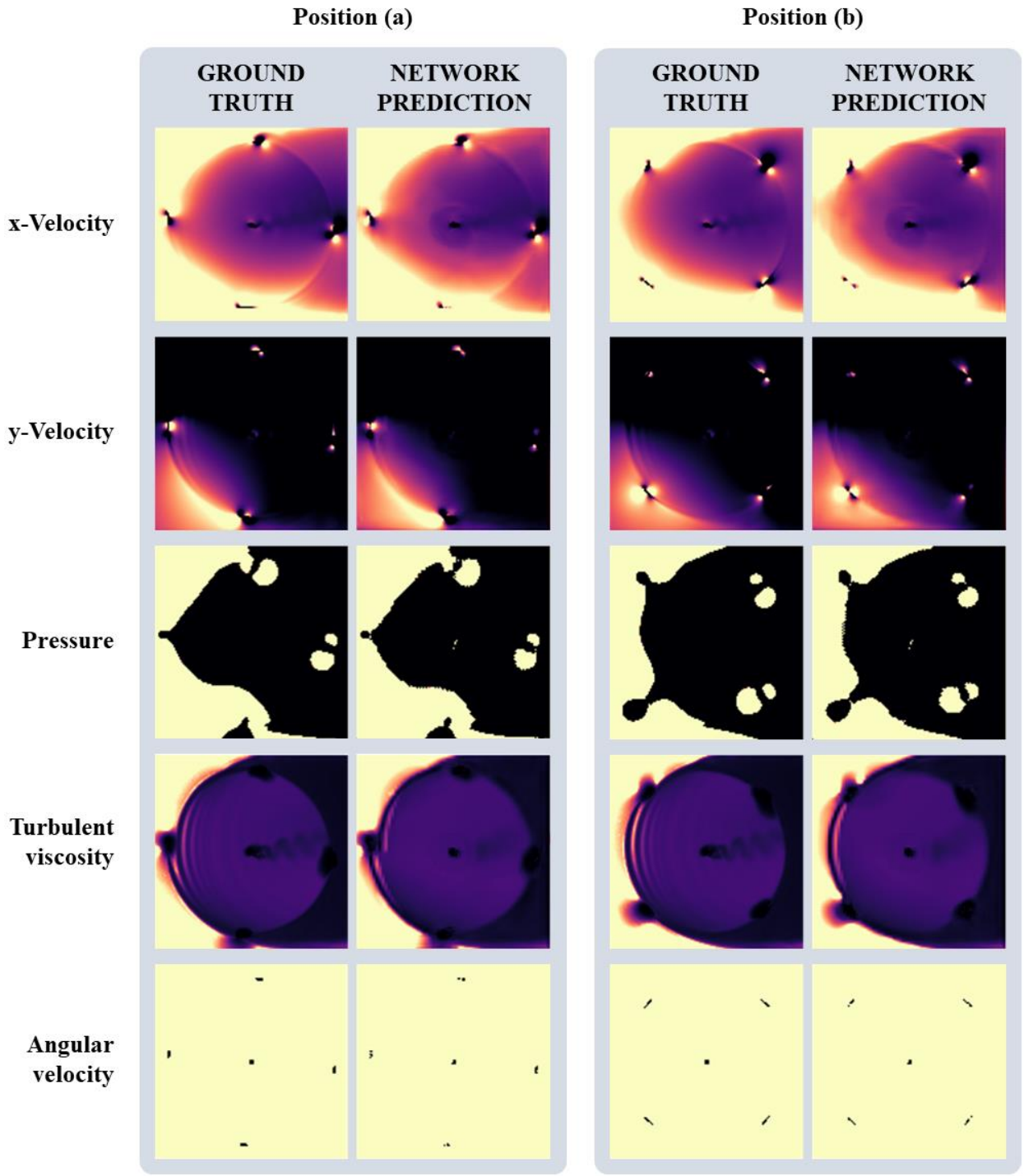


Figure 46: Ground truth and network predictions at two rotor positions, for v_∞ = 2 m/s (Model 3)

The pre-computed ground truths and network predictions for a free-stream velocity of 2 m/s are juxtaposed in Figure 47, for two different rotor angles.

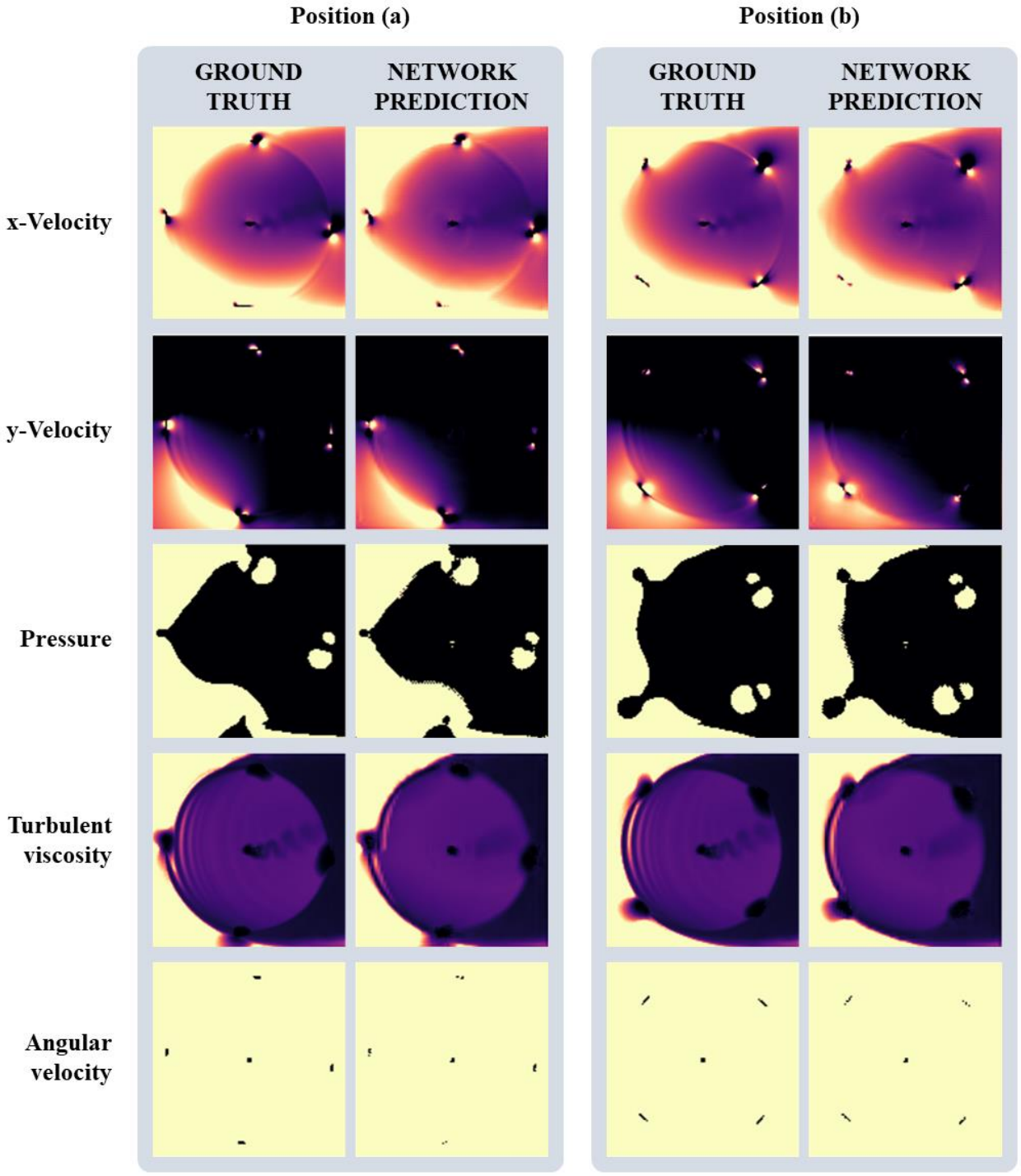


Figure 47: Ground truth and network predictions at two rotor positions, for v_∞ = 2.5 m/s (Model 3)

5. Conclusions and Recommendations

The chief objective of this research was to investigate the potential of a deep learning-based turbine modelling method, in hopes of laying a foundation for an alterative approach to optimizing turbine array layouts. Three CNNs were trained to predict the solutions of a transient, 2-D RANS model, for a vertical-axis hydrokinetic turbine operating in free-stream velocities in the range of 1 and 3 m/s. For the boundary conditions of free-stream velocity and rotor position, the flow gradients of x-velocity, y-velocity, pressure, and turbulent viscosity were predicted, as well as the angular velocity of the rotor. Training and testing data for the CNN were derived from the solutions of five RANS simulations, with free-stream velocities of 1, 1.5, 2, 2.5 and 3 m/s. To investigate the effect of training data dimensions on prediction accuracy, two different CNN architectures were developed, for data sizes of $10 \times 1024 \times 1024$ and $10 \times 128 \times 128$. The smaller data size was found to produce significantly smaller error in the x-velocity, y-velocity, and turbulent viscosity channels. In order to assess how the diversity of the training data would impact the learning of the CNN, models were also developed using different combinations of simulations as training cases. Specifically, one dataset was based on the free-stream velocities of 1 and 3 m/s, while the other was based on the free-stream velocities of 1, 1.5, and 3 m/s. Using three training cases was found to produce dramatically lower errors in all flow gradient channels. The angular velocity of the turbine was predicted relatively uniformly by all three evaluated models. For the best achieved CNN model, the unknowns of x-velocity, y-velocity, pressure, turbulent viscosity, and rotational velocity were inferred with mean relative errors of 5.97%, 8.98%, 12.68%, 7.37%, and 0.88%, respectively.

For future works, it is recommended that the effects of smaller data dimensions be further investigated. This issue would be particularly relevant for the inference of solutions over larger

flow domains with multiple turbines. In this study, simulation exports were interpolated onto a grid size of 1024 × 1024 so that the foils would be well defined, and data resizing was encoded in the training and testing codes as required. From a practical perspective, it is recommended that unnecessarily high-resolution data not be generated from the exports of RANS simulations. In this project, the computational cost of processing the simulation exports was significant, taking several days in total. Moreover, the generated datasets were cumbersome to store and transfer.

In order to further investigate the viability of deep learning as a tool for turbine array optimization, the methodology developed in this study should be adapted to 3-D, dual-turbine simulations with MPPT. To minimize computational cost, it is recommended that 3-D simulations be performed using a foil span of two chord lengths. For a dual-turbine model, the unknowns predicted by the CNN could include the combined instantaneous power output of the two turbines, and the field gradients of x-velocity, y-velocity, pressure, and turbulent viscosity. To minimize the complexity of the learning task for the CNN, it is recommended that the CNN be trained using two-dimensional field gradients in the x-y plane, at the mid-height of the fluid domain.

Lastly, for future works employing more rigorous CFD models, it is recommended that the turbine simulations be validated prior to being used to generate training and testing data for the CNN. Since turbine interaction models require that both the power output and wake field of the turbines be accurately rendered, both aspects of the simulations should be substantiated. While this could be achieved through field experiments, a more efficient and accessible approach would be to base the study on a turbine model for which validated solutions are readily available. Since validated solutions for dual-turbine models are not easily procured, the most practical course of action may be to validate a 3-D, single-turbine model with MPPT, prior to developing a dual-turbine model of equivalent quality.

References

Bachant, P., Goude, A., & Wosnik, M. (2016). Actuator line modeling of vertical-axis turbines. *arXiv preprint*.

Bjorck, N., Gomes, C. P., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. *Advances in neural information processing systems 31*, 7694–7705.

Cheng, Z., Madsen, H. A., Gao, Z., & Moan, T. (2016). Aerodynamic modeling of floating vertical axis wind turbines using the actuator cylinder flow method. *Energy Procedia*, 94, 531-543.

Cunningham, P., Cord, M., & Delany, S. J. (2008). *Supervised learning. In Machine learning techniques for multimedia (pp. 21-49).* Berlin, Heidelberg: Springer.

d'Auteuil, S., Birjandi, A., Bibeau, E., Jordan, S., Soviak, J., & Friesen, D. (2019). Riverine hydrokinetic resource assessment using low cost winter imagery. *Renewable and Sustainable Energy Reviews*, 105, 293-300.

De Tavernier, D., & Ferreira, C. (2019). An extended actuator cylinder model: Actuator-in-actuator cylinder (AC-squared) model. *Wind Energy*, 22(8), 1058-1070.

Dhanak, M. R., Duerr, A. E., & VanZwieten, J. H. (2016). Marine hydrokinetic energy resource assessment. In *Springer Handbook of Ocean Engineering* (pp. 1099-1116). Springer, Cham.

Ferreira, C. S., Madsen, H. A., Barone, M., Roscher, B., Deglaire, P., & Arduin, I. (2014). Comparison of aerodynamic models for vertical axis wind turbines. *Journal of Physics: Conference Series (Vol. 524, No. 1, p. 012125)*.

Haywood, A. M., Hill, D. J., Dolan, A. M., Otto-Bliesner, B. L., Bragg, F., Chan, W.-L., Chandler, M. A., Contoux, C., Dowsett, H. J., Jost, A., Kamae, Y., Lohmann, G., Lunt, D. J., Abe-Ouchi, A., Pickering, S. J., Ramstein, G., Rosenbloom, N. A., Salzmann, U., Sohl, L., ... Zhang, Z. (2013). Large-scale features of Pliocene climate: results from the Pliocene Model Intercomparison Project. *Climate of the Past*, 9(1), 191-209.

Haywood, A. M., Dowsett, H. J., & Dolan, A. M. (2016). Integrating geological archives and climate models for the mid-Pliocene warm period. *Nature communications*, 7(1), 1-14.

International Energy Agency. (2021, December 1). *Renewable electricity growth is accelerating faster than ever worldwide, supporting the emergence of the new global energy economy*. Retrieved from https://www.iea.org/news/renewable-electricity-growth-is-accelerating-faster-than-ever-worldwide-supporting-the-emergence-of-the-new-global-energy-economy

Karimi, E., & Kazerani, M. (2017). Impact of renewable energy deployment in Canada's remote communities on diesel generation carbon footprint reduction. *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, (pp. 1-5). Windsor, ON, Canada.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lovekin, D., Moorhouse, J., Morales, V., & Salek, B. (2020). *Diesel reduction progress in remote communities: Research summary* . Pembina Institute.

Marsh, P., Ranmuthugala, D., Penesis, I., & Thomas, G. (2017). The influence of turbulence model and two and three-dimensional domain selection on the simulated performance characteristics of vertical axis tidal turbines. *Renewable energy*, 105, 106-116.

Mikkelsen, R. (2003). Actuator disc methods applied to wind turbines (Doctoral dissertation, PhD thesis, Technical University of Denmark).

Moss, R. H., Babiker, M., Brinkman, S., Calvo, E., Carter, T., Edmonds, J. A., Elgizouli, I., Emori, S., Lin, E., Hibbard, K., Jones, R., Kainuma, M., Kelleher, J., Lamarque, J. F., Manning, M., Matthews, B., Meehl, J., Meyer, L., Mitchell, J., ... Zurek, M. (2008). Towards new scenarios for analysis of emissions, climate change, impacts, and response strategies (No. PNNL-SA-63186). *Intergovernmental Panel on Climate Change.* Richland, WA (United States): Pacific Northwest National Lab.(PNNL).

National Oceanic and Atmospehric Administration. (2021, December 14). *NOAA Research's top 5 stories from 2021*. Retrieved from NOAA Research News: https://research.noaa.gov/article/ArtMID/587/ArticleID/2816/NOAA-Researchs-top-5-stories-from-2021#:~:text=In%20May%2C%20carbon%20dioxide%20measured,Scripps%20Institution%20of%20Oceanography%20scientists.

Ning, A. (2016). Actuator cylinder theory for multiple vertical axis wind turbines. *Wind Energy Science*, 1(2), 327-340.

Obiols-Sales, O., Vishnu, A., Malaya, N., & Chandramowliswharan, A. (2020). CFDNet: A deep learning-based accelerator for fluid simulations. *Proceedings of the 34th ACM international conference on supercomputing*, (pp. 1-12).

Paraschivoiu, I. (1988). Double-multiple streamtube model for studying vertical-axis wind turbines. *Journal of propulsion and power*, 4(4), 370-377.

Rezaeiha, A., Kalkman, I., & Blocken, B. (2017). CFD simulation of a vertical axis wind turbine operating at a moderate tip speed ratio: Guidelines for minimum domain size and azimuthal increment. *Renewable energy*, 107, 373-385.

Ridgill, M., Neill, S. P., Lewis, M. J., Robins, P. E., & Patil, S. D. (2021). Global riverine theoretical hydrokinetic resource assessment. *Renewable Energy*, 174, 654-665.

Schwalm, C. R., Glendon, S., & Duffy, P. B. (2020). RCP8. 5 tracks cumulative CO2 emissions. *Proceedings of the National Academy of Sciences*, 117(33), 19656-19657.

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *towards data science*, 6(12), 310-316.

Statista. (2021). *Territorial carbon dioxide (CO2) emissions in Canada from fossil fuel combustion and industrial processes from 1960 to 2020*. Retrieved from https://www.statista.com/statistics/209619/canadian-co2-emissions/#statisticContainer

Tans, P., & Thoning, K. (2018, March). *How we measure background CO2 levels on Mauna Loa.* Retrieved from NOAA Earth System Research Laboratory Global Monitoring Division: https://gml.noaa.gov/ccgg/about/co2_measurements.pdf

Thuerey, N., Weißenow, K., Prantl, L., & Hu, X. (2020). Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows. *AIAA Journal*, 58(1), 25-36.