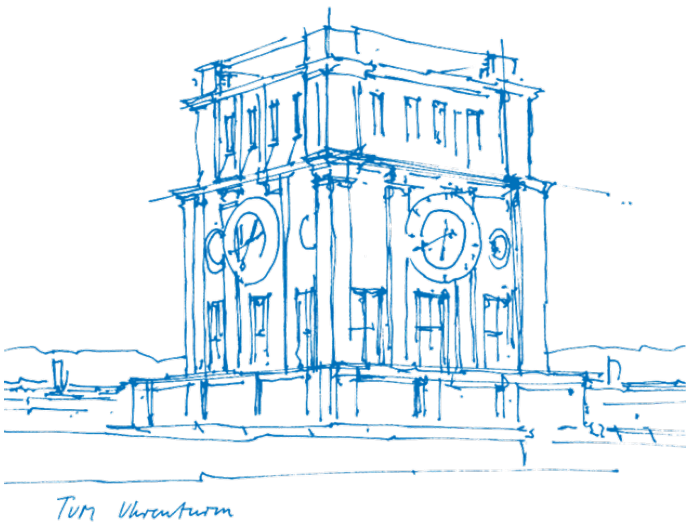
Technische Univeristät München
TUM School of Engineering and Design

Predicting Urban-Scale Wind Simulations -

Artificial Intelligence for Simulation Engineering in Urban Planning

Ivan Bratoev



Technische Univeristät München
TUM School of Engineering and Design

Predicting Urban-Scale Wind Simulations -

Artificial Intelligence for Simulation Engineering in Urban Planning

Ivan Bratoev

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:
Prof. Dr. Pierluigi D'Acunto

Prüfer*innen der Dissertation:
1. Prof. Dr.-Ing. Frank Petzold
2. Prof. Dr. Nils Thuerey

Die Dissertation wurde am 14.08.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 11.10.2023 angenommen.

Abstract

Wind and fluids, in a broader context, are a significant validation aspect in urban planning, civil engineering and architecture. Wind can have a wide range of varying effects on the local environment. Introducing obstructions to existing wind paths and weather patterns may have serious ramifications on the urban landscape - from reduced air quality or pedestrian wind comfort, to the creation of urban heat islands or even cause structural damage. The introduction of tools capable of providing decision support with regard to wind simulations in the early stages of urban planning can provide crucial information to the planner. State-of-the-art solutions currently utilized for wind simulations are predominantly integrated as validation steps in the later stages of design. This is due to the complex nature of simulating and modelling wind and fluids.

Through the advancement of Artificial Intelligence (AI) breakthroughs in the approximation of such simulations has been achieved on a theoretical basis. The goal of this dissertation is to explore the applicability of such solutions on a larger urban scale, that have complex environments and then expand upon them.

A significant challenge that is addressed within this dissertation is the difficulty with which data is created and prepared for use in training AI. Due to the wide range of scenarios and environments that exist for wind simulations in urban planning, there does not exist a singular solution in representing and modelling them. A flexible pipeline is proposed and prototypically implemented. This allows for the creation of data sets, tailored to the specific requirements of the planner. These conditions and simulation parameters are abstracted to a simplified set, extracted through a detailed literature overview and analysis of current state-of-the-art user studies conducted in the field of Computational Wind Engineering (CWE).

Analyzing the current advancements in the field of AI and specifically Deep Learning (DL), a solution, based on Diffusion Models, is proposed. Expanding upon the existing solutions, a novel approach that integrates the geometry and temporal aspect of wind simulations is presented. The concept is prototypically implemented and validated. Based upon the results of this analysis further steps for the improvement of the concept and prototype are outlined.

Zusammenfassung

Wind und Fluide in einem allgemeineren Aspekt sind ein wichtiger Validierungsaspekt in der Stadtplanung, im Bauwesen und in der Architektur. Wind kann eine breite Reihe von unterschiedlichen Auswirkungen auf die lokale Umgebung haben. Die Beeinträchtigung bestehender Windwege und Wetterverhältnisse kann schwerwiegende Auswirkungen auf das Stadtbild haben - von einer verminderten Luftqualität oder einem geringeren Windkomfort für Fußgänger bis hin zur Entstehung städtischer Wärmeinseln oder sogar zu Bauschäden. Die Einführung von Werkzeugen, die in der Lage sind, in den frühen Phasen der Stadtplanung Entscheidungsunterstützung in Bezug auf Windsimulationen zu gewährleisten, kann dem Planer grundlegende Informationen liefern. Moderne Lösungen, die derzeit für Windsimulationen eingesetzt werden, werden überwiegend als Validierungsschritte in späteren Phasen der Planung integriert. Grund dafür ist die komplexe Natur der Simulation und Modellierung von Wind und Fluiden.

Durch die Weiterentwicklung von KI wurden auf theoretischer Basis Durchbrüche bei der Annäherung an solche Simulationen erzielt. Ziel dieser Dissertation ist es, die Anwendbarkeit solcher Lösungen in größeren Stadtgebieten mit komplexen Umgebungen zu erforschen und sie anschließend zu verfeinern.

Eine große Herausforderung, die im Rahmen dieser Dissertation behandelt wird, ist die Schwierigkeit, Daten für das Training einer KI zu erstellen und aufzubereiten. Aufgrund des breiten Spektrums an Szenarien und Umgebungen, die für Windsimulationen in der Stadtplanung existieren, gibt es keine einheitliche Lösung für deren Darstellung und Modellierung. Eine flexible Pipeline wird entworfen und prototypisch umgesetzt. Damit können Datensätze erstellt werden, die auf die spezifischen Anforderungen des Planers ausgerichtet sind. Diese Bedingungen und Simulationsparameter werden zu einem abstrahierten Satz zusammengefasst, der durch einen detaillierten Literaturüberblick und eine Analyse aktueller, dem "State of the Art" entsprechender Nutzerstudien auf dem Gebiet der CWE ermittelt wurde.

Durch die Analyse der aktuellen Fortschritte im Bereich der KI und insbesondere der DL wird eine auf Diffusionsmodellen basierende Lösung entworfen. In Erweiterung der bestehenden Lösungen wird ein innovativer Konzept vorgestellt, das die geometri-

schen und zeitlichen Aspekte von Windsimulationen integriert. Das Konzept wird prototypisch implementiert und validiert. Auf der Grundlage der Ergebnisse dieser Analyse werden weitere Schritte zur Verbesserung des Konzepts und des Prototyps vorgestellt.

Acknowledgment

This dissertation was created during my work as a research associate at the Chair of Architectural Informatics at the Technical University of Munich.

A wide group of people supported me during this process, and I would like to express here my sincerest gratitude.

First I want to thank my partner, Brittany Engle, for providing me an unending support in all aspects of my life. Without her positive energy and encouragement I would have been lost. I also want to thank my family, for providing me with support, advice and constant motivation to push forward.

I would like to especially thank Prof. Dr. -Ing Frank Petzold and Dr.-Ing Gerhard Schubert. I could not have wished for more dedicated and supporting Supervisor and Mentor. Your insights have helped me become a better researcher. I would like to also thank you for the personal support and constant motivation and energy that you provided me with. Finally, I am grateful for the provided freedom to explore my ideas.

I would also like to thank my colleagues. Without the constant conversation with Alejandro Rueda, I would not have come as far as I did with my technical concepts. Your knowledge provided invaluable insights into my research. I would also like to thank Nick Förster and Ilayda Memiş as well. You have provided me with the time and support I need, to focus on this work when it was necessary, especially in the final stages.

Finally, I would like to express my thanks to all the students that participated in different projects and lectures, or worked as assistants at the Chair of Architectural Informatics. Without your assistance the prototypes would not have been implemented.

Contents

1 Introduction

This dissertation focuses on the exploration and application of AI methods to improve upon the existing limitations of urban scale simulations, such as long computation times, focus on singular models/designs and the inability to handle imprecise or vague data. This will provide additional decision support for the planners and architects in the early stages of design. Numerical simulations used in urban planning are computationally intense, with vast arrays of parameters that direct a multitude of components within the simulation. The long, typically, non-interactive simulation times, the large requirements on information and complex configurations negatively disrupt the typical design process in these early stages of urban planning (Østergård et al., 2016). AI methods excel in inferring hidden dependencies between various parameters of the data from which they learn. This coupled with their lower memory overhead and faster evaluation times make AI an ideal area for the exploration of potential new improvements to existing simulation methods and algorithms, that are currently used in urban planning. Due to the vast amount of varied numerical simulation types and the different models they utilize, this dissertation will focus on the application of Computational Fluid Dynamics (CFD) on an urban planning scale, focusing specifically on wind simulations in the context of urban planning in the early design stages.

This Architectural Informatics dissertation explores topics in the fields of Architecture and Urban Planning, Civil Engineering, CWE, and Computer Sciences and Numerical Simulations. To better the understanding of the underlying problems within these fields and their relationship to one another, a brief overview of the current established principals will be presented. A critical observation of the issues preventing the integration of wind simulations in early design stages of urban planning will be made. On this basis the relevant research gab, goals of the dissertation and the accompanying research questions will be defined.

1.1 Current Situation / State of the Art

The early stages of urban planning and design are complex and dynamic. Within them designs and concepts are changed rapidly and iteratively. The major design

decisions, that are made within these stages, have wide-reaching effects on the final outcome (Steinmann, 2004). Although there are established simulation tools capable of evaluating proposed designs, they are most often used in the later stages in order to validate a final design. This is due to the high, non-interactive computational times of such simulation tools. Furthermore, they require a vast array of concrete details about the design that are not readily available in the beginning. If an issue with the final design is detected the cost to address it is significantly higher than had it been addressed earlier (Macleamy, 2004). The advantages of ad-hoc results for the early stages of design have been explored and analyzed over the past years and highlight the necessity for Design Decision Support (DDS) within the early stages (Förster et al., 2021; Bratoev et al., 2018, 2017; Gabriel et al., 2021).

Due to the vast complexity of urban models and the complex relation between the different components of these models, the main focus of research and development has been in the creation of more detailed simulation models, capable of integrating multiple data models in one simulation (Sola et al., 2020; Widl et al., 2018). Some fields of simulation such as CWE, that relay heavily on research in the field of CFD, have proposed various simulation models that incorporate different complex models (Toparlar et al., 2017a). The field has become a central research point for urban simulations due to the important role they play in pedestrian comfort (Blocken et al., 2012), urban heat islands predictions and modelling (Aghamolaei et al., 2021), pollutant dispersal scenarios (Tominaga and Stathopoulos, 2013), and many more (KC et al., 2019; Ishugah et al., 2014). The focus of the research has also started incorporating performance, exploring potential alternative methods (Schubiger et al., 2020), different hardware architecture and uses (Obrecht et al., 2014). The results offer significant runtime improvements already, but still require hours of computational time. As can be seen from the work done in this field, CFD and CWE play crucial roles in urban planning validation scenarios, so much so that there have been country wide validation scenarios and guidelines for their use (Tominaga et al., 2008; Franke et al., 2007; NEN, 2006). The improvement of their performance, precision and ease of use and their wide applicability in validation scenarios make them ideal test candidates for the goals of this dissertation.

1.2 Critical Observation

As can be seen from the state-of-the-art overview, wind simulations play a crucial role in a wide range of simulation and validation scenarios utilized in urban planning. Such tools can then clearly be employed in the early design stages, where they can

greatly assist the planners by providing crucial insights into their designs. In spite of the achieved advances in the field of CFD and CWE (with regard to both simulation models, performance and flexibility) there are no optimal approaches in integrating them into the early design stages of urban planning. Drawing upon criteria outlined in Schubert (2021) for the successful implementation of tools in these stages several requirements can be defined: precision and reliability, flexibility and adaptability, user-friendliness, interoperability, and responsiveness. In the following section these requirements will be clearly defined. Furthermore, the capability of current CFD solutions to fulfill these requirements will be briefly discussed.

Reliability of a simulation tool is connected to its capability of providing accurate enough results in comparison to established validation scenarios. The reliability of wind simulations on urban scales have been a large focus of research, with various guidelines and user studies focusing on exact domain modelling criteria for such problems. Due to the high complexity of such tools, tailoring all the requirements towards specific scenarios and urban layouts requires a deep understanding of the underlying numerical models governing the simulations. In spite of such guidelines, it is still considered that scaled physical models of the environment, integrated into wind tunnels, provide more reliable results (Li et al., 2018).

Flexibility and Adaptability are crucial requirements for the integration of any tool into the early design stages. These requirements describe the capability for the tool to solve a wide range of scenarios, while also being able to handle different levels of data precision. Wind simulation tools need to provide a way for the planner to easily modify either the domain conditions, the layout and geometry of the urban area or other factors, such as wind speed and direction. Although current state-of-the-art solutions can solve a wide range of scenarios, they focus on singular simulation executions. The singular nature of such simulations is due to the high computational times and large memory requirements associated with such models (Morozova et al., 2020; Jóczik et al., 2022; Houzeaux et al., 2022). Changes to such simulations, such as geometry changes, or boundary conditions, e.g, wind direction, or wind speed, require significant effort, due to the complexity of the underlying numerical solutions.

User-friendliness provides a level of abstraction between the underling numerical tool and the planner, through the use of Graphical User Interface (GUI), or other input options (Schubert, 2021). Wind simulations are based on complex mathematical, physics-based models and are implemented through numerical techniques, making them challenging to be represented through user-friendly interfaces. Wind simulations, being an extension of CFD, focus on solving the highly non-linear Navier-Stokes equations. These equations have various numerical techniques that they

can be solved with, each consisting of a vast array of variables and equations that greatly influence the result. Furthermore, such simulations require precise modelling of boundary conditions, to represent the topography and weather patterns associated with the domain. All this leads to the necessity of tailored software solutions and a deep understanding for the field of CFD, thus making it difficult to communicate the potential implications of changes made to the models by the user.

Interoperability describes the capability of software tools to be easily integrated into other tools. A crucial part of the early design stages is the use of a wide spectrum of digital and analog tools within the planning process. With advancements such as the Collaborative Design Platform (CDP) (Schubert, 2021), it has become easier to blend both sides into a unified approach. This has led to the necessity that DDS (such as simulations) have the capability to be integrated into other processes. This requirement is in stark contrast to established software solutions for wind simulations, that have rigid predefined input and output structures, interfaces, and execution scenarios.

Responsive tools provide the user with results directly, regardless of the complexity of the task. Furthermore, the early design stages of urban planning are characterized by an iterative dynamic design process. For a tool to be successfully integrated into these stages, it must not only provide results directly, but it must also not interrupt the design process itself. (Bratoev et al., 2018, 2017; Förster et al., 2021) have shown, that the results of such complex tools integrated into the early design stages do not have to deliver the final results directly, but that intermediate results can also be utilized to inform the planner sufficiently. As discussed in the flexibility and adaptability requirements, wind simulations have long computational times, where the extraction of intermediate results is also not always possible ad-hoc.

From this overview it is clear, that CFD simulations, and more specifically urban scale wind simulations, are not well suited for the successful integration into early design stages processes. The introduction of alternative ways of providing the planner with such simulation results is required. AI has shown to have the capability to bridge some of these issues. Such methods offer a significant improvement in performance and resource requirements (Xie et al., 2018; Zhong et al., 2022), enabling for ad-hoc results. Although such approaches have been shown to not possess as high of a precision and reliability as established approaches, their accuracy has been established as sufficient for early stage planning, where the vagueness of models would impact the precision of traditional approaches (Calzolari and Liu, 2021). In spite of the large amount of data required to train AI methods, the resulting models can be easily reused in further applications (Mohan and M.V.S.S, 2022). AI methods infer relations and dependencies between the various parameters in the data they are trained

on. This allows for the simplification of parameters that a user has to control, allowing for more user-friendly interactions with the simulation tool. The challenge remains to have a flexible enough input and output structure of such models, that it allows for a wider range of flexibility for its integration.

As can be seen from this brief overview into the current state-of-the-art solutions in the area of CFD and CWE, there are certain discrepancies between the requirements necessary for their integration in the early stages of urban design. On the other hand, AI methods offer ways to circumvent some of these challenges. Existing solutions applying such approaches in combination with CFD solutions have shown on a smaller scale, that such concepts are feasible (Raissi et al., 2019; Kim et al., 2019; Obiols-Sales et al., 2020). Thus, an approach that expands upon existing solutions, in scale, complexity, and technology holds the potential to enable the integration of wind simulations into the early design stages of planning while fulfilling all the requirements of such tools.

1.3 Goal of the Dissertation

As briefly discussed in previous sections the goal of this dissertation is to integrate CFD simulations in the early stages of design by overcoming the aforementioned limitations with the use of AI. This dissertation will focus on the hypothesis that:

Artificial Intelligence methods and algorithms in conjunction with urban scale physics based numerical wind simulations are capable of outputting more reliable than established simulation tools in the context of urban planning, while also maintaining a high level of precision.

This dissertation will focus not only on exploring the applicability of AI for urban scale physics based CFD simulations but will also research and propose an approach for the full integration of such modified simulations into the early design stages. With these goals in mind the dissertation will attempt to provide answers and insight to the following research questions:

- What kind of AI methods and algorithms are best suited for learning and predicting CFD simulation results on an urban scale?
- What are the minimal requirements for a successful integration of such an approach?
- How would an integration pipeline look like?

These further research questions will server as cornerstones for the following research, concept development, prototypical implementation and validation.

1.4 Approach

The integration of wind simulations through the utilization of AI in the early design stages of urban planning can significantly improve the urban environment by providing crucial design relevant information to the planner. This dissertation proposes the exploration of this potential by conceptualizing and developing a framework capable of seamlessly integrating CWE simulations into the early design stages of urban planning. The use of AI aims to not explicitly substitute the existing approaches in these fields, but rather provide a complementary method, focused on providing accurate predictions of the effects of the design in a way, better suited for the early stages of urban design.

With data-driven approaches, such as AI, a large emphasis is placed on the quality and diversity of the data that is trained. This can limit the applicability scope of the trained models greatly, or optimize them for specific scenarios (Sambasivan et al., 2021). As part of this dissertation an approach for the automatic generation of data will be conceptualized, prototypically implemented and validated. Through theoretical overview of the governing CFD equations and the established state-of-the-art approaches, extracted from the literature review, a data generation pipeline is proposed.

Building upon the existing advancements in the field of AI and CFD, an overview of existing solutions is presented and summarized. The advantages and disadvantages of the most promising approaches are outlined and based on the requirements for urban scale CFD simulations, a novel approach integrating various aspects of a CWE simulation is proposed and prototypically implemented. Finally, an analysis of the approach, based on the outlined requirements defined above, is performed.

1.5 Chapter Overview

As mentioned above systematic research and analysis in the core subjects of CFD and AI is required. Following this analysis and research, the current state of these two fields will be presented, and relevant adjacent fields will also be briefly discussed. Based on this information a deficit analysis is conducted. This analysis will provide an overview into how current solutions in the field of CFD and CWE are suited for the integration in early design stages. This chapter provides an overview of the suitability of existing AI solutions to bridge the requirements gap outlined in Section 1.2. Building

upon these the following chapters will present the concept, prototypical implementation and its validation. This is visually represented in Figure 1.1.

Chapter 2 Computational Fluid Dynamics: The main focus of this chapter is to establish the core definitions and mathematical and numerical principals in the field of CFD. Expanding upon these base components, the relevance of the field for urban scale wind simulations and CWE is introduced. Building upon this the requirements and guidelines for numerical simulations on an urban scale will be presented.

Chapter 3 Artificial Intelligence: The chapter will establish a clear definition between various subfields in the area of Artificial Intelligence. It covers the theoretical basis of the state-of-the-art approaches from the subfield of DL, specifically Artificial Neural Networks. Expanding upon the base knowledge, current relevant Neural Network architectures are presented.

Chapter 4 Related Works: Utilizing the definition and knowledge obtained from Chapter 2 and Chapter 3 we will present current state-of-the-art solutions utilizing either CFD algorithms, AI methods in conjunction with simulations, both on an urban scale. The chapter will also cover relevant and important advancements in relevant adjacent fields such as CWE and Numerical Simulations.

Chapter 5 Deficit Analysis: Based on the requirements and limitations of CFD and CWE algorithms and AI methods, presented in Chapter 2 and Chapter 3 and the current state-of-the-art solutions in these and adjacent fields(discussed in Chapter 4) a deficit analysis is performed. This analysis will highlight the discrepancies and issues in current approaches utilizing CFD in urban context, AI methods in urban scale simulations and existing AI integration in CFD. Based on this a set of criteria will be defined, that will serve as guidelines for a proposed concept and solution that will attempt to bridge the gap o‘f such deficiencies.

Chapter 6 Concept: Based on the deficiencies and requirements defined in Chapter 5 a concept for the integration of an AI method with urban scale CFD simulations is presented. This concept will consist of separate autonomous parts, that will contribute to the final result.

Chapter 7 Validation of Prototype: Following the evaluation criteria defined in Chapter 5 the prototypical implementation of the concept presented in Chapter 6 is evaluated and analyzed further.

Chapter 8 Discussion and Future Work: In this chapter the results of the dissertation will be summarized. A discussion about the concept and its prototypical implementation is presented and potential future applications and improvements are discussed.

Appendix Prototype: The prototype, based on the concept in Chapter 6, is presented in detail.

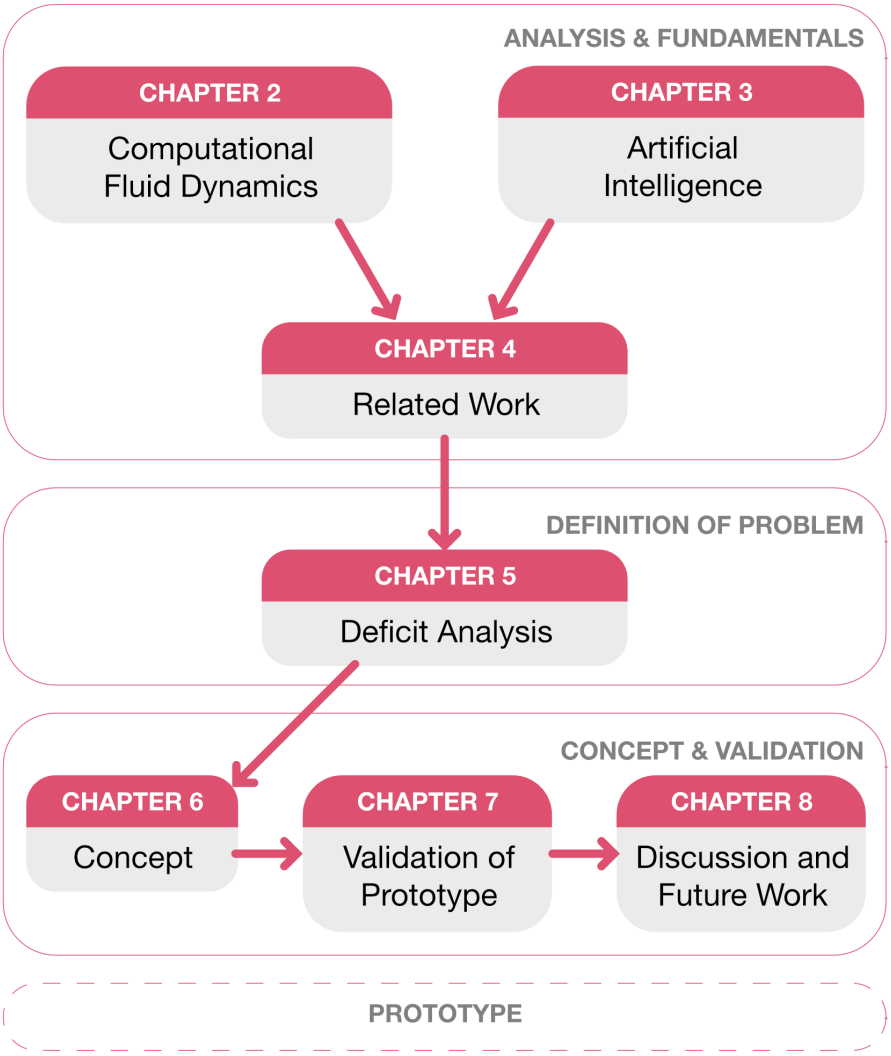


Figure 1.1 Chapter layout based on relevant topic areas

2 Computational Fluid Dynamics for Urban Wind Simulations

This chapter highlights the advantages and disadvantages of the various numerical solvers in the field of CFD. It provides an overview of the main governing physical equations and highlights current numerical approximations to the physical models. The chapter also highlights the specific construction and requirements needed to obtain accurate results for numerical wind simulations on an urban scale. This chapter will result in an overview of the established best practices for using CFD solvers for urban scale wind simulations and their requirements is the result of this chapter.

2.1 Overview of the Application of Computational Fluid Dynamics

The field of CFD covers the physical modelling and numerical simulation of all fluids. This covers a vast area of practical applications - from meteorological models and simulations, through the design of aerodynamic vehicles, to pedestrian comfort studies. Originally such problems were solved through the use of physical experimental studies - in the case of aerodynamic tests or urban scale pedestrian comfort using wind-tunnels (Cermak, 2003). Although this technique is still used to obtain realistic and precise information, it is a very time and resource consuming approach. These limitations and the rise of computing power have encouraged the adoption of numerical solvers that perform similar experiments with much lower costs (Blocken et al., 2016).

In order to describe and understand all the general scenarios and fields of application it is necessary to cover the different types of fluids, defined through their physical properties. A fluid is a substance that has a non-solid physical state, which allows for the volume to deform or react to external forces that are applied to it. The fluid has several properties which influence how the governing physical equations are defined.

Viscosity and Shear stress: Viscosity describes a fluids' resistance to deformation. It describes the internal friction of the fluid. A fluid that has zero viscosity is

known as an inviscid fluid. The general friction of the fluid, either with the domain walls, obstacles in the flow, or within itself, is defined as the shear stress (Nakayama et al., 2018).

Newtonian and Non-Newtonian Fluids: This property is based on Newton's law of viscosity. The law states that shear stress is directly proportional to the velocity gradient. Meaning, that a Newtonian fluid, one that fulfills the law, has a viscosity that does not change with the rate of flow. Non-Newtonian fluids conversely are fluids, that do not follow this law (Nakayama et al., 2018).

Multiphase Flow: A multiphase flow describes the simultaneous flow of multiple materials with different thermodynamic phases. This type of flow observes the interaction, or lack there of, between the different types of flow that intermix in the observed domain. A flow, that consists of only one type of fluid, is called a singlephase flow (Nakayama et al., 2018).

Reynolds Number: The Reynolds Number is a dimensionless value, that represents the ratio between internal and viscous forces in a fluid. It is widely used to predict the flow patterns and the transition between them (Nakayama et al., 2018). It is given as:

$$Re = \frac{\rho u L}{\mu} \tag{2.1}$$

where ρ is the density of the fluid, u the flow speed, L the characteristic length of the observed flow domain and μ the viscosity of the fluid. The concept was developed by Stokes (2009), later popularized through Reynolds (1883) and officially coined by Sommerfeld (1908).

Building upon these fundamental concepts of how a fluid can be described, the chapter expands upon them with the physical foundations describing how a fluid and the changes it can undergo are defined.

2.2 Theoretical Foundation

In the center of this field is a set partial differential equations known as the Navier-Stokes equations, named after Claude-Louis Navier and George Gabriel Stokes. They describe the flow of incompressible fluids and represent a generalization of the Euler Equations for incompressible steady flows (Euler, 1757).

2.2.1 Conservation of Mass

The first equation fulfills the conservation of mass requirement and states that the sum between the outlet and inlet of any system is zero. It is give by the following equation:

$$\frac{D\rho}{Dt} + \nabla \cdot (\rho u) = 0 \tag{2.2}$$

where ρ is the density, u the velocity vector, ∇ is the gradient operator and $\frac{D\rho}{Dt}$ - the material derivative, given as $\frac{Dy}{Dt} = \frac{\partial y}{\partial t} + (u) \cdot \nabla y$. Assuming that the fluid remains incompressible, as is the case with urban scale wind systems, the equation can be simplified to

$$\nabla \cdot u = 0 \tag{2.3}$$

2.2.2 Conservation of Momentum

The second equation governs the Conservation of Momentum. This keeps the momentum in the domain constant and is based on Newton's Second Law of Motion. In the context of fluid dynamics the equation can be represented simply as:

$$\rho \frac{Du}{Dt} = f_{external} + f_{pressure} + f_{viscous} \tag{2.4}$$

where $\rho \frac{Du}{Dt}$ represents the total body force acting on the fluid and $f_{external}$ represents the force applied upon the whole domain through external forces and is given through

$$f_{external} = \rho g \tag{2.5}$$

where g is the gravitational force. The external forces applied to the fluid domain, represented through $f_{pressure}$ and $f_{viscous}$ - the external pressure and viscous force applied, can be represented through the stress tensor (Cauchy, 1827). In the case of fluid dynamics this tensor is split into two terms - the volumetric stress tensor, which represents change to the volume of the body, and the stress deviator tensor, which represents the deformation to the body. Using the definition of the stress tensor for a Newtonian fluid (Stokes, 2009) we obtain the equation for external forces as:

$$f_{pressure} + f_{viscous} = -\nabla p + \nabla \cdot T \tag{2.6}$$

where T is the stress deviator tensor. Substituting eq. (2.6) and eq. (2.5) in eq. (2.4) we obtain:

$$\rho \frac{Du}{Dt} = \rho g - \nabla p + \nabla \cdot T \tag{2.7}$$

One further assumption can be made, namely, that most Newtonian fluids can be seen as incompressible. Although it is not generally true, as for instance, wind is considered incompressible only up to a certain speed, this greatly simplifies the original stress tensor representation. This assumption further simplifies the equation obtained in eq. (2.7), giving us:

$$\rho\frac{Du}{Dt} = \rho g - \nabla p + \mu\nabla^2 u \tag{2.8}$$

It is important to point out, that although the further sections of this chapter will focus on different ways to solve these equations, they cover only incompressible Newtonian fluids.

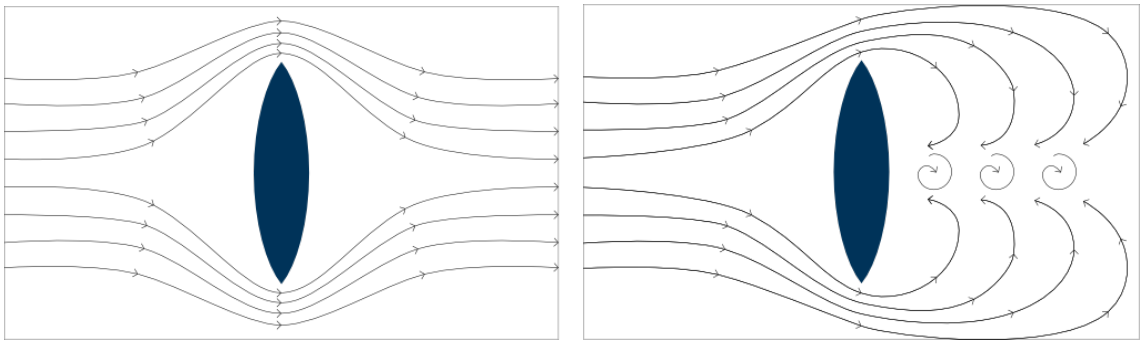
2.2.3 Turbulence

Fluid flows can be further classified into three groups - laminar (Figure 2.1a), turbulent (Figure 2.1b) and a transitional group that transform the flow from one of the previous groups into the other. This classification is based not on properties inherent to the fluid type itself, but rather on forces acting in the flow. The flow type can be accurately categorized based on the Reynolds' number of the observed flow - low Re numbers for laminar flow and very high Re numbers for turbulent ones.

In a laminar flow the fluid moves in layers parallel to the flow direction. They do not mix or intersect. This type of flow is typically very easy to simulate precisely in either numerical simulations or through analytical solutions using only the conservation equations in Section 2.2.1 and Section 2.2.2.

A turbulent flow on the other hand is characterized by its chaotic and erratic flow. Such kind of flow arises, when a velocity difference occurs. Simple examples for a source of such a difference would be a wall or any other solid obstacle. The solid causes a friction resistance between it and the flow, which in turn slows down the fluid and in turn creates the velocity difference (Figure 2.1). The fluid mixes with itself, and it is difficult to predict changes in pressure and velocity. Solving the conservation equations for this type of flow is prohibitive and requires an approximation to achieve this. This type of numerical modelling is often referred to as turbulence modeling.

This type of flow has 4 major characteristics, which make it difficult to model: irregularity, diffusivity, rotation and dissipation. The irregularity of the flow makes modelling it prohibitive or even impossible. This leads to modelling the turbulent flows statistically. The energy responsible for the turbulent flow leads to a rapid mixing of the fluid and increases the rates of momentum, and transfer of heat and mass within the fluid. Furthermore, they have a non-zero vorticity, which is the representation of rotational change of the fluid (Nakayama et al., 2018), which in turn causes vortices.



(a) Laminar Flow **(b)** Turbulent Flow

Figure 2.1 Comparison between flow types

These vortices are then further subject to stretching, due to the conservation of angular momentum. The kinetic energy of the turbulent flow converts quickly into internal energy due to the viscous shear stress and thus requires a persistent source of energy to sustain it. This energy usually transfers from larger vortices to smaller ones until the vortices are small enough that molecular diffusion dissipates the energy and no further vortices are generated.

As can be seen from the physical model representing not only the base equations for laminar flow, but also the complexity that turbulence introduces into such models, requires a complex approach in order to even approximate it using digital tools. In the following section the basic numerical approaches for solving both the Navier-Stokes Equations are presented. Expanding upon these approaches, different approaches to modeling turbulence are also introduced.

2.3 Numerical Solution

To solve the conservation equations, even without the presence of turbulence requires the application of discretization - the representation of continuous problems through a finite scheme, which involves the domain, variables and governing equations. There is a vast array of different numerical methods that solve these equations in different ways: the finite volume, finite element, finite differences, Lattice Boltzmann, and many more. Due to the widespread use of the finite volume method in commercial solvers and in research, this dissertation will only delve deeper into this methodology.

2.3.1 Discretization Scheme

As previously explained, the discretization of the Navier-Stokes equations will require several finite representations of continuous components - namely the domain, equa-

tions and variables associated with them. This section will present the underlying approaches for discretizing the domain, the interpolation schemes required for the computation of the variables in the Navier-Stokes Equations, how the equations are solved and what the most established algorithm is for solving them utilizing the finite volume method approach.

Domain Discretization - Finite Volume Method

The first documented use of this approach was by Evans et al. (1957) named as Particle-in-Cell Method. The methodology has seen a widespread adoption because of its efficient implementation with regard to memory use and speed of computation. The approach also guarantees that momentum and energy are conserved. The core of this method is to divide the observed fluid domain into cells that are arranged into a grid.

Each cell implements the conservation equations given in Section 2.2.1,and Section 2.2.2 and the divergence theorem (de Lagrange, 1773) is applied. To compute the derivative terms in the differential equations, values at the edges (faces) of each cell are needed. This transforms the equations into linear algebraic equations. Each cell can serve as either a boundary cell (at the boundary of the observed volume), a computational node (if fluid can pass through it), or as an obstacle/solid (if the fluid cannot pass through it, and it is not part of the domain boundary).

Each cell consists of edges through which the flux for each parameter changes. These edges do not overlap with each-other and are directly connected to neighboring cells. Generally for each variable ω in the Navier-Stokes equation it is discretized and integrated into each cell C to be solved separately as follows:

$$a_C \omega_C = \sum_n a_n \omega_n + b \tag{2.9}$$

where a and b are coefficients unique to each cell and n refers to the cells that are directly connected to the current cell C (neighbors). The values for the neighboring cells refer to the change to each variable in the current cell coming from that neighbor. These values are obtained from the edge that the current cell shares with them.

Variable Discretization - Interpolation Schemes

To obtain these neighboring values interpolation needs to be applied. There are countless schemes that can be used and in this dissertation we will cover only the core ones, an overview of which can be seen in Figure 2.2. The first and most simple one is the first-order upwind scheme. It assumes that the value at the boundary

between the current cell C and each neighboring cell is equal to the value at the respective centers of each neighbor. Building upon this approach is the central differencing scheme. Similar to the first-order upwind scheme it uses only the current cell and its direct neighbors to compute the boundary values between them. To compute the boundary values it uses a linear interpolation between the values at the center of the current cell C and each neighbor. The Power Law Scheme substitutes the linear interpolation of the previous scheme with a power law equation given as:

$$\omega_{boundary} = \omega_C - \frac{(1 - 0.1Pe)^5}{Pe}(\omega_N - \omega_C) \tag{2.10}$$

where Pe is the Péclet Number. Expanding upon these schemes is the second-order upwind scheme. In this scheme instead of using only the direct neighbor with which the cell shares the boundary one further cell is taken into consideration (Figure 2.2). Finally, the Quadratic Upwind Interpolation for Convective Kinetics (QUICK) Scheme uses two points upstream and one downstream of the current cell and fits a quadratic curve through them.

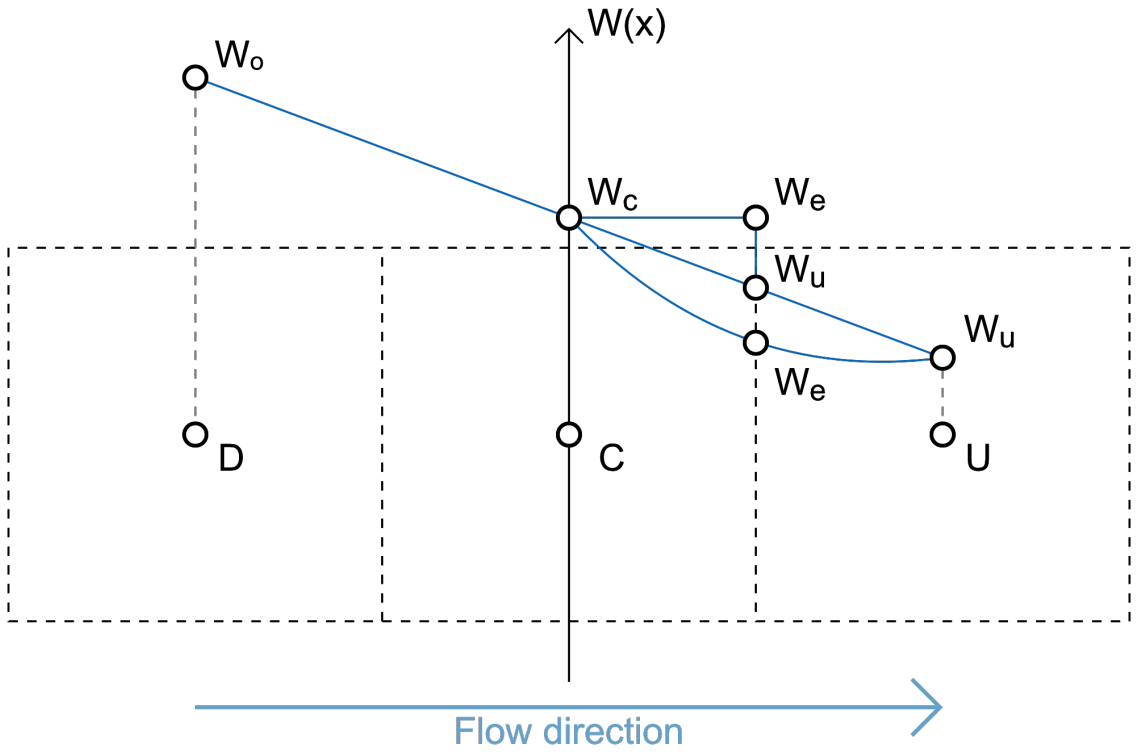


Figure 2.2 Interpolation Schemes

Equilibrium

The goal of these discretized equations is to reach a point where the flow is in equilibrium. To assist in reaching this goal, an additional relaxation factor may be used that can speed up the convergence at the cost of stability of the solution, or it can slow it down, but allow for more stable calculations. The relaxation factor is simply applied as:

$$\omega^{new} = \omega^{old} + \alpha(\omega^{predicted} - \omega^{old}) \tag{2.11}$$

where α is the relaxation factor, ω^{old} the value in the previous step, and $\omega^{predicted}$ is the value obtained from eq. (2.9). Instability in the algorithm comes from oscillations in the computations, caused by numerical errors and imprecision. Such errors often occur with higher Reynolds numbers, where the flow is faster and turbulence can be observed.

To reach equilibrium or convergence the linear equations are solved iterative until the changes of each value, computed this way, are small enough. It is a difficult task to determine when the solution has actually converged, and traditionally it is recommended to observe the residual values for the solution. Residual values measure the error in the solutions of the conservation equations. There are three main residual values that can be followed, an absolute residual in a specific cell, a relative residual in a specific cell, and an overall residual, that measures the error in the whole domain.

SIMPLE Algorithm

Every variable in the conservation equations has an equation that can be solved. Exception to this rule is the pressure. As we can see from Equation (2.7) the gradient of the pressure is used in the momentum equation and this leads to the requirement that the pressure is recomputed every step. For compressible fluids this can be easily achieved, since the pressure is linked to the density of the fluid. This complicates the matter for fluids that are considered incompressible. Although there is no explicit equation for pressure, the Navier-Stokes equations for incompressible fluids consist of four equations with four unknowns - the velocity in each direction and the pressure. Linking the pressure to velocity is possible and the most commonly used algorithm for this is the Semi-Implicit Method for Pressure Linked Equations (SIMPLE)(Patankar and Spalding, 1972). True to its name the algorithm follows a straight forward scheme in correcting the pressure field in each iteration step. It evaluates the pressure at each cell. If the continuity is not fulfilled, due to more or less mass entering it, in comparison to neighboring cells, the pressure is corrected with the application of a correction term, obtained from a pressure correction equation. Based on this algorithm further

improved versions have been developed - SIMPLE Revised (SIMPLER) (Patankar, 2018), SIMPLE Consistent (SIMPLEC) (Doormaal and Raithby, 1984), the Pressure-Implicit with Splitting of Operators Algorithm (PISO) (Issa, 1986) and many more.

Even with an algorithm capable of numerically solving the Navier-Stokes equations, initial values for each cell are needed and appropriate special handling of non-fluid cells - boundaries, inlets, outlets and obstacles - is still needed.

2.3.2 Boundary Conditions

Controlling the values in each cell that is not fluid is called boundary conditions. These conditions dictate how the flow will work and are usually handled at either the start or the end of each iteration step. For each variable that the main numerical algorithm is solving in each cell, a different boundary condition can be set. Based on which component of each value we want to prescribe to these cells - the value itself or its derivative - we have three categories.

The first one is the Dirichlet Boundary Condition, named after Peter Gustav Lejeune Dirichlet. It defines the solution of a differential equation as a known value. For an ordinary differential equation, this is done by assigning a given number to the function evaluation for its interval values. For a partial differential equation the function is substituted for a defined and known function that has the boundary of the original one as a domain.

The second type is the Neumann Boundary Condition. It specifies the solution to the derivative of the differential equation. It operates in the same way as the Dirichlet Boundary Condition for both ordinary and partial differential equations, but instead of assigning known values and functions to the main function, it assigns them to its derivative.

Finally, there are also mixed boundary conditions that combine both the Neumann and Dirichlet Boundary Conditions.

Inlet and Outlet

The inlet and outlet cells are the main source of change inside the fluid domain. There is a vast range of different boundary conditions that can be assigned to these cells, but for the purposes of this dissertation, only a relevant handful will be examined.

The first type of boundary conditions are the pressure inlet and outlet. They are often utilized when the velocity and flow rate of the domain are unknown. For a pressure inlet boundary for an incompressible fluid this is given as:

$$p = p_{static} + \frac{1}{2}\rho|v|^2 \tag{2.12}$$

where p_{static} is a predefined value given at the start of the simulation. If the observed fluid also has a heat transfer, the inlet temperature is set to a constant at the start. For the pressure outlet boundary condition, a static value is defined and is considered the pressure of the environment into which the outlet connects.

The second type of boundary condition is the velocity inlet. As the name suggest it defines the velocity vectors at each cell, that is assigned as an inlet. Different types of profiles can be defined for the inlet velocity field - typically a uniform profile, but it may also depend on the geometry of the inlet or have a function to compute each cell's velocity value.

Outflow boundary conditions can be used to set up the outlets of the observed fluid, when exact details about the flow velocity and pressure are not explicitly known. The outflow boundary condition assumes a zero normal gradient for every variable, except pressure.

Walls and Obstacles

These types of conditions are used to describe obstacles within the flow (e.g. buildings in an urban context) and/or to describe the boundary regions of the simulation domain, where no inlet or outlet conditions need to be specified (e.g. the ground boundary in an urban context).

A typical boundary condition for solid objects is the no-slip boundary condition. This condition sets the velocity component that is normal to the cell to 0 and the tangential component to the velocity at the wall. If the wall or obstacle has small details, that are prohibitive to model, a further wall roughness value can be assigned, that describes the smoothness of each cell that is assigned as a wall or no-slip boundary. Such roughness wall functions are usually connected with the turbulence model used in the numerical solver and are usually represented as functions over the surface domain.

If the fluid domain is large, but with no obstructions in the flow, a symmetric boundary condition may be applied. Such condition state, that the fluid in the domain and the fluid on the other side of the domain have the same values. It can be viewed as a mirroring boundary/plane that allow us to reduce computation by artificially shrinking the domain.

Similar to the symmetric boundary condition the periodic boundary condition aims to reduce computation time by limiting the observed domain as well. It is used when there is a repeating pattern in the fluid domain (such as a tube with a fan at one side). The domain can be reduced by focusing on only one instance of the pattern and enforcing through the periodic boundary conditions on each side, where the pattern would repeat, that the values are correct.

Porous Medium

Some obstacles and walls have small pores in them, through which fluid can pass, e.g. vegetation. Modelling such geometry and refining the grid domain to such details would increase the computational overhead to prohibitive levels. Instead, the cells of such obstacles are labeled as porous medium and the flow through them is defined by Darcy's Law (Whitaker, 1986):

$$flowRate = \frac{-kAp_{drop}}{\mu L} \tag{2.13}$$

where k is the permeability of the medium, A the cross-section area of the medium, p_{drop} the pressure drop through it and L the length of the medium.

2.3.3 Meshing

One key factor that controls how well and how fast a solution would converge is the choice of the type of cells used and their arrangement in grids. The process of fitting the cells and the grid in the simulation domain, with the goal of maximizing the accuracy of the numerical simulation, is called meshing.

Based on the dimensions of the observed fluid domain, either 2D or 3D, the cells can have different basic geometrical shapes. In 2D they are triangles and quadrilaterals and in 3D we have various types of prisms, tetrahedrons and pyramids. Additionally, in both scenarios it is possible to use an arbitrary shape to describe each cell.

Each cell represents a domain of the fluid observed by a node, traditionally a point on the grid, attached to each cell. Each boundary of a cell is called a face and in the case of 3D, each boundary of each face has edges (those are the same in the case of 2D). This can be more easily visualized in Figure 2.3. Groups of cells that have similar functionality are often grouped into zones - such as boundary zones, fluid zones, and inlet and outlet zones.

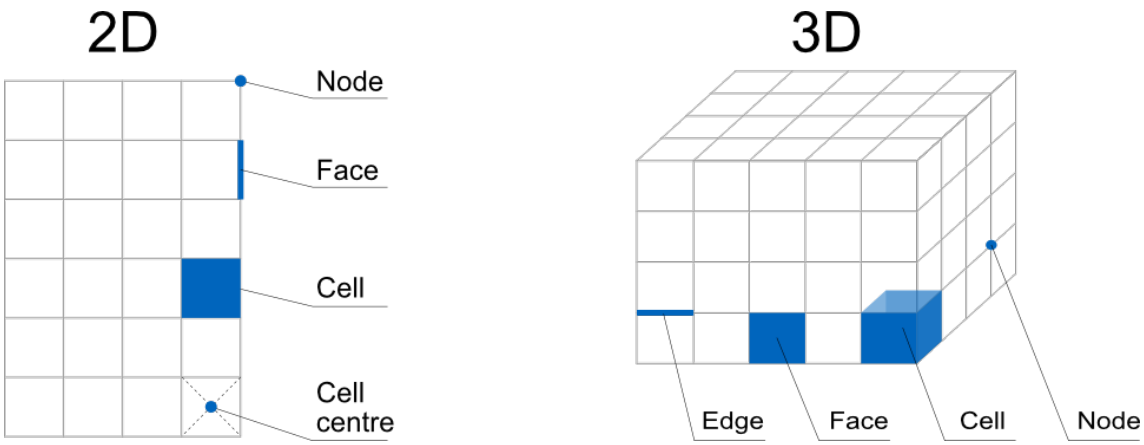


Figure 2.3 Grid Structure and Nomenclature

Structured Grids

Structured Grids are the most straightforward way of grouping and organizing cells in the fluid domain. Its lines go through the whole domain which requires that each cell is of the same type - a quadrilateral in 2D or a hexahedron in 3D. On one hand, this can lead to regions in the domain, where nothing needs to happen (e.g. in a pipe system). On the other hand this has the advantage of providing a very easy way to access neighboring cells through an easy indexing system.

This type of grid leads to the problem, that if a region has small details that need to be captured by the grid, then the size of each cell will be significantly smaller. This in turn leads to higher computation times due to the increase of cells. A way to alleviate some of these issues is to use a multi-block approach. In this way the domain is split into subdomains, each with its own grid size and refinement level. This adds a requirement of handling the edge between differently sized subdomains.

Unstructured Grids

On the opposite side of this spectrum of grids is the unstructured grid. Cells can be arranged freely and the cells themselves do not need to have the same size, although they do need to be of the same type, e.g. triangles in 2D or tetrahedrons in 3D. This gives a greater flexibility when discretizing the fluid domain without the need of multi-block structures as was the case with structured grids. This approach has the downside of requiring a more complex system of indexing neighboring cells, e.g. linked lists.

Hybrid Grids

Finally, it is possible to apply a hybrid approach in the generation of grids. The use of structured grids can be used in the parts of the domain, where there is little happening (no obstructions or turbulence), while for the more critical locations of the simulation, an unstructured grid is applied. Both components can then be connected through a grid line, that does not need to match the block boundaries. This type of modelling offers the optimal compromise between the precision and speed of the result, but requires the largest amount of effort to model correctly.

2.3.4 Turbulence Solver

Due to the characteristics of turbulence, detailed in Section 2.2.3, it is impractical to attempt to solve the equations analytically for the whole domain. Instead, different models have been developed to simplify the equations in different ways.

Reynolds-Averaged Navier-Stokes

An approach in modeling turbulence is to predict the time averaged velocity, pressure and temperature fields and any other field that will contain fluctuating values. The additional equations stemming from this method are called the Reynolds-Averaged Navier-Stokes (RANS) equations. To model each varying value they need to be decomposed into an equilibrium and a fluctuating component as follows:

$$x(t) = X + x'(t) \tag{2.14}$$

where $X = \frac{1}{\triangle t} \int_0^{\triangle t} x(t)\, dt$ represents the equilibrium component and $x'(t)$ the fluctuating part. The time averaged value of each such component, given as $\overline{x(t)}$ is reduced only to the equilibrium value of the component since the fluctuating component will average out to 0 for a large enough time frame by definition.

Using this definition of a fluctuating value we can represent the velocity as $u(t) = U + u'(t)$ and the pressure as $p(t) = P + p'(t)$. With this decomposition approach, known also as the Reynolds Decomposition, the continuity equations can be modified. For Equation (2.3) we obtain:

$$\nabla \cdot U = 0 \tag{2.15}$$

Expanding the material derivative for Equation (2.8) and substituting each turbulent value for its decomposed variant, the equation yields for each direction as follows:

$$\frac{\partial \rho U_u}{\partial t} + \nabla \cdot (\rho U_u U) = -\frac{\partial P}{\partial x} + \mu \nabla^2 U_u + \left[-\frac{\partial (\rho \tau_{xx})}{\partial x} - \frac{\partial (\rho \tau_{xy})}{\partial y} - \frac{\partial (\rho \tau_{xz})}{\partial z} \right] + \rho g \quad (2.16)$$

$$\frac{\partial \rho U_v}{\partial t} + \nabla \cdot (\rho U_v U) = -\frac{\partial P}{\partial y} + \mu \nabla^2 U_v + \left[-\frac{\partial (\rho \tau_{yx})}{\partial x} - \frac{\partial (\rho \tau_{yy})}{\partial y} - \frac{\partial (\rho \tau_{yz})}{\partial z} \right] + \rho g \quad (2.17)$$

$$\frac{\partial \rho U_w}{\partial t} + \nabla \cdot (\rho U_w U) = -\frac{\partial P}{\partial z} + \mu \nabla^2 U_w + \left[-\frac{\partial (\rho \tau_{zx})}{\partial x} - \frac{\partial (\rho \tau_{zy})}{\partial y} - \frac{\partial (\rho \tau_{zz})}{\partial z} \right] + \rho g \quad (2.18)$$

where the τ values are part of the Reynolds stress tensor. Finding accurate ways in predicting or computing this stress tensor has lead to the development of multiple solvers. This dissertation will cover briefly some of the more widely used solutions. The main approach to these models is to compute the turbulent viscosity. This underlying theory was experimentally observed by Boussinesq (Boussinesq, 1903) and stated that the Reynolds stresses can be linked to the mean rate of deformation as follows:

$$\tau_{ij} = \mu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) \quad (2.19)$$

where μ_t represents the turbulent viscosity.

$\mathbf{k - \epsilon}$ **Models** focus on the turbulent kinematic energy k and its dissipation rate ϵ. If both values are available, then the turbulent viscosity can be obtained from

$$\mu_t = \frac{k^2}{\epsilon} \quad (2.20)$$

Following the time averaging approach in Equation (2.14) we can represent the kinetic energy as a sum of an equilibrium and fluctuating component as $k(t) = K + k$. The equation for both values are given as:

$$\frac{\partial (\rho K)}{\partial t} + \nabla \cdot (\rho K U) = \nabla \cdot (-PU + 2\mu U E_{ij} - \rho U \overline{u_i' u_j'}) - 2\mu E_{ij} E_{ij} - (-\rho \overline{u_i' u_j'} E_{ij}) \quad (2.21)$$

$$\frac{\partial (\rho k)}{\partial t} + \nabla \cdot (\rho k U) = \nabla \cdot (-\overline{p' u'} + 2\mu \overline{u' e_{ij}'} - \rho \frac{1}{2} \overline{u_i' u_i' u_j'}) - 2\mu \overline{e_{ij}' e_{ij}'} + (-\rho \overline{u_i' u_j'} E_{ij}) \quad (2.22)$$

Where E_{ij} is the mean rate of deformation tensor and e_{ij}' its fluctuating component. These equations contain further unknowns which make it hard to model. Furthermore, the analytical equation for the dissipation rate ϵ consists of several unknown terms of higher order. A more simplified version proposed by Launder and Spalding

(1974) minimizes these problems and presents two equations for the kinetic energy and dissipation rate that can be solved:

$$\frac{\partial(\rho k)}{\partial t} + \nabla \cdot (\rho k U) = \nabla \cdot (\frac{\mu_t}{\sigma_k} \nabla k) + 2\mu_t E_{ij} E_{ij} - \rho\epsilon \tag{2.23}$$

$$\frac{\partial(\rho \epsilon)}{\partial t} + \nabla \cdot (\rho \epsilon U) = \nabla \cdot \left[\frac{\mu_t}{\sigma_\epsilon \nabla \epsilon}\right] + C_{1\epsilon}\frac{\epsilon}{k} 2\mu_t E_{ij} E_{ij} - C_{2\epsilon}\rho\frac{\epsilon^2}{k} \tag{2.24}$$

where σ_k and σ_ϵ are the Prandtl numbers for respectively k and ϵ and connect the respective diffusivity of them to the turbulent viscosity. $C_{1\epsilon}$ and $C_{2\epsilon}$ are model constants and can be freely set. From these two new simplified equations the Equation (2.20) can be solved as:

$$\mu_t = C_\mu \frac{k^2}{\epsilon} \tag{2.25}$$

where C_μ is again a constant. There are several further RANS models that build upon original $k - \epsilon$ model. A brief overview of a few of the most established version and their advantages is presented.

Renormalization Group Method (RNG) $\mathbf{k - \epsilon}$ **equations** builds upon the original $k - \epsilon$ method by introducing additional terms in the ϵ Equation (2.24) and takes into consideration swirls (Yakhot et al., 1992). In this way it achieves higher accuracy when computing wall heat, mass transfer, high streamline curvature and strain rate. Its modified k and ϵ equations are given as follows:

$$\rho U_i \frac{\partial k}{\partial x_i} = \mu_t S^2 + \frac{\partial}{\partial x_i}\left(\alpha_k \mu \frac{\partial k}{\partial x_i}\right) - \rho\epsilon \tag{2.26}$$

$$\rho U_i \frac{\partial \epsilon}{\partial x_i} = C_{1\epsilon}\left(\frac{\epsilon}{k}\right)\mu_t S^2 + \frac{\partial}{\partial x_i}\left(\alpha_\epsilon \mu \frac{\partial \epsilon}{\partial x_i}\right) - C_{2\epsilon}\rho\left(\frac{\epsilon^2}{k}\right) - R \tag{2.27}$$

where the $\alpha_k, \alpha_e, C_{1\epsilon}, C_{2\epsilon}$ are values derived through the use of RNG theory, R is a term representing the mean strain and turbulence and $S = \sqrt{2S_{ij}S_{ij}}$, $S_{ij} = \frac{1}{2}\left(\frac{\partial U_j}{\partial x_i} + \frac{\partial U_i}{\partial x_j}\right)$

Realizable $k - \epsilon$ **model** utilizes the same equations for k as the standard $k - \epsilon$ model, but improves on the equation for ϵ and also changing C_μ in Equation (2.25) from a constant to a variable dependent on the velocity field, k and ϵ. The advantages of these changes can be observed in higher accuracy of prediction for flows focusing on planar and round jets, rotation, recirculation and strong streamline curvature. The new equation for ϵ is given as:

$$\rho\frac{D\epsilon}{Dt} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x_j}\right] + \rho C_1 S_\epsilon - \rho C_2 \frac{\epsilon^2}{k + \sqrt{v\epsilon}} + C_{1\epsilon}\frac{\epsilon}{k} C_{3\epsilon} G_b \tag{2.28}$$

where $C_1, C_2, C_{1\epsilon}, C_{3\epsilon}$ are constants and G_b represents the generation of kinetic energy due to buoyancy.

Large Eddy Simulation

An alternative approach to creating a turbulence model was originally proposed by Smagorinsky (1963) and then later refined by Deardorff (1970), called Large Eddy Simulation (LES). Instead of observing turbulence as random and chaotic occurrence, that has to be solved through statistical averaging, it focuses on solving the turbulence equations for the largest eddies in the simulated domain and filtering out the smaller scale ones. This omits the effect of the smaller eddies completely from the numerical simulation and has to be introduced through various Sub-Grid-Scale (SGS) models.

A filtering function in the scope of LES can be applied to the spatial or temporal component of a value, or both. For a given value field $X(x,t)$ it is defined as follows:

$$\overline{X(x,t)} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} X(\alpha,\beta)G_{\triangle}(x-\alpha,t-\beta)d\beta d\alpha \tag{2.29}$$

where $G_{\triangle}$ is a filter convolution kernel. This filtering function eliminates the scales that are smaller than $\triangle$ which are then resolved through the SGS model. Typical filters for G are the Gaussian filter, box filter or top-hat filter. Given this, every variable can be split into a filtered and sub-grid component as follows: $X = \overline{X} + X'$. Applying this decomposition for the velocity and pressure variables and substituting them in the incompressible Navier-Stokes equations (Equation (2.8) and Equation (2.3)) we obtain:

$$\frac{\partial \overline{u}}{\partial t} + \overline{u}\cdot\nabla\overline{u} = -\frac{1}{\rho}\nabla\overline{p} + \mu\nabla^2\overline{u} - \nabla\cdot\tau \tag{2.30}$$

$$\nabla\cdot\overline{u} = 0 \tag{2.31}$$

where τ_{ij} is the SGS stress tensor defines as:

$$\tau_{ij} = \overline{u_i u_j} - \overline{u_i}\,\overline{u_j} \tag{2.32}$$

The spatial resolution of the grid needs to be on the order of $\triangle$ so that the LES Equations can be solved numerically. The challenge of these equations is that τ_{ij} is open since it contains terms that are obtained through the filtered velocity variable $\overline{u}$. To approximate and close this equation an eddy-viscosity model for the SGS stress tensor is defined as follows:

$$\tau_{ij}^{ev} = -2\mu_{sgs}\overline{S}_{ij} \tag{2.33}$$

where μ_{sgs} is the kinematic eddy viscosity and $\overline{S}_{ij}$ is the filtered rate of strain tensor. The most widely used model for the μ_{sgs} is the Smagorinsky model (Smagorinsky, 1963):

$$\mu_{smag} = (c_s \triangle)^2 |\overline{S}| \tag{2.34}$$

where c_s is the Smagorinsky coefficient, $\triangle$ the characteristic scale and $|\overline{S}|$ is an estimation of velocity differences over length-scale. Building upon this model is the Dynamic SGS (DSGS) model (Germano et al., 1991) that transforms c_s from a constant value for the whole field to a variable that has to be computed locally every time step. There are further models that solve the SGS eddy viscosity in various different approaches, but are not relevant for the purpose of this dissertation and will not be covered further.

Direct Numerical Simulation

The most precise information can be obtained by solving the Navier-Stokes equations directly without introducing any turbulence models, but instead solving the equations directly. This approach is often called Direct Numerical Simulation (DNS). This requires that the complete range of spatial and temporal scales of the turbulence must be resolved in the computational mesh (Orszag, 1970).

The mesh for the domain must satisfy the requirement $n * \triangle n > L$, where n is the number of points along a direction with $\triangle n$ distance between them and L is the integral scale. Which in turn leads to the fact that the step between each point $\triangle n$ cannot be greater than the smallest dissipative scale of the turbulence model (the Kolmogorov scale η). This in turn leads to the stipulation that for the three-dimensional DNS; a number of mesh points greater than $Re^{9/4}$ are required, where $Re = \frac{u'L}{\mu}$ is the turbulent Reynolds number, u' the Root Mean Square (RMS) of the velocity and μ is the kinematic viscosity.

A further limitation of this approach is the integration of the simulation in time, as it is done by an explicit method as well. This requires that the timestep to be small enough that the fluid particle does not move the full length of $\triangle n$. This makes the number of time-integration steps proportional to $\frac{L}{C\eta}$, where C is the Courant number.

Based on the scale of the time steps and the number of points required for a detailed enough mesh, the number of floating-point operations required grows directly in proportion with Re^3. This leads to the conclusion that even scenarios with low Reynolds numbers require a prohibitively large amount of memory storage and even more computational time.

2.4 Urban Wind Simulations

Originally the field of CFD was based on the failed experiments and simulations of Lewis Fry Richardson, which he used for the prediction of weather patterns (Richardson, 2007). Since then the field has expanded to cover smaller scales all the way to cardiovascular systems. In this sense there are three defined simulation domain scales that can be observed - macroscale, mesoscale and microscale (Blocken, 2014). Based on the size of the simulated domain certain phenomena can be more accurately captured - for instance urban heat island can be observed in a simulation domain on the scale of roughly 10 kilometer and cyclones and global weather patterns at a much larger scale - at almost 10 000 kilometers. The importance of such simulations in the urban planning are clear - through the improvement of natural passive ventilation, cities can alleviate issues associated with climate change, such as heat islands, but also improve the energy efficiency of cities and help them towards their goal of being climate positive Liu et al. (2022).

There are various areas in which CFD simulations can be applied, and every area has its own drastically different set up, boundary conditions and specific geometry nuisances that need to be taken into consideration. In this section a more detailed overview of the unique boundary conditions, geometry detailing and dimensions of urban scale wind simulations will be presented. It is important to point out, that there have been several best practices and guidelines for the set-up of urban scale wind simulations over the past 50 years with two being most prominent - the COST Guidelines (Franke et al., 2007) and the AIJ Guidelines (Tominaga et al., 2008). The COST Guidelines have a robust and detailed overview about the setup of the simulation domain but focus primarily on RANS approaches, although they also cover LES. The AIJ Guidelines build upon the COST Guidelines and expand upon them by introducing validation cases for simulation models that use wind tunnel experiment data for comparison. The three main areas that these guidelines focus on are: the dimension of the domains and explored scenario, the meshing and the importance of geometry in the domain, and the boundary conditions. A more detailed overview of the requirements set out for these areas is presented in the following sections.

2.4.1 Dimensions of the Problem

The most established application of CFD simulations in an urban context happen on the microscale of Figure 2.4. Experiments and validation research usually select a small domain varying from a few hundred meters in each direction to a few kilometers

(Wang et al., 2018; Reda et al., 2017b; Hang et al., 2009; Weerasuriya et al., 2018; Liu et al., 2017; Ramponi et al., 2015).

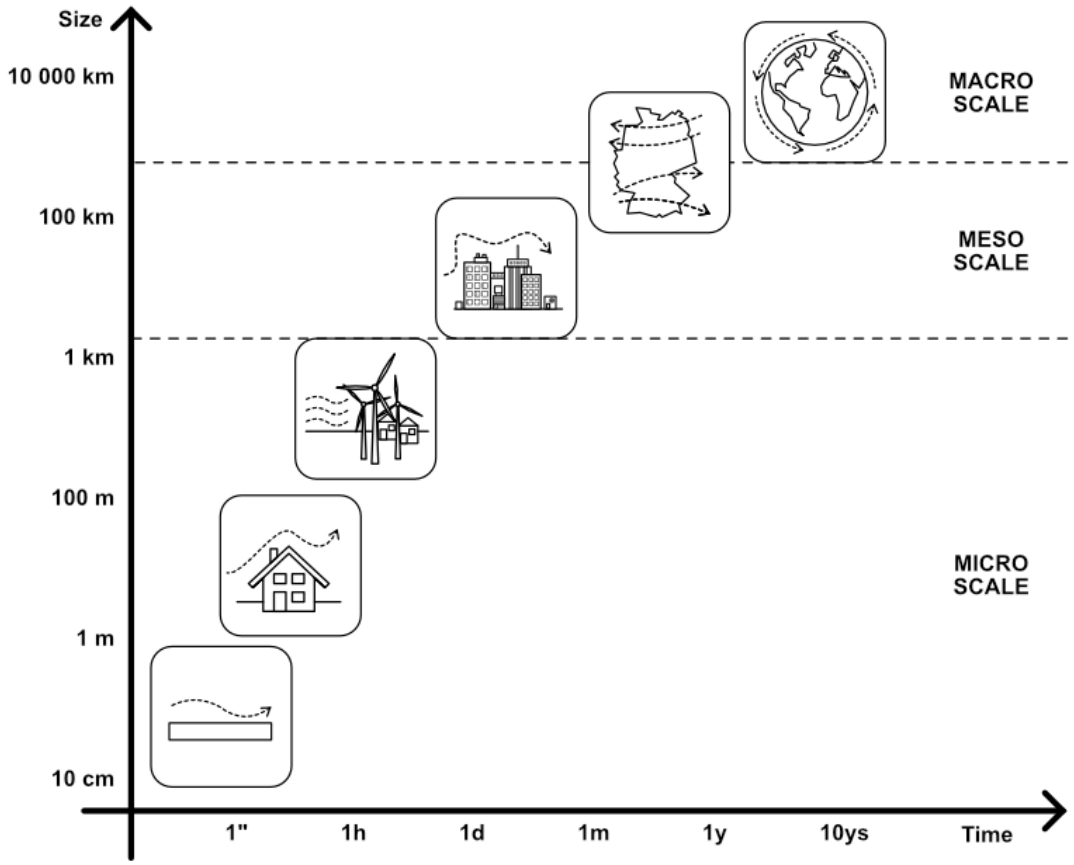


Figure 2.4 Domain scale

According to both Franke et al. (2007) and Tominaga et al. (2008) the selected simulation domain needs to be extended in all cardinal directions and the top boundary needs to be expanded as well. This is a crucial modification that allow for several phenomena associated with wind simulations to be taken into consideration:

- The additional domain space between the inlet, outlet and other sides of the simulation and the actual region allows for the wind to be fully formed and for the turbulence in the wake of the buildings to be fully develop

- Additional roughness parameters added to the ground boundary of the domain allows for an approximation of potential small scale geometry that would be realistically between the inlet and the objects

- The additional space between the objects and the top boundary of the simulation domain allows for more accurate forming of wind patterns throughout the simulation domain

Through various and multiple experiments it has been shown that the minimal requirements for these additional parts of the domain can be obtained using rules based on the geometry in the domain (Franke et al., 2007). Assuming that the tallest building in the simulation domain has a height of H_{max} the minimal values for the additional domain space can be seen in Table 2.1.

Boundary Side	Additional Space
Top	$5H_{max}$
Lateral	$5H_{max}$
Flow Direction In front	$5H_{max}$
Flow Direction Behind	$15H_{max}$

Table 2.1 Minimal additional domain space based on H_{max}

It is also important to note that Franke et al. (2007) recommend that the blockage ratio remains below 3%. This ratio is defined as the projected area of the objects in flow direction to the cross-sectional area of the surrounding domain. This leads to very large simulation domains which is one of the primary reasons for the longer computational times needed.

The observed simulation domain is traditionally selected as a rectangle or circle area, that is then expanded to a larger rectangle that encompasses the requirements in Table 2.1.

2.4.2 Geometry

Another critical part of a precise urban scale wind simulation is the detail and refinement of the geometry involved. There are 3 main components that factor into the simulation: buildings - defined as solid objects; vegetation - traditionally observed as a porous medium; moving objects - such as vehicles and pedestrians can still have an adverse effect in the lower urban layers.

Buildings

Buildings in the context of urban planning have a large range of refinement levels, known as Level of Detail (LoD) (Abualdenien and Borrmann, 2022). In the early

stages of urban planning it is typical, that the buildings are represented through simple volumes such as cuboids. It is only through the later steps of planning that the model gets refined.

Since buildings are modeled as obstacles, a higher level of detail to the model would allow for more precise simulation results. Depending on the simulation scenario, whether it be pedestrian comfort or general exploration of larger weather patterns in an area, the refinement of the model may play a crucial role.

Due to the large simulation domain and potential amount of geometry in it, most simulation setups represent buildings through a 2D contour, that is extruded to the average height of the building. The complexity of the contour is dependent on the goal of the study - simple shapes such as squares and rectangles are often used for the exploration of optimal city design or are used as validation models, to verify that a CFD simulation is set up correctly. Complex contours are used most often in all other cases - to study wind patterns or to analyze pedestrian comfort. Generally roof detailing is omitted and can be seen only in studies that explore the feasibility of wind energy farming in urban context (KC et al., 2019). Finally, facades are usually not modeled directly, since such small details will require a higher order of grid refinement around buildings. In order to approximate the details of a facade - balconies, windows and decoration - a wall roughness factor is usually used (Franke et al., 2007).

Vegetation

Vegetation can play a big role in the thermal comfort and wind conditions in urban areas. Due to their structure it is impractical to model each element as a complex mesh. They have to be represented through a model that integrates into the existing environment. A typical approach would be to specific additional terms that are added onto the selected equations for momentum, turbulent kinetic energy and turbulence dissipation rate for the cells that would contain vegetation. Similarly, an additional term to control the heat transfer can be added. Due to the diverse nature of vegetation and its integration into the urban environment there are multiple ways of representing the geometry as simplified shapes that implement various versions of a porous medium (Gromke et al., 2015; Qin et al., 2019; Santiago et al., 2019; Mughal et al., 2021; Sonnenwald et al., 2016).

Moving Objects

Moving objects in the environment can be observed on the ground level in the first few meters above the ground level. These elements can have a drastic effect on

pedestrian wind and heat comfort (Mochida and Lun, 2008). Usually such elements are omitted from the models. To achieve an approximation of how such moving objects could affect the simulation a high roughness factor for the ground boundary can be applied in areas where high traffic can be observed.

2.4.3 Boundary Conditions

A crucial element for obtaining accurate results is defining the boundary conditions of the simulation domain and assigning appropriate ways to handle non-fluid cells in the domain.

Domain Boundaries

The inflow and outflow boundary conditions are usually positioned on opposing sides of the domain and are responsible for the generation of change within the domain. The selection of appropriate models for inflow conditions largely depends on the scenario and the underlying turbulence model used (RANS, LES, etc.). The outflow conditions are usually modeled in such a way that depends on the inflow model itself. Inflow boundary conditions are set for the sides (or parts of each side) from which the wind direction should come. This means that for the sides that do not have an inflow or outflow boundary conditions custom boundary conditions need to be set that approximate an open domain.

Franke et al. (2007) and Tominaga et al. (2008) suggest utilizing a logarithmic profile for the inflow that matches the terrain via a roughness length z_0. For the simulation of existing urban areas it is suggested to use local weather local stations or use physical wind tunnel simulations. For the various models of RANS the velocity profile and turbulence values are required. Utilizing the assumption that the inflow layer is in equilibrium until reaching the simulation domain, the values can be obtained. To create such inflow profiles it is often suggested to first perform a simulation in the same domain, but without any obstacles in the environment in order to validate that the prescribed profile remains constant.

For LES and other similar turbulence models time dependent inflow conditions are required. A widespread approach is the creation of artificial stochastic data utilizing statistical description of the turbulence. (Kempf et al., 2005; Kondo et al., 1997)

For outflow boundary conditions, where the majority of the flow would leave the domain, conditions must be applied in such a way that the flow leaving through this boundary corresponds to a fully developed flow. Typical conditions for RANS models is to either prescribe a constant pressure boundary condition or set the derivatives

of all variables in the outflow domain to 0. For LES a convective outflow condition is most often used. (Ferziger et al., 2002; Driller and Kaltenbach, 1999)

In order to approximate an encapsulating environment outside the simulation domain symmetrical boundary conditions are applied to the lateral sides of the domain, that do not have inflow or outflow boundary conditions set.

Obstacles

Defining proper boundary conditions for obstacles are crucial for the proper creation of vortices and capturing the intricate phenomena of turbulence. The direct approach requires solving the turbulence models very close to the wall. In this scenario a damping component is added to the turbulence equations in order to increase the importance of the molecular viscosity. Various implementations are available based on the exact model utilized (Chien, 1982; Hassid and Poreh, 1978; Wilcox and Rubesin, 1980; Hoffman, 1975; Launder and Sharma, 1974). Such methods resolve the velocity close to the wall and require a finer meshing next to the obstacles in order to capture the large fluctuations in the velocity components that may occur. This in turn increases the computational costs for urban wind simulations greatly.

This has led to the development and refinement of so-called wall functions, that avoid the necessity of finer grids near walls. Such functions are empirically derived and are used as a transitional layer between the obstacle and the fully developed turbulence flow. These functions are based on the assumption that velocity close to the wall is the same for all types of turbulent flow. This assumption was first observed by Von Kármán (1931) and is known as the Law of the Wall. It simply states the velocity of the turbulent flow is proportional logarithmic distance to the specified obstacle and can be expressed as:

$$u^+ = \frac{lny^+}{k} + C^+ \tag{2.35}$$

where u^+ is the velocity parallel to the wall, k the Von Kármán constant, C^+ a constant and y^+ is the distance to the wall. Blocken et al. (2007) later compared various interpretations of this type of wall function in commercial software solutions with the additional terms for taking into consideration the roughness of the wall.

Similarly, for LES problems wall functions can be applied (Mason and Callen, 1986), or alternatively the distributed roughness approach (Nakayama et al., 2005) can be used.

Atmospheric Boundary Layer

The top boundary of the simulation domain is crucial in the development of the turbulent wind flow before reaching the actual observed domain. Due to minimal space requirements needed in the simulation domain (see Table 2.1) the simulation domain encompasses the lower regions of the atmosphere - the Atmospheric Boundary Layer (ABL). Richards and Hoxey (1993) proposed and experimentally validated that for a $k-\epsilon$ RANS model the ABL can be represented through a constant shear stress that is similar to the inflow profile of the domain. Later Hargreaves and Wright (2007) demonstrated and validated the assumptions made by Richards and Hoxey (1993) and that the boundary conditions for the top of the domain need to be specified separately from the inflow parts of the domain. Alternatively Blocken et al. (2007) proposed assigning the inflow values at the top of the inflow side to the entire top boundary domain.

2.5 Conclusion

This chapter highlighted the governing equations for incompressible laminar Newtonian fluids, to which air and wind count to, and provided a brief explanation of the complexity behind turbulence modeling. Furthermore, the chapter provided a brief overview of the established modern day approach of solving the Navier-Stoke Equations in a numerical environment and presented some core approaches in turbulence modeling - Reynolds-Averaged Navier-Stokes and Large Eddy Simulations. Finally, the chapter outlined the minimal requirements for the set-up of an accurate urban scale wind simulation - from expanding the simulation domain in all directions, to introducing roughness values for solid boundaries in the domain, such as the ground and building walls.

From this chapter it is clear that there are no clear ways to model fluid simulations in a numerical environment, with different approaches to turbulence being developed for specific scenarios. The requirement of urban scale simulations, to expand the simulation domain beyond the selected area, to accommodate for the correct simulation of wind patterns, further compounds the issues discussed in Chapter 1 - making such simulations even larger and more costly to perform. Due to challenges introduced in modeling turbulence, this dissertation will focus primarily on novel approaches to resolve the issues of interoperability, user-friendliness, adaptability, and flexibility.

3 Artificial Intelligence

The modern day origin of AI can be traced to the 1950s with the creation of Turing's Test (Turing, 1950), the use of the Ferranti Mark 1 to develop the first machine that can play games (Schaeffer, 2013) and the creation of the Logic Theorist that was capable of proving mathematical theorems (Newell and Simon, 1956). The actual term Artificial Intelligence (AI) was first coined by John McCarthy at the Dartmouth Summer Research Project on Artificial Intelligence in 1956. The field has had a very turbulent history only achieving a more established use in the last few decades with the increase in performance of modern day machines and the focus on specific problems, rather than more general concepts. AI attained relevance and importance in the commercial sectors largely due to its capability to process vasts amount of data in only a fraction of the time needed by traditional approaches.

3.1 Definition and Overview

In modern day literature the terms AI, Machine Learning (ML) and DL have been used interchangeably, often times describing vastly different approaches, methods or algorithms. In order to achieve a higher consistency within this dissertation a clear distinction and the purpose of each these three definitions will be shortly outlined. These definitions' server predominantly to classify algorithms based on their complexity, a further refinement, classifying the methods based on what tasks they solve, how they process the data and what the desired outcome of the method is introduced, is also presented.

3.1.1 Classification based on complexity

Artificial Intelligence serves as an encompassing super set of various categories of algorithms, approaches and methods that solve specific or broad tasks by following a rule-based structure of extracting relations and knowledge through observation and data. Over the years various definitions of what AI have emerged. Russell (2010) provide an overview of the four main definitions of AI that place emphasis on one

of four types of intelligence. They can broadly be summarized as thinking humanly, acting humanly, thinking rationally and acting rationally. The thinking definitions focus on the thought process and the reasoning behind decisions, while the acting focuses on the behavior. The human and rational parts of the definitions focus on the expected results compared to either the human performance or an ideal performance. Many definitions and focus areas have been developed over the years and subsequently discarded or have lost traction. The modern day AI methods focus predominantly on the "intelligent" or "rational" agent, which focuses on the fourth definition of AI - acting rationally.

Machine Learning is a subset of algorithms, methods and approaches inside the field of AI. ML builds models based on information in order to predict outcomes or make informed decisions without the need for user control. It is capable of finding patterns within the provided information and extrapolate patterns which it uses to adapt to new input and react rationally (Russell, 2010). Examples for such algorithms can be Linear Regression, Decision Trees, Support Vector Machines (SVM), Random Forests and many more.

Deep Learning is a subset of ML that applies the same core principals as its superset. It expands upon the agents used in ML by introducing a more complex structure in its learning process thus uncovering further hidden relations between various components of the information that it is learning from (Russell, 2010). Its modern day application and research is primarily focused on multi-layer Artificial Neural Network (ANN).

3.1.2 Classification based on task

As can be seen from the definitions of AI, ML and DL it makes little sense to subdivide the methods, algorithms and techniques based on these sets. A more apt way of describing and categorizing them is through the way they process the information. There are three main ways an agent can learn from data, based on not only the input data, but also the outcome.

Supervised Learning has as a main goal to create a mapping between a set of input and output parameters, the supervision in such methods is represented through the known or expected outcome of the input, thus limiting the agents scope in extracting information from the data. Formalized mathematically the agents try to find a function f for a set of pairs (x, y), where x represents the input information and y the output information so that $f(x) = y$.

Unsupervised Learning methods are only provided with the input information. Agents then attempt to either learn new representations of the provided input or to

generate a model, that allows for the creation of new inputs. Such agents learn a model $P(\boldsymbol{x}, \boldsymbol{z})$, where $\boldsymbol{x}$ is a vector of inputs and $\boldsymbol{z}$ are unobserved, unknown variables that can be used to represent the data in some way. The goal of the learning is to find an appropriate representation for $\boldsymbol{z}$.

Reinforcement Learning is similar to unsupervised learning in that the agents are provided only with a set of input information and actions that it can take. Based on the actions it takes, and what the outcome is based on the information it is provided with, it receives a score. Their goal is to maximize this result.

Due to the vast and heterogeneous nature of AI agents, the dissertation will primarily focus on the application of the forefront of research in the field. The current state-of-the-art algorithms and methods come from the fields of ML and DL (Moshayedi et al., 2022; Jiang et al., 2022). The most relevant concepts from both fields will be briefly explained.

3.2 Machine Learning Models

In the core of most modern day ML methods are fundamental concepts and approaches often utilized in computational statistics, mathematical optimization and data mining. In this section a brief overview of some of these concepts that more complex algorithms build upon is provided.

3.2.1 Regression Analysis

Regression analysis consists of several statistical processes used to mathematically represent the dependency between variables and represents a form of supervised learning. These processes are a representation of very simple supervised learning, since the obtained estimation is based on two sets of variables - the dependent variable, or also referred to as outcome, and the independent variable (feature).

The most commonly used process is linear regression. In this approach a set of scalar outcomes are matched to a set of one or more features. If $\boldsymbol{x_i}$ is the vector of features representing y_i, the scalar outcome, then the general representation of this approach is given as:

$$\hat{y}_i = \boldsymbol{x_i}^{\intercal} \boldsymbol{w} \tag{3.1}$$

where $\hat{y}_i$ is the predicted value and $\boldsymbol{w}$ represents coefficients, often also called weights. The vector of features in this equation consists of an additional constant parameter and is given as $(x) = (1, x_0, x_1 ..., x_n)^{\intercal}$. This is necessary for the vector representa-

tion since there is a coefficient w_0 that introduces some disturbance, or noise, to the model.

3.2.2 Clustering and Classification

Clustering is a form of unsupervised learning in which the model attempts to find patterns and groupings of observed data $\boldsymbol{x}$, that has no assigned outcomes $\boldsymbol{y}$. The challenge of clustering comes from the broad way of defining similarities between data points - it can be the distance between data points, the density of a specific area of the domain, repeating intervals or distribution functions. Due to the lack of prior knowledge about the observed data (missing outcomes $\boldsymbol{y}$) the process of clustering is iterative. Multiple varied clustering algorithms can be used, and their parameters modified until the desired observation is reached.

Classification on the other hand has not only a set of observed data $\boldsymbol{x}$, but also a set of outcomes $\boldsymbol{y}$, most commonly referred to as classes or labels in literature. This is similar to regression approaches, but while such methods can predict a continuous outcome, classification approaches have only a discrete set of outcomes. Such problems are done with approaches such as logistic regression. Using logistic regression as an appropriate fit for the weights $\boldsymbol{w}$ to the equation are found:

$$p(x) = \frac{1}{1 + e^{-\boldsymbol{x}\boldsymbol{w}}} \tag{3.2}$$

With the output of a fitted logistic equation, the output would give a form of soft classification between the discreet classes, representing the log-odds of a specific input to belong to an output class.

3.2.3 Gradient Descent

Gradient Descent is a first-order iterative optimization algorithm. It is used to find the local minimum of a function. The function itself needs to fulfill two major requirements: it has to be convex and differentiable. The property of being convex follows a similar logic to the geometrical concept of convex objects, i.e. any segment between two points on the graph of the function is above the graph itself. A differentiable function is one that has a derivative for each point in the function's domain, examples for functions that do not fulfill such a requirement are: $f(x) = \frac{1}{x}$, $f(x) = |x|$, etc.

The gradient is the representation of the slope of the function, for a function with a single variable this is the first derivative, for a function with multiple variables this is the vector of derivatives, one for each variable in the function:

$$\nabla f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{pmatrix} \tag{3.3}$$

The algorithm itself follows a straight forward approach. It first selects a starting point, by selecting values for each variable of the function. Then it evaluates the gradient of it using Equation (3.3). It then scales the gradient value by a predefined variable and subtracts it from the position:

$$\boldsymbol{x_{n+1}} = \boldsymbol{x_n} - \eta \nabla f(\boldsymbol{x_n}) \tag{3.4}$$

where $\boldsymbol{x_n}$ and $\boldsymbol{x_{n+1}}$ are the current point in the algorithm and the next one respectively and η is the scaling factor, referred to as a "learning rate". The algorithm terminates either when the maximum number of steps has been reached, or the scaled gradient value is less than a predefined tolerance value.

These core ML approaches provide an overview how algorithms can extract information and find hidden relations in the data they are trained. In the following section, the DL application of ANN will showcase how it reuses these fundamental algorithms and expands upon them by introducing additional complexity.

3.3 Neural Network Structure

In the core of the current state-of-the-art in DL is the ANN. These networks use existing ML concepts such as classification and regression and expand upon them through the introduction of more complex, and thus "deep", computational paths. Expanding upon this "deeper" computation is the fact that the input variables are no longer observed on their own - they are observed jointly and thus influence the output in a combined way (Russell, 2010). This expands the capabilities of existing models, which up to this point could only represent simple functions and boundaries of the observed domain.

One of the first avenues that were explored at the start of the field of AI was the mimicking of the human brain. Specifically it focused on the concepts of interconnected neurons (McCulloch and Pitts, 1943) - this created the basis for the first ANN. Although the neurons, or more precisely nodes, of such networks were modelled

originally to work similarly to how actual neurons were perceived then, the field of computational neuroscience has created far more complex systems to represent the brain. Nevertheless, this original structure has proven successful even to this day. Although current state-of-the-art research in DL has many variations of an ANN, some core principals and building blocks are consistent throughout - Neurons/Nodes, Layers, and Learning. In the following sections a brief overview of these concepts is presented.

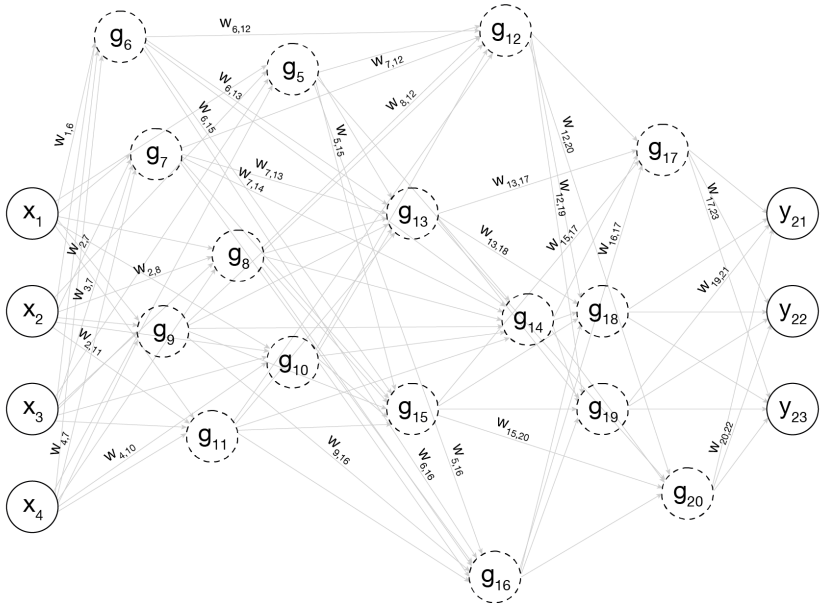


Figure 3.1 Example Node Connectivity. Some weight values left out for clarity.

3.3.1 Nodes

In the core of the modern ANN is the node, often also referred to in literature as a neuron, circuit or perceptrons. Each node can be connected to one or more other nodes and along this connection the values from one node is propagated to further ones. Each connection is further assigned a unique weight which describes the importance of the connectivity between them (Figure 3.1). The goal of each node is then to evaluate its inputs based on their associated weights and propagate the further. This evaluation is performed through the application of predefined function, called an

"activation function". A simple mathematical representation of the concept is given as follows:

$$a_j = g(\sum_{i=0}^{n} w_{ij} a_i) \tag{3.5}$$

where a_j is the output for node j and a_i for node i, $g()$ is the activation function, and w_{ij} is the weight for the connection between i and j. A few well established activation functions are:

- the Sigmoid (Figure 3.2a) function, given as:

$$\sigma(x) = 1/(1 + e^{-x})$$

- the Tanh (Figure 3.2b) function, a scaled and shifted sigmoid function, given as:

$$tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} z$$

- the Rectified Linear Unit (ReLU) (Figure 3.2c) given as:

$$ReLU(x) = max(0, x)$$

- the Leaky Rectified Linear Unit (LeakyReLU) (Figure 3.2d) given as:

$$LeakyReLU(x, a) = \begin{cases} a * x, & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases}$$

- the Softplus (Figure 3.2e) function, a smoother version of the ReLU, given as:

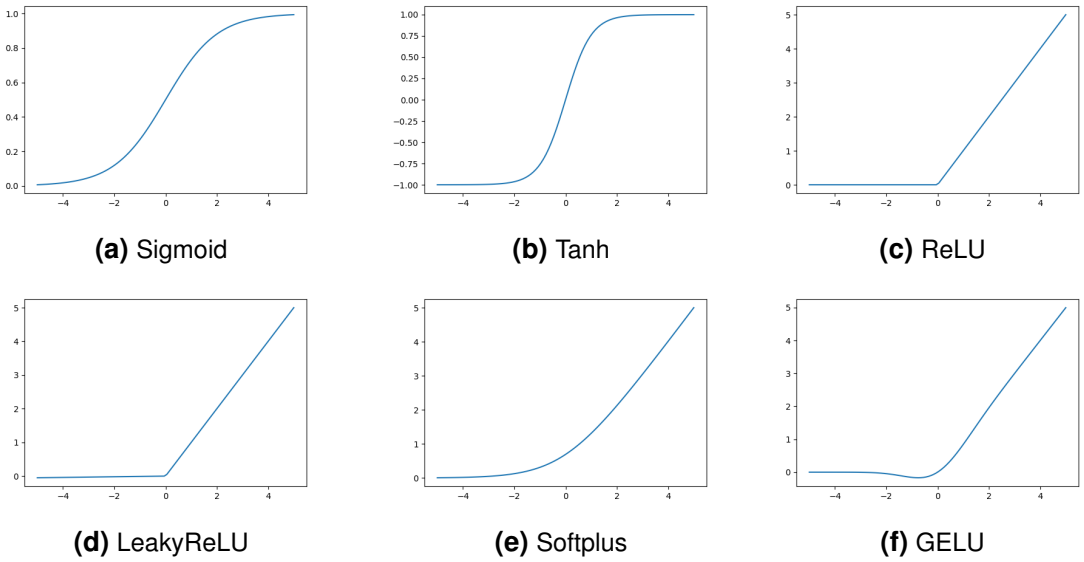
$$softplus(x) = log(1 + e^x)$$

- the Gaussian Error Linear Unit (GELU) (Figure 3.2f) given as an approximation:

$$GELU(x) \approx 0.5x(1 + tanh([\sqrt{(2/\pi)}(x + 0.044715x^3)]))$$

3.3.2 Layers

Assigning the activation function and all connections of each node in large ANNs is a cost inefficient task. An approach to better organize and manage them is to arrange them into layers (Figure 3.3). The advantage of organizing nodes into layers is, that it enables for parameter sharing - all neurons can automatically have their activation functions, connections to other neurons and further parameters set directly through the layer. Furthermore, they allow for easy modularity - allowing for ease of design of such networks. Finally, this approach to abstract nodes into layers allows for a better representation of the information the network learns. Through this hierarchical



(a) Sigmoid **(b)** Tanh **(c)** ReLU

(d) LeakyReLU **(e)** Softplus **(f)** GELU

Figure 3.2 Example of Activation Functions

structure, features and patterns that are learned at each layer and can present how the network is processing the information learned. The layers are typically split into three different categories - input, hidden, and output layers. The input layer processes the original data and provides it to the further hidden layers of the network. Hidden layers are the layers between the input and output layer are called "hidden", because the user does not have direct control over how or what they learn from the data and each other. They transform the original data, provided by the input layer, into different representations on which the output layer can base its final prediction. The amount of hidden layers varies between ANN models and is dependent on the data and task that it solves.

Due to the layer structure and how nodes are connected all weights can be generally represented in a matrix structure, often denoted as $\boldsymbol{W}$, where $\boldsymbol{W^i}$ represents the weights in layer i. If the weight functions per layer are represented similarly as $g^i(x)$ for layer i, then a basic ANN n can be mathematically represented as:

$$n_{\boldsymbol{W}}(\boldsymbol{x}) = g^n(\boldsymbol{W^n} g^{n-1}(\boldsymbol{W^{n-1}} \cdots (\boldsymbol{W^1 x}))) \tag{3.6}$$

3.3.3 Learning

Once the nodes and layers of a ANN are set up, and their weights are initialized, either with random values or with a predefined set, the network is ready to be trained

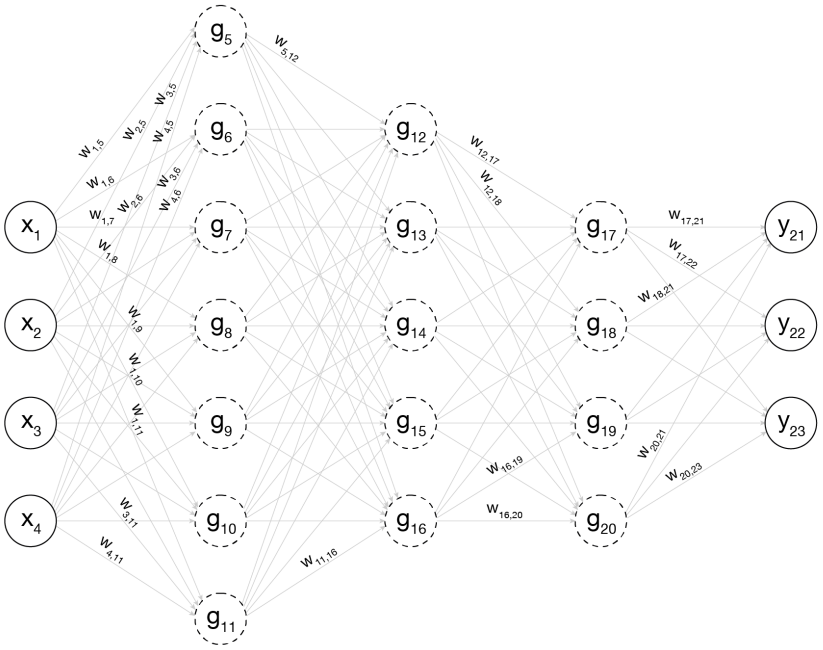


Figure 3.3 Example Layer ordering. Some weight values left out for clarity.

and to learn. The goal of training a ANN is to compute the most ideal weights for the network, so that the differences between the results it outputs, and the ground truth remain minimal. This process happens by providing a data set of $X(\boldsymbol{x}, \boldsymbol{y})$, which consists of a pair of values with $\boldsymbol{x}$ consisting of input vectors and $\boldsymbol{y}$ their evaluation. The ANN takes each input vector in $\boldsymbol{x}$ and produces an evaluation $\hat{\boldsymbol{y}}$ (e.g. regression) or a probability function $p(\boldsymbol{x})$(e.g. classification) that determines what is the likelihood for the prediction to be of a specific output class. A correction, generated based on the differences between the ground truth and the predicted result, is produced. This correction is then in turn used to update the weights of the ANN. This process is repeated a multitude of times until a desired amount of iterations is reached, or the network has achieved a certain termination criteria.

Using a predefined error function the difference between the estimated value, produced by the ANN $\hat{\boldsymbol{y}}$ or $p(\boldsymbol{x})$, and the actual result $\boldsymbol{y}$ of the network can be evaluated. There are a few established loss functions that are well suited to different types of ANNs, e.g. regressions or classifications. The most common are:

- Mean Squared Error (MSE): $E(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N}\sum_{i=0}^{N}(\boldsymbol{y_i} - \hat{\boldsymbol{y}}_{\boldsymbol{i}})^2$
- Mean Squared Logarithmic Error (MSLE) : $E(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N}\sum_{i=0}^{N}(log(\boldsymbol{y_i} + 1) - log(\hat{\boldsymbol{y}}_{\boldsymbol{i}} + 1))^2$

- Mean Absolute Error (MAE): $E(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N} \sum_{i=0}^{N} |\boldsymbol{y_i} - \hat{\boldsymbol{y}}_{\boldsymbol{i}}|$

- Categorical Cross-Entropy: $E(\boldsymbol{y}, p(\boldsymbol{x})) = \sum_{i=0}^{C} \boldsymbol{y_i} log(p(\boldsymbol{x}))$

where N is the total number of evaluated results and C is the number of classes to which the input vectors $\boldsymbol{x}$ can be classified to.

The utilization of error functions to evaluate the predictions of ANNs, an appropriate approach to update the weights of the network is presented in the gradient decent approach. This approach, as discussed in Section 3.2.3, is designed to find the local minimum of a given function, i.e. reduce the value of the error function. Similar to how gradient decent can be used in supervised learning, it can be applied to ANNs to update the weights. For the weights for connections leading to the output layer, the gradient descent approach can be applied directly. For all other layers, the chain rule has to be applied to allow for the use of the gradient descent. The chain rule is given as : $\frac{\partial g(f(x))}{\partial x} = g'(f(x)) \frac{\partial f(x)}{\partial x}$.

The procedure of updating the weights of ANNs through this approach is referred to as backpropagation. It is important to note, that the application of the gradient descent in the case of ANNs also utilizes various improved implementations, building on the concept of the simple relaxation term. A few of the readily available solutions will be briefly outlined and presented.

One of the first proposed improvements is the Adaptive Gradient (Duchi et al., 2011), named AdaGrad by the authors. It consists of learning rates based on each dimension of the input data and as the name suggest they are adaptable. Simply put, the AdaGrad reduces the importance of learning rates based on how often they are used - the more often they are used, the smaller the learning rate is. The learning rate η for every dimension θ_i at step t is given as:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot grad_{t,i} \tag{3.7}$$

where G_t is a diagonal matrix, where each element on the diagonal is the sum of the square of the gradients for that dimension θ_i up to step t, ϵ is a smoothing term, to avoid division by zero and $grad_{t,i}$ is the gradient for dimension θ_i at step t.

Building upon this approach AdaDelta (Zeiler, 2012), also has a learning rate per dimension. But instead of accumulating and adapting the learning rate based on all past updates, like the AdaGrad does, it restricts this to a fixed size based of the last w steps. An improvement to just storing the past w squared gradients, it is proposed

for the sum of gradients to use a decaying average of all past squared gradients. For step t this is given as:

$$E[grad^2]_t = \rho E[grad^2]_{t-1} + (1-\rho) grad_t^2 \tag{3.8}$$

where ρ is the decaying factor. This decaying average is then used to substitute the matrix G_t in eq. (3.7), and thus obtain the final version of AdaDelta as:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{E[grad^2]_t + \epsilon}} \cdot grad_{t,i} \tag{3.9}$$

Finally the Adaptive Moment Estimation (Kingma and Ba, 2015),or Adam for short, furthers the concept of AdaDelta. The algorithm uses a momentum equation to further optimize the convergence of the gradient descent. The algorithm thus stores two values - the decaying averages of the past gradients and the past squared gradients. These equations can be written as follows:

$$\begin{aligned} m_t &= \beta_1 \cdots m_{t-1} + (1-\beta_1) \cdot grad_t \\ v_t &= \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot grad_t^2 \end{aligned} \tag{3.10}$$

where β_1 and β_2 are the decay factors. The authors note that there is an initialization bias inherited to these parameters, but propose a bias-correction equation for both as:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{3.11}$$

Using the bias-correction terms to substitute the matrix G_t and the $grad_{t,i}$ in eq. (3.7) the final version of Adam can be obtained as:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t \tag{3.12}$$

3.3.4 Information Propagation

Based on the direction of connectivity between layers there are two main types of ANNs - the feedforward ANN and the recurrent ANN. For the focus of this dissertation feedforward ANNs are of importance, but both types will be briefly explained and their difference highlighted.

Feedforward ANNs stem from the original single layer perceptron network (McCulloch and Pitts, 1943). In such networks the connections are only in one direction, moving information from the input layer, through the hidden layers and finally to the

output layer. They do not form any cycles or loops inside them. Most modern day approaches apply a feedforward structure as we will see in further chapters.

The Recurrent Neural Network (RNN) concept builds upon the feedforward ANNs in order to be capable of processing input of variable length and were meant to also handle temporal sequences of information. To achieve this goal recurrent ANNs allow for cycles within the network. These cycles usually have a form of temporal delay, that allows for neurons to take as additional input values that they have computed in previous steps. This means that the network has a memory, or an internal state.

3.4 State of the Art Artificial Neural Network Structures

Due to the booming popularity of AI, specifically Deep Learning and the straightforward nature of ANNs, it would be impractical to cover all possible state of the art ANN base structures in this dissertation. That is why a handful of cornerstone structures, with high relevance for this work, are selected and presented. Their unique features and the advantages they bring will be briefly presented and discussed.

3.4.1 Multi-layer Perceptron Neural Networks

The Multi-layer Perceptron (MLP) ANN, often referred to in literature as a feedforward ANN or dense ANN, is the fundamental type of ANN. This type of ANN consists of only densely connected layers. Each neuron from one layer is connected to all neurons from the previous layer as well as to all neurons from the next layer. This structure enables the discovery of global patterns within the provided data. A disadvantage of such an approach is, that for each atomic part of the data, that is provided for training, i.e. for each color channel of a pixel in an image, a separate node in the input layer is required. This leads to prohibitively high amount of neurons per layer, if the data has a high amount of dimensions.

3.4.2 Convolutional Neural Networks

The advent of Convolutional Neural Network (CNN) arose from the challenges of using MLP on images. The dense connectivity of such networks would require the user to provide with a vast number of training images and would require an impractically high computational time and resource to train the network. To solve this issue the CNN

does away with the dense connectivity. Instead, each neuron receives input from only a small set of adjacent data points.

This limitation to the connectivity of the network provides 2 major benefits. On one hand it reduces the amount of connections, and thus weights in the model. Secondly such connectivity means, that only other data points that are adjacent to each other, have influence on the learning process. This approach has proven vastly more efficient with tasks such as image recognition.

Furthermore, such ANNs enforce local spatial invariance, i.e. there are small features that can be recognized throughout the data. To achieve this, each hidden layer enforces, that the weights connecting to a node in it are the same as each other. This transforms the nodes into feature detectors, capable of detecting the same feature throughout the whole data. This further reduces the number of weights in the network.

This creates a pattern of weights, that can be replicated across multiple local regions and is called a kernel. The application of this kernel to the data is called convolution. For a given data object $\boldsymbol{x}$ and kernel $\boldsymbol{k}$ we can represent the convolution as:

$$\boldsymbol{z} = \boldsymbol{x} * \boldsymbol{k} \tag{3.13}$$

where the $*$ is the convolution operator. Simply put the operation uses the dot product between the kernel and a subsection of $\boldsymbol{x}$ focused around a specific data point with the size equal to the kernel size. The kernel is traditionally not applied on every data point in the data, but can utilize an offset, called a stride. This in turn reduces the output of each convolution layer by a factor of $\frac{1}{\prod_{i=1}^{n} s_i}$, where s_i represents the stride in dimension i.

It is possible to add a padding to the data with the size of the kernel in each dimension in order to avoid the shrinking original dimensions of the data. Furthermore, if there are d kernels applied per layer, i.e. detect several features per layer in the data, then the output of the layer is expanded with a further dimension of size d, where each entry in it represents one feature map detected by a specific kernel.

3.4.3 Autoencoder Neural Networks

Autoencoder (AE) ANNs build upon the concept of CNN - using two concepts from image processing, downsampling and upsampling. An AE network consists of two CNN components, one for each process - an encoder and decoder respectively. In the encoder component one or more convolutional layers are used to process the data and extract features. This is then followed by the reduction of the size of the

original dimensions of the data. This is often done by so-called pooling layers - utility layers that, similar to the convolutional layer, apply a predefined, fixed kernel to the data. The process is then applied several times to the data. Shrinking the original dimensions of the data allows for higher amount of features d to be extracted per level. The output of the final level of the encoder CNN is often called in literature the latent (feature) space, i.e. the higher resolution data is downsampled, or encoded, into a more compact representation.

The decoder CNN is the inverse of the encoder network. Convolutional layers are again used to transform the latent space, followed by upsampling the data. After each upsampling, the latent feature space d is reduced in size.

3.4.4 U Shaped Neural Networks

U Shaped Neural Networks (U-Net), were first developed for biomedical image segmentation purposes (Ronneberger et al., 2015). It expands upon the AE by enforcing on one hand the amount of encoder and decoder components to be equal, and on the other introduces a direct connection between each level of the autoencoder. This direct connection is often referred to in literature as a "skip connection". There are two main advantages of using such skip connections. This introduces more connections between different hidden layers, reducing the issue of the "vanishing gradient" problem (Basodi et al., 2020). A further advantage is that the direct connection between encoder and decoder allows for the introduction of higher resolution data in the decoder.

3.4.5 Transformer Neural Networks

Transformer ANNs take a different approach in processing information. Originally used for Natural Language Processing (NLP) tasks (Vaswani et al., 2017), they have since been in used in various other tasks, e.g. image classification, segmentation and regression tasks. Transformers introduced a novel approach in modelling the relation between various parts of a sequence of information - the self-attention mechanism. This mechanism allows modeling global learn the importance between each part of the data, and not just locally, as CNN models do.

Each piece of the data is transformed into three tokens, traditionally called query, key and value. Each query token is the representation of each piece of data, the key token is used to map the relevance between the query token of each other piece of data and itself, and the value token is the representation of the piece of that data from the perspective of the rest of the data sequence. The self-attention mechanism

calculates the attention weights for each piece of data based on its relation to the rest of the sequence, computed through use of the query, key and value tokens. The self-attention layer produces a context specific representation for each part of the data.

Through the use of this self-attention mechanic several advantages can be obtained in comparison to the current models. Due to the structure of self-attention layers, Transformers are capable of capturing long-term dependencies, i.e. it does not focus only on adjacent dependencies, allowing to uncover more complex features in the data. Unlike RNN, which are also capable of capturing such dependencies, the lack of a recursion, allows for them to be more optimally implemented on modern parallel solutions. Transformers also do not modify the dimensionality of the data, in the same way AE networks would, allowing for greater scalability.

3.4.6 Bayesian Neural Networks

Bayesian Neural Network (BNN) build upon traditional ANNs by introducing Bayesian inference. While traditional networks try to find the optimal values for its model, i.e. for the weights, the Bayesian ANN attempts to find the distribution and probability of each weight in the model.

Once a model is trained in traditional ANNs, using it on a new piece of data would always produce the same result. When using a BNN, each weight is sampled for their posterior distribution every time, thus the result would vary every time. This provides the advantage to easily observe how certain the network is in its results - small variance meaning higher confidence, while larger variance indicates uncertainty.

This redefinition of the weights requires an extended approach in handling how the network is trained as well. Stochastic models are used in place for the loss and inference is applied in the training process. Instead of learning and optimizing single parameters, a BNN attempts to obtain the conditional distribution of the weights, given as $p(w|D)$, where D represents the tuple of training data (x, y). This posterior distribution, or posterior for short, can be obtained using the Bayes' theorem:

$$p(w|D) = \frac{p(D|w)p(w)}{\int_w p(D|w')p(w')dw'} \tag{3.14}$$

For this to be achievable, the prior $p(w)$, which represents the original distribution of the weights and their likelihood $p(D|w)$ are needed. The challenge in computing directly the Equation (3.14) is the presence of the integral over all possible w. This requires an approximation of it, which is possible in shallow networks, where the amount of w is small enough. For deeper ANNs there are two main set of approaches that can tackle this issue: sampling based methods, such as the Markov Chain Monte

Carlo method, and variational inference methods. While the sampling based methods are more easy to implement, there are inherit challenges with the sampling of large spaces, such as with deep ANNs.

Utilizing a posterior distribution over the weights, instead of single estimates, allows the expression of predictions from such networks to be represented as probabilistic predictions:

$$p(\hat{y}|D) = \int_w p(\hat{y}|w)p(w|D)dw \tag{3.15}$$

The integral in Equation (3.15) will be approximated as well using one of the aforementioned techniques, due to the impractically large space that it integrates over. Similar to normal ANNs, BNN utilize gradients to optimize their parameters, but rather use the typical gradient descent a stochastic gradient ascent approach is utilized to optimize variational inference function, the Evidence Lower Bound (ELBO) function.

3.4.7 Generative Adversarial Networks

The original concept of a Generative Adversarial Network (GAN) was proposed by Goodfellow et al. (2014) and consists of two competing ANNs in a zero-sum game. Originally these networks were developed as an unsupervised approach for generating realistic imagery. The goal of one of the networks, the generator/generative network is to produce samples from the distribution $P(\boldsymbol{x})$ by learning a mapping between $\boldsymbol{z}$ and $\boldsymbol{x}$. The second network, the discriminator network, attempts to ascertain if the input $\boldsymbol{x}$ that it receives is from the original training set or if it was produced from the generator network. The goal of such networks is then to reach a state, where the generator network can mimic the training data $\boldsymbol{x}$ exactly, and thus also be capable of producing new realistic samples, while in the discriminator network has a probability of assigning it to the right class of 50%.

3.4.8 Diffusion Networks

Diffusion models were first introduced by Sohl-Dickstein et al. (2015). They proposed a form of generative ANN, based on the concept of non-equilibrium thermodynamics. The unique property of such models is, that they introduce a Markov chain model for iteratively introducing noise into the data. The process consists of two key phases - the forward diffusion process and the reverse diffusion process. The forward process introduces in each step additional noise to the data. The original model (Sohl-Dickstein et al., 2015) utilized Gaussian noise to introduce iteratively additional noise into the data. The reverse diffusion process is what the network itself learns. The

objective of the network is to learn how to remove the introduced noise iteratively as well. This means that the network learns how much diffusion happens per singular iteration step and how to inverse that step. Such models have shown improved results in comparison to established GAN solutions (Song and Ermon, 2019; Ho et al., 2020)

3.5 Conclusion

This chapter established the core distinction between the three main, often conflicting terms of AI, ML, DL. The focus of the dissertations with regard to AI is then placed on state-of-the-art DL approaches. The chapter introduces the core concepts of the two main tasks such approaches can solve - regression and classification predictions and establishes a consistent nomenclature for the further chapters with regard to ANNs. A brief overview of the core structure of ANNs is presented and the concept of learning in that regard are explained. Finally, fundamental state-of-the-art concepts, with relevance to the further parts of this dissertation, are presented - such as CNNs,GANs and BNNs.

As can be seen from the chapter, there are no clear optimal solutions with regard to the use of AI. Various implementations of ANN provide a wide range of different advantages, making many such solutions potentially ideal for the goals outlined in Chapter 1, e.g. AE Networks, U-Net, BNN, GAN, Transformers, and Diffusion Models. It is necessary for a detailed literature review of the state-of-the-art implementations of such networks with regard to numerical simulations. The following chapter provides an overview of such solutions, while also highlighting the advances in the field of CFD and CWE.

4 Related Works

Chapter 2 and Chapter 3 presented the theoretical basis of computational fluid dynamics, the base guidelines for application in urban environments and the core concepts of modern deep learning approaches. Based on these pillars the following chapter explores historical and modern applications of these fields. Due to the limitations of how deep learning methods learn from information, a crucial point in the research is the modern day research performed in the area of CWE - the application area for CFD simulations in the Architecture, Engineering and Construction (AEC) field.

Due to the emerging nature of DL, the dissertation will focus not only on specific studies that connect the fields of CFD and CWE with it, but also on a broader range of related research. Through the flexibility in application of DL methods and approaches, as will be demonstrated through the literature review, a large set of them can easily be translated or re-purposed for the goal of predicting wind simulation results. The areas that will be broadly covered in this sense are the application of DL in AEC, CFD and Computational Physics.

4.1 Computational Wind Engineering

The field of CWE has grown steadily over the last decades. It encompasses not only the application of CFD in engineering scenarios, but also the digital modeling and the application of wind-tunnels as a source of evaluation and support. The origins of the field can be traced to the original attempts at creating numerical models for weather prediction (Smagorinsky, 1953, 1958; Charney et al., 1950). From there on the field expanded to smaller and smaller scales, observing and modelling scenarios with obstacles such as buildings and complex terrain (Hirt and Cook, 1972). During the same time frame the results of numerical simulations were compared and validated against ABL wind tunnels for the first time (Derickson and Meroney, 1977), creating a long-standing practice of utilizing wind tunnel models as the baseline of evaluation for newly proposed numerical turbulence models. The field further expanded, focusing on more complex microscale simulations, as well as on the measurement and prediction of 3D pressure and velocity fields around bluff bodies (Baskaran and Stathopoulos,

1989; Paterson and Apelt, 1990). Many challenges, the basis of which were outlined in Chapter 2, were observed for the first time, such as high Re numbers, complexity of modeling turbulence and boundary conditions(Murakami, 1998).

With the increase of computational power, enabling the creation of more complex turbulence models and allowing for higher grid density, the focus has now shifted to the study of various turbulence models, their application based on various conditions and the development of more refined and precise approaches.

This increase, coupled with new software solutions has led to easier to set up CFD simulations. This allows for application to use more complex turbulence solvers, such as implementations of LES, making them more prevalent in research. Reda et al. (2017a) provides a comparative study between an OpenFOAM (OpenFOAM Foundation, 2022) LES solver and scaled physical model used in a wind tunnel. The authors highlight the strong overall agreement between the two approaches, and argue that the discrepancies between them is due to the necessity of much larger Re numbers in the scaled physical model.

Further more complex boundary conditions that match the respective turbulence models are also required. Weerasuriya et al. (2018) proposes an inflow model that is derived from the RANS $k - \epsilon$ turbulence model. They validate their proposal utilizing OpenFOAM, ANSYS Fluent (Ansys, 2022) and a wind tunnel. Both numerical software solutions were incapable of accurately predicting the mean wind speed in the downstream far-field low wind speed zone, but were inconclusive as to why this discrepancy exists. Nevertheless, The OpenFOAM solutions was capable of predicting the same flow fields as the wind tunnel experiment in all other zones, outperforming the ANSYS Fluent software.

A core focus for the CWE field is the simulation of wind comfort for pedestrians. This is done by focusing on the lower parts of the simulation domain, usually up to 10 meters above the ground level, by providing a higher density of cells. Blocken et al. (2016) provides an in-depth analysis and comparison between RANS and LES models, and the various approaches to performing measurements in wind-tunnels. In their overview they provide sufficient proof that wind-tunnel simulations are capable enough of predicting the measurements obtained in the real environment, establishing them as a baseline of comparison for the RANS and LES methods. An alternative to RANS models, Unstable RANS (URANS), also provide potentially better results, but the necessity for higher spatial resolutions makes it almost as expensive as LES models, and thus it is rarely used in case studies. The authors argue that although the faster and more inexpensive versions of wind tunnel measurement techniques and

RANS models have a lower accuracy than other methods, they can still be utilized in pedestrian level wind simulations, and thus are justified to be utilized in research.

Although the RANS $k - \epsilon$ model suffers from such drawbacks, Mochida and Lun (2008) highlights the challenges of applying LES Models in practice. They provide a brief overview of advancements made to this model. Such changes alleviate some of the original issues of the turbulence model, but introduce new challenges and issues that require correction. The authors provide an overview of geometry modeling principles for pedestrian wind comfort simulations for various turbulence models. Their work highlights the necessity of modeling small scale geometries, such as trees and signs, or moving objects, such as cars and pedestrians.

Although small scale geometries are crucial for capturing the turbulence details for pedestrian wind comfort analysis, there is a necessity to reduce geometrical complexity in buildings, where a large amount of small details leads to prohibitively large computational domains. The case study of Liu et al. (2017) selected a dense urban area for a case study, focusing on the importance of details in buildings. Utilizing a RANS RNG $k - \epsilon$ model for their turbulence solver, they performed a parameter optimization study with regard to grid density and wall roughness factor, which is used as a replacement to facade and ground detailing of small geometry. They validated each combination of simulation parameters against real world measurement data. Their study highlighted, that past a certain amount of cell density, the precision does not increase significantly, but that the wall roughness factor plays a crucial role in the accurate representation of the data.

Due to the wide range of use case scenarios for such simulations it is necessary to have an established collection of benchmark scenarios and simulations, with which newly proposed approaches can be compared against. The Architectural Institute of Japan (AIJ) has developed a set of case studies, consisting of single buildings or large city configurations. These cases were simulated in wind tunnels and are meant to serve as benchmarks and comparison tool for numerical simulations (Tominaga et al., 2008). Li et al. (2018) uses them as a base to compare various industry established simulation tools (Autodesk CFD, OpenFOAM, UrbaWind (Fahssis et al., 2010)), with a variety of RANS models and mesh density. The study focuses on the dense urban scenario with one high-rise in the middle. They define points of interest to monitor around the high-rise, based on the wind direction, and where the expected turbulence effect would form. The study concluded, that although OpenFOAM allowed for finer control of the simulation environment, that neither combination of turbulence solver nor mesh density for any of the three solvers was able to sufficiently reproduce the results of the AIJ case study in all wind directions.

The density of meshes and the complexity of geometry are not sufficient parameters for optimization in order to obtain higher precision results. As presented in Chapter 2 the RANS solvers have many constants and parameters, that although having default recommendations, benefit from calibration to the task at hand. Shirzadi et al. (2018) performs a series of wind tunnel validation studies on idealized city layouts. Using these as the base, the $k-\epsilon$ and SST RANS turbulence models were used inside the ANSYS CFX commercial software. Utilizing a linear sensitivity analysis for different closure coefficients, optimal values for the different scenarios were obtained. The case study showed, that with an increase of urban density, the precision of both RANS models significantly decreased. For higher density, the closure coefficient calibration provided drastic improvement on the results, but for highly packed urban areas even further calibrations are recommended by the authors.

Although the parameter optimization of such models is crucial for achieving better results, more complex approaches are required, e.g. leveraging the advantages of multiple solvers. An alternative approach to utilizing only one turbulence solver was explored in the case study of Millar et al. (2020). A RANS-LES hybrid solver is utilized to attempt and maximize the advantages of both solvers. They apply the RANS solver only in the proximity of building facades, which according to their computations amounted for roughly 1% of the simulation domain. Nevertheless, the decrease in cells in comparison to a full LES simulation with wall-resolved model was significant. The use case further refined the cell resolution in the most important regions for the studies - there they were small enough to be able to resolve around 80% of the turbulence. Their results showcased a higher agreement between wind tunnel and real life measurements and the simulation results, although there were some discrepancies in the complex scenarios.

This mixing of numerical solvers has shown that higher order of accuracy can be achieved. Utilizing two different types of solvers - one for fluid simulations and one to capture the specific issue observed, e.g. particle tracking or aerodynamic models, has been developed in the fields of pollutant dispersal analysis and urban wind turbine planning. Bahlali et al. (2018) utilizes a Lagrangian atmospheric dispersion model in conjunction with a traditional RANS $k-\epsilon$ solver. First the CFD solver computes the mean fields of the values of interest (velocity, pressure, temperature, turbulence), and then in a second phase utilizes the chosen dispersion model to predict the spread of pollutants. Their initial validations have shown more accurate results than when utilizing traditional approaches. Balduzzi et al. (2018) similarly utilizes a second type of simplified solver when dealing with the simulation of wind turbines in urban environments. Utilizing a simplified model for wind turbines, based on the Blade Element

Momentum (BEM) theory, a hybrid RANS-BEM solver is constructed. The model utilized in the study expands upon the existing BEM by introducing additional source terms to account for the turbulent kinetic energy and eddy dissipation. The domain is subdivided into two major regions - the BEM region, around the wind turbine, where the BEM equations are solved, and the RANS region for the rest of the simulation domain. And although the standard $k - \epsilon$ RANS turbulence model is utilized, the study proves, that this hybrid approach achieved a much higher accuracy than just utilizing a normal CFD simulation.

As can be seen from this overview of advancements in the field of CFD and specifically CWE, there is still much to be desired of the already complex simulation domains and turbulence models. For this purpose a large amount of research has been done with regard to simulations on large scales in the urban environment focusing directly on CFD/CWE simulations or various representations there of. With the introduction of more complex approaches in solving these issues, such as with multimodel simulations, a natural expansion of this research can be seen in the field of ML and DL. In the following sections, an overview of the various applications of DL and ML techniques in the field of AEC will be presented - highlighting their flexibility, efficiency, and reusability.

4.2 Deep Learning in AEC

Various applications of DL and ML methods and approaches in the field of AEC will be presented. A broader overview of such solutions is explored to showcase a more comprehensive overview of current state-of-the-art solutions. Furthermore, such a wide overview assists in the identification of cross-disciplinary insights, methods, or techniques that are beneficial for the dissertation. This allows for the identification of shared challenges and limitations, that are faced in the application of DL/ML in the field of AEC - such as handling large scale simulations, or the availability and creation of training data.

Air pollution and pollutant dispersal are critical topics in the field of CWE. Such simulations take into account only very roughly the terrain and the actual sources of pollution, such as cars. In Adams and Kanaroglou (2016) a simple ANN model is proposed, that attempts to extrapolate information from a few points in the urban environment a heat map of the air pollution for the whole city. The advantage of the approach is, that each point is populated with greater amount of complex measurements, such as road information, land use and congestion. The different variables were scaled with an additional weight to represent their importance in advance. The

results highlighted the potential in mapping out future information for existing spots and performs better than established models for unobserved areas as well.

A similar approach to providing a complete overview of the Physiologically Equivalent Temperature (PET) in the city of Stuttgart was done (Ketterer and Matzarakis, 2016), comparing two numerical approaches - a Step-Wise Multiple Linear Regression (SMLR) and a small MLP. The data used for these predictions combines a mixture of measured and real data - temperature from a specific day, topology of the city, land cover, and synthetic data - sky view factor and meso- and microscale wind simulations. The study found that the neural network approach was able to capture and represent correlations between parameters better than the SMLR approach, but both approaches were in good agreement with the measured data. The authors conclude that although the approaches provided good results, the complexity in modeling such a phenomenon would require further parameters, but in its current state serves as a guideline for urban planners.

Expanding upon the complexity in modeling large scale phenomenon, the introduction of a temporally dependent data, through the use of RNN, has been attempted for various urban tasks. It has been successfully utilized in predicting water levels for urban flooding (Chang et al., 2014), forecasting the traffic load in urban environments (Xiangxue et al., 2019) or even predicting the changes in urban areas through the use of remote sensing data (Khusni et al., 2020).

A challenge faced with the utilization of early design stages data in analysis and simulation tools is that it is vague and uncertain. Such models need to still be validated through high precision simulations. To offer the planners, designers and engineers a way to receive estimations about their concepts, surrogate models are often utilized. Such models attempt to represent the complex simulations through statistical models, with correlating input and output parameters. Due to the high complexity of such models, traditional statistical approaches struggled with providing sufficient details. With the rise of DL these concepts have been revisited. Vazquez-Canteli et al. (2019) proposes a set of simple MLPs for the prediction of thermal losses and solar gains of an urban area for the purposes of energy simulations. The combined accuracy of the two networks reached approximately 85%, but managed to deliver the results in seconds, which equates to a speed-up by a factor of roughly 2500 times.

Although DL methods allow for the creation of surrogate models for various simulation tasks (Chen et al., 2021; Tarabishy et al., 2020; Tripathy and Bilionis, 2018) such models are often imprecise and carry with them a level of imprecision and uncertainty. Westermann and Evins (2021) attempts to address this issue utilizing BNNs and stochastic variational Gaussian process models to produce surrogate models of

the energy performance of a building. A threshold for the certainty of the probabilistic surrogate model is used to filter out precise, but uncertain models, and use these surrogate models in actual simulations to produce high accuracy results. The hybrid approach of using BNNs to produce accurate and confident surrogate models, that are later used as the base for high precision simulations, provide agreeable results.

A further challenge with the utilization of surrogate models is the parameterization of the model. Depending on the stage of development and the LoD required, such models can significantly increase in input dimensionality. Singaravel et al. (2018) proposes subdividing the domain into different component types and having a dedicated smaller MLP for each of them. This approach is called Component-Based Machine Learning (CBML). This approach allows for easier creation of datasets, since for each component a separate process needs to be utilized. Furthermore, this also allows for the reuse of the different components in later stages. The study achieved reasonable accuracy compared to traditional Building Performance Simulations, but was capable of providing feedback at a fraction of the time, roughly by a factor of 1300.

This approach, in dividing the computational and DL model complexity, is further adopted by Yousif and Bolojan (2021). They focus on applying a GAN to predict the daylight simulation of a room, allowing for the reuse in larger scale scenarios, such as for buildings or even urban environments. The study uses the pix2pix GAN (Isola et al., 2017), that was originally designed for art style transfer tasks. The study produces a data-set of singular rooms, represented through a segmented image. Various objects, such as doors and windows, and their parametric information are represented through various colors. While the rooms equivalent annual daylight simulation results, obtained through the use of established simulation tools, is used as the expected outcome of the network. The accuracy of the trained GAN reached a range of 84% to 96% for the different annual daylight simulation metrics, when compared to the established tools.

The pix2pix GAN has also been applied to larger scales, namely urban simulations of pedestrian wind comfort (Mokhtar et al., 2021). The study focuses on introducing further information into the input of the network, such as topography, vegetation and height. The study also expands upon the approach by introducing a representation of the confidence of the result of the neural network, similar to BNNs, by using a sampling method that utilizes a dropout approach. Although providing acceptable results, around 80% accuracy, the study observed some larger discrepancies in more complex geometrical scenarios.

Through the overview of applications of DL in the AEC field, it can be clearly seen, that it is still in its infancy and there is a vast potential for improving. Even with the

simplistic DL methods, or the adapted more complex approaches, the results are already achieving some of their goals, i.e. vastly faster computational times without large inaccuracies. As can be seen from the state-of-the-art approaches, the strategy of scaling solutions to larger issues has provided acceptable results. Furthermore, the flexibility of DL method has allowed for the reuse of models for various different tasks. Due to these observations the following section, focuses on DL applications and methods in the theoretical applications with emphasis on models optimized for the underlying simulations.

4.3 Deep Learning in Computational Physics and CFD

As shown in Chapter 3, ANNs represent complex non-linear interpolation schemes and are not constrained by linearization. This makes them ideal candidates for solving complex Partial Differential Equations (PDE). In Raissi et al. (2019) explore the concept of differentiating the neural networks based on their input and model parameters. They coin this type of neural networks as Physics-Informed Neural Networks (PINN). The authors propose the use of cost and activation functions that are uniquely tailored to the problem. The authors distinguish between two main approaches - data driven solutions or discoveries of PDEs. The study successfully validates their assumption in the case of continuous and discreet time models for both approaches, using the Schrödinger Equation (Schrödinger, 1926), the Allen-Chan Equation (Allen and Cahn, 1972), the Navier-Stokes Equations and the Korteweg-de Vries Equations (Korteweg and De Vries, 1895) as examples. Although the results of the experiments show great promises, and they utilize only very shallow MLPs, the authors concede that such solutions are not meant to substitute existing numerical approaches, but rather complement them, as that there are many open-ended questions that still need to be addressed.

Although the proposed concept is promising, the technical solution provided by the authors is ill-suited for adaptation and expansion. Building upon the concepts of this study Haghighat and Juanes (2021) proposes a programming library, that builds upon the TensorFlow/Keras (Abadi et al., 2015) library and provides its users with a high level Application Programming Interface (API) for creating complex PINNs with greater ease. The authors demonstrate the proposed solutions capabilities by recreating the solution of Raissi et al. (2019) and achieving comparable results.

Utilizing technical solutions similar to Haghighat and Juanes (2021), Luo et al. (2020) focuses on using PINNs to more precisely estimate the parameters of the RANS $k - \epsilon$ model. The study does not focus on solving the Navier-Stokes equations

or the RANS equations themselves, but rather optimize the constant parameters in the equation to achieve better results. The authors propose a cost function, derived from the equations for k and ϵ respectively. The study focuses on the channel flow with a lower curved wall scenario with two different Re values, for which there are DNS data sets available and are considered as the ground truth. Although the study is successful in reducing the error rate between the PINN calibrated RANS to the DNS solution in comparison to the normal RANS simulation, the difference remains relatively high. The authors argue that due to the simplifications that RANS models make and in order to achieve higher fidelity more accurate models, such as LES, should be utilized.

Expanding upon more complex CFD examples, Laubscher (2021) focuses on introducing further aspects into the simulation such as temperature and species for the purpose of a simple dry air humidification. The study compares differently constructed PINNs - between a singular PINN structure and a multi PINN structure, in which the energy and species have a separate PINN from the fluid variables and the output of all 3 are used in the final predictions and error estimation. As a baseline for the study an equivalent simulation is constructed in OpenFOAM. Both proposed PINNs achieve agreeable results with the ground truth. Using the best performing multi PINN approach, the authors also demonstrated how they can be utilized in surrogate models by expanding input parameters.

Although the use of correctly modeled PINN provide very high accuracy in comparison to their numerical counterparts, such networks are tailored to a specific problem and boundary conditions and are not well suited for reuse in similar scenarios. A more robust approach, with regard to the re-usability, is proposed in Kim et al. (2019). The authors parameterize the input conditions for a set of fluid simulation scenarios in order to reconstruct the full vector fields using a decoder CNN. The authors argue that for precise reconstruction of fluid fields, an accurate loss function is required and propose a modified L_1-Norm utilizing the curl term (Clerk-Maxwell, 1869) for incompressible flows, while dropping it for compressible ones. The architecture of the decoder CNN allows for high precision reconstructions of the learned scenarios, but for a fraction of the time (up to a factor of 700) and are more memory efficient than their numerical counterparts (up to a factor of 1300). The authors also propose a second smaller encoder neural network to handle the compression of increasing parameterization of problems, such as moving inlet sources.

A similar concept was developed by Ribeiro et al. (2020), in which the authors focus on utilizing different set of traditional AE to solve 2D steady laminar flows, by producing accurate pressure and velocity fields. Such flows have an equilibrium state, that is

dependent on the boundary conditions of the simulation domain and the geometry in it. This allows them to parameterize the input of simulation domain similarly as in Kim et al. (2019). Instead of defying crucial points such as inflow positions and velocity the inputs of the CNN is a combination of a vector field of the domain, where each value is computed based on the Signed Distance Function (SDF) from it to a specific surface in the domain, and a second vector field that represents the segmentation of the domain into regions such as inlet, outlet, walls, fluid and obstacles. The authors apply 4 specific types of AE:

- A baseline one
- A U-Net
- A baseline version, but with three separate decoder parts
- A U-Net, but with three separate decoder parts

The authors argue that the advantage of an AE with three separate decoder heads for each of the output components, allows for separate error functions to evaluate the results, in their test cases the L_2-norm was used for the velocity and the L_1 was used for the pressure. The authors achieved relatively high precision compared with the ground truth predictions of the simulation, but with significant increase in speed.

To tackle the issue of having diverging results between the numerical simulation, used as ground truth, and the predictions of a DL method, Obiols-Sales et al. (2020) proposes incorporating neural networks in the simulation process directly. The proposed neural network architecture is a simple AE structure, but rather than receiving a parameterization of the domain as in Ribeiro et al. (2020), the input for the CNN consists of 4 2D Cartesian grids - one for the mean velocity components in each direction, one for the mean kinematic relative pressure and one for the eddy viscosity. In order to inform the network of the geometry and boundary conditions, the study first allows the numerical solver, in the experiment OpenFOAM, to resolve the first K steps of the simulation until the residual error reduces between steps to a certain level. After this the last output of this warm up step is used as input to the CNN in order to predict the converged solution of the simulation, and in order to ensure that the solution is correctly converged, a final set of M iteration steps are performed until the conservation laws are satisfied. The authors report a speed-up by a factor of up to 7.4 for their test cases and an error rate, based on relative mean error, of 0%.

The approach in Obiols-Sales et al. (2020) is akin to more typical surrogate models discussed in this chapter. An inverse approach, inspired from recent advancements in the field of computer graphics with upsampling of imagery (Dong et al., 2016), is

explored in Bai et al. (2021). The goal of the research is to train a neural network capable of correctly upsampling coarser turbulence based CFD simulations. The authors base and expand upon their original concept - that turbulence observed only on small patches of the whole domain are simple enough for a neural network to learn efficiently. They utilize and improved version of their previous work, a dictionary-based CNN, that learns to reconstruct high precision details of turbulent flow in the whole domain through observation of coarse details in specific patches of the domain. Simply put the approach up-scales the coarse domain with a trilinear interpolation, but with low frequency, and uses the dictionary elements of the CNN as high frequency residuals to augment the result. The results are compared predominantly visually, where they achieve a high overlap with established baselines and also achieve a significant increase in speed-up to a factor of 7. The authors also highlight the capabilities of the approach to successfully upscale coarse simulations on domains and scenarios, that were not part of the training dataset.

With a similar task Xie et al. (2018) proposes a different approach in reconstructing high precision CFD simulations from coarser domains. The authors utilize a GAN structure in order to correctly reconstruct the spatial and temporal aspects of the domain. The study uses randomized 2D smoke simulations and the input of the generator is a 2D image map of the down-sampled domain with 4 channels - pressure, velocity and vorticity. The GAN utilizes two separate discriminators - one for the spacial aspects of the generated up-sampled results, and one for the temporal aspects. The authors showcase, similar to Bai et al. (2021) that the network can successfully visually reproduce the higher precision CFD simulation from down-sampled examples, capturing the spatial and temporal details.

One important aspect that has been briefly touched upon in Xie et al. (2018) and Bai et al. (2021) is the temporal aspect of CFD simulations. Simple examples such as the Kármán Vortex Street (Von Kármán, 2004) show that not all flows are convergent, and thus for a DL method to be capable of predicting their outcomes a temporal aspect needs to be introduced. Hasegawa et al. (2020) proposes a two-step approach in predicting the future time-step results of a CFD simulation. The concept consists of two neural networks working intermixed to achieve the ability to predict the future steps of the simulation - a typical AE and a Long Short-Term Memory (LSTM) network. The simulation scenario used in this study is the 2D flow in a channel around a bluff body, producing the aforementioned Kármán Vortex Street effect. The autoencoder is trained on DNS of the scenario and the LSTM is trained on the latent space output of the encoder in the CNN, once it has been trained. The LSTM is then capable of predicting the future latent space of the simulation domain, which is in turn reconstructed

from the already trained decoder component of the CNN. The authors show that the results achieve a high agreement with the DNS results used as a baseline, and argue that this methodology can be expanded for other scenarios.

4.4 Conclusion

As can be seen from the literature review into current state-of-the-art applications of CWE, surrogate models in AEC and DL approaches in the field of Computational Physics and CFD, there are many varied approaches in tackling a multitude of issues inherent to the complexity of fluid flow. Nevertheless, this chapter provides an overview of solution that are capable of addressing them.

It is clear from this overview, that there is no singular optimal approach for performing urban scale wind simulations. The complex geometrical scenarios and layouts, the focus of the simulation (heat islands, pedestrian wind comfort, pollutant dispersal, etc.) contribute to this issue further. Studies agree upon the fact, that even simpler turbulence models, such as RANS $k - \epsilon$ are sufficient estimates.

Furthermore, this overview also showcases, that there is no exact way to model an ANN to predict the outcome of a simulation. Key aspects for achieving better results can be obtained from this overview. It is of crucial importance that the geometry and the boundary conditions of the problem are modelled correctly. Furthermore, utilizing auxiliary information, e.g. in the case of CFD not only velocity, but also pressure, vorticity, turbulence model parameters, provide a vast improvement to the results of the ANN.

Based on these observations, the following chapter provides criteria and more detailed analysis on the requirements for the use of a DL method in the context of urban scale wind simulation. An overview of current solutions with regard to these points will be also discussed and analyzed.

5 Deficit Analysis

Chapter 1 outlined the requirements for the integration of simulation and analysis tools as DDS in the early design stages of urban planning. Chapter 4 presented the current state-of-the-art solutions in CFD/CWE and advancements made in DL with regard to the creation of more efficient surrogate models of urban scale simulations and accurate and reliable prediction of results. Integrating the knowledge presented in Chapters 2 and 3 and building upon the works presented in Chapter 4, the evaluation criteria (defined in Section 1.2) are utilized to analyze existing tools for urban wind simulations. Building upon the deficits observed from this analysis, concrete additional requirements are defined. They will be used to evaluate existing DL solutions for CFD simulations. The analysis of these solutions provides the necessary outlines for the concept definition in Chapter 6.

5.1 CWE Solution Analysis

Due to the wide range of solutions to solving the Navier-Stokes Equations for steady-state incompressible turbulent flow a plethora of numerical solvers and implementations exist. In this section the core criteria for evaluation of such solutions for the purpose of DDS are restated and brought into the specific context of urban scale wind simulations. With these concrete requirements, established simulation solutions are analyzed and evaluated. The underling issues with such solvers are highlighted.

5.1.1 Criteria

Drawing upon the criteria defined in Section 1.2 the requirements for such tools can be grouped into the following parameters:

- **Reliability and Precision** - Section 4.1 showcased that there are no numerical solutions that can consistently and reliably provide exact solutions. For a software solution to be considered reliable and precise it must provide results of established validation cases, such as those of (Tominaga et al., 2008; Franke et al., 2007; The Langley Research Center Turbulence Modeling Resource, 2022).

- **Flexibility and Adaptability** - Building upon the need for a numerical solution to provide reliable and precise results is the ability to more easily modify the domain parameters or the boundary conditions of the simulation in comparison to wind tunnel experiments or real-life measurements. Furthermore, such software solutions must have the capability of being interrupted, i.e. they must allow for the user to change dynamically during the simulation process the aforementioned parameters.

- **User-Friendliness** - As discussed in Chapter 2 the theoretical base for correctly understanding and configuring the boundary and initial conditions of a CFD simulations are complex. This is further compounded by the wide rage of use case scenarios in which such simulations can be applied in an urban context, as presented in Section 4.1. Correct turbulence models need to be selected (Li et al., 2018), their constant parameters optimally selected for the scenario, simulation mesh density needs to be defined as well (Luo et al., 2020) as all boundary conditions (Weerasuriya et al., 2018) and special patches, such as vegetation (Lin et al., 2008), need to be correctly modelled. Such expertise in CWE cannot be expected or mandated from urban planners and architects. Thus ease of use measures the knowledge required for the user to use the tool and set up all boundary and initial conditions.

- **Interoperability** - The complex nature of CFD simulation necessitates that such software solutions traditionally come as independent software solutions. Their ability to be integrated into existing urban planning tools as DDS systems is crucial.

- **Responsiveness** - The simulation tools need to be capable of reacting to the users input in an ad-hoc way. This means they need to be capable of providing results, even if they are intermediate. As has been shown in (Bratoev et al., 2018; Förster et al., 2021) complex numerical simulations, capable of providing such information, and not just the final results are greatly beneficial for the early stages of urban planning.

The refined criteria from Section 1.2 are the basis on which the software solutions are evaluated and analyzed for their discrepancies.

5.1.2 Analysis

Table 5.1 provides an overview of the most established software solutions available for urban planning. In this section a more detailed analysis of each one of them will be provided. The grading scale is between "very well" (++) to "impractical"(--).

OpenFOAM

There are two official distributions of the OpenFOAM (OpenFOAM Foundation, 2022) - one is from the ESI Group and one is the OpenFOAM Foundation. Both versions of the software are based on the same open-source distribution but have differing release cycles and numbering conventions for their versions, thus they differ at most in some of the more advance features. As can be seen from Section 4.1 a large majority of user studies utilize a version of OpenFOAM. The setup for a simulation consists of the creation of several folders that hold various settings for said simulation. This setup requires solid knowledge in the field of CWE, since everything from the meshing and definition of the domain, through the setup of all initial and boundary conditions, to the selection of appropriate solvers for pressure, velocity, and turbulence need to be defined by the user. A strong advantage of this separated approach is that it allows smaller components to be easily changed (such as geometry) without having to redefine the major components such as solvers and boundary conditions. The speed at which results are outputted is heavily dependent on the amount of cells in the domain and the hardware limitations of the system. Due to its robust nature and steady development over the past decade, the software has been extensively validated. The software allows for intermediate results to be stored at the end of each time step they are needed at, but are nevertheless constrained by the computational time of the solver itself. Due to its established accuracy and robust setup schemes, a vast majority of further software solutions have been developed. They provide the user with simplified interfaces or predefined scenarios (Kastner and Dogan, 2021; FSD blueCAPE Lda, 2022).

Ansys Fluent

Ansys Fluent (Ansys, 2022) is a commercial CFD solver, developed and distributed by Ansys. It offers an all-in-one package for performing large scale CFD simulations. The software solution provides the user with a detailed UI for the set-up of any type of laminar or turbulent simulation. The vast majority of parameters have predefined default values, but tailoring the simulation domain to the exact scenario requires solid knowledge in the field of CWE, similar to OpenFOAM. The speed of computation

is also heavily dependent on the simulation domain and the hardware limitations of the system. Unlike the OpenFOAM solution, it provides the option to utilize not only Central Processing Units (CPU) but also Graphical Processing Units (GPU), which provides significant increase in computational speed. It does not however, provide any intermediate results to the user. Due to the way Fluent is set up, a new simulation needs to be defined from the grounds up. Though the user can export major component setups, such as solvers and inlet/outlet profiles, to templates for reuse. The software has been successfully validated against established use cases.

DesignBuilder

Similar to Ansys Fluent, DesignBuilder (DesignBuilder Software Limited, 2022) offers the user an all-in-one package for performing various numerical simulations and analysis for buildings. It has a dedicated focus geared towards engineers, architects, and planners. The simulations covered by the tool include energy analysis of buildings, daylight, design of Heating, Ventilation and Air Conditioning (HVAC) systems and CFD. The tool offers very detailed control over the simulation initial and boundary conditions, suggesting also useful default values. The software package also offers a plug-in for Revit, an established tool for designing buildings, allowing the user to utilize the full potential of the tool in a familiar environment. The tool has a predominant focus for single building scenarios (interior and exterior), and thus does not offer a full range of validations for more generalized cases.

Online Tools

While OpenFOAM and Ansys Fluent offer desktop solutions that could be utilized at any point by the user, a greater portion of CFD and CWE software solutions offer their services through online tools. In this way the software can outsource the heavy computations to external servers, with hardware capable of performing full simulations minutes. Examples for such established tools are SimScale (SimScale GmbH, 2022), IES (McLean, 2022) and UrbaWind (Fahssis et al., 2010). While SimScale and UrbaWind offer simplified interfaces to the user with which they can set up their whole simulations, IES offers consultation services with experts in the respective fields. Although such tools greatly simplify the use of complex CWE and CFD simulations, they are difficult to integrate into the early stages design process. Each of the aforementioned software solutions offer a large array of industry established validation scenarios.

Table 5.1 CFD Software Comparison

Software	Reliability/Precision	Flexibility/Adaptability	User-Friendliness	Interoperability	Responsiveness
OpenFOAM	++	+	–	o	+
Ansys Fluent	++	-	-	-	-
SimScale	++	+	++	–	+
IES	++	–	++	++	–
DesignBuilder	-	–	++	–	+
UrbaWind	++	-	++	–	+

5.1.3 Findings

As can be seen from the analysis of existing CFD/CWE software solutions, there are no currently existing implementations that fulfill all the criteria required for their integration as a DDS in early stages of design. The main issue that such products face is the need to balance between opposing requirements - a responsive and user-friendly design, and a greater flexibility in the set-up of the simulation. Although online solutions such as SimScale and UrbaWind utilize the advantages that come with such approaches, i.e. faster computation times, the lack of APIs for their integration in existing workflows introduces a negative disruption to the design process, typical for the early stages of urban planning. Based on this analysis clear goals for the DL solutions will be defined in the following section.

5.2 DL Solution Analysis

The prerequisites for the usability of DL solutions build upon the CFD requirements. Since the goal of the DL methods is to bridge the gap between the limitations that state-of-the-art solutions in CWE/CFD possess, the core criteria, from Section 1.2 can be limited to responsiveness, reliability and precision, adaptability and flexibility. Due to the nature of modern DL solutions, i.e. the trained models can be reused with ease (Hugging Face, 2023; Foundation, 2023), the criteria for interoperability is not taken into consideration. Based on this the criteria for user-friendliness is also not taken into consideration, due to the fact that such solutions do not provide GUIs and have a standard API approach.

5.2.1 Criteria

Due to the goal of the DL model to alleviate the challenges that are faced by CWE/CFD solutions, the criteria require a separate context specific definition, different from the one outlined in Section 5.1.1:

- **Responsiveness** - As was seen from Section 5.1.2 the responsiveness issue is rooted in the high computational times. To evaluate the responsiveness of a DL method the potential speed-up, in comparison to the established approach will be taken into consideration (Xie et al., 2018; Bai et al., 2021; Obiols-Sales et al., 2020).
- **Precision** - The CWE software solutions have a large set of validation schemes, that ensure the implemented numerical solvers are sufficiently precise and ac-

curate. Due to the nature of DL methods (i.e. learning from the provided data) precision is measured by the deviation between the predicted outcome and the ground truth.

- **Reliability** - Not all CFD simulations reach an equilibrium state. It is crucial for the reliability of DL methods to be capable of predicting not just the final state of the simulation but also how the simulation develops over time.

- **Adaptability and Flexibility** - DL methods are constrained by their training data. It defines what the input requirements and what the expected outcome is. As Sections 4.2 and 4.3 showed, there is a vast way of representing CFD simulations. This would greatly affect the flexibility of such solutions. Methods with more constrained data requirements are better fine-tuned to a specific scenario, but can be integrated in fewer scenarios. The adaptability and flexibility of DL methods is then evaluated based on the requirements they enforce on the data.

These refined criteria, based on the ones defined in Section 1.2 are the base on which the state-of-the-art CFD-DL solutions analyzed.

5.2.2 Analysis

Table 5.2 provides an overview of the evaluation of current state-of-the-art approaches in DL that have successfully been utilized in the prediction of CFD simulations, regardless of the problem domain. The table represents responsiveness through a speed-up factor and the precision through the minimum error rate between prediction and ground truth.

DeepFluid

Utilizing a parametrical model of CFD simulations, Kim et al. (2019) explore the feasibility of decoder CNN in outputting predictions of the vector fields of observed domains for a specific time step, based on only a handful of domain specific parameters (such as inlet position and size, and time step). The authors propose a loss function tailored to the specific scenarios, for which the CNN is being trained. Although the study is capable of parameterizing more complex simulation scenarios, such as with moving inlet sources, using an encoder to reduce the ground truth results into latent space vectors, there are challenges for its scalability. Obstacles in the simulation domain need to be parameterized as well, making it impossible to apply the network on unobserved constellations of geometry. Nevertheless, the network shows promising results with a significant speed-up effect and a high accuracy.

U-Net

The Neural Network structure, proposed by Thuerey et al. (2020), explored the application of a simple U-Net structure in the prediction of the equilibrium state of flow around an airfoil. The network is trained on input-output data pairs consisting of the fluid mask of the domain, where solids and fluid are, and freestream values for the domain for the input and the equilibrium values for pressure and velocity as output. The network uses an unmodified L_1 norm for the training process and achieves highly accurate predictions. Due to the structure of the data, the network is not capable of providing temporal information about the flow, but nevertheless achieves a relatively high accuracy and a large speed-up factor.

DeepCFD

Building further on the concept of a U-Nets capability of predicting laminar flow, Ribeiro et al. (2020), expands the concept of the decoder further. The authors propose that a dedicated decoder for each vector field value can provide a higher accuracy than one unified decoder. The ability to assign different error functions, in this case MSE for velocity and MAE for pressure, provides the network with more precise correction in the learning process. The study uses 4 networks to compare their approach, with a basic AE as a baseline, and the results obtained with the U-Net with three decoders, named DeepCFD by the authors, achieves a higher accuracy compared to the ground truth. In terms of speed-up, the authors note that if the network is used on a GPU the resulting speed-up is significant. Through the introduction of more varied scenarios the authors argue, that a more generalized and scalable solution is possible, nevertheless the study focuses on steady laminar flow and the output of DeepCFD is only the equilibrium state of the simulation domain.

CFDNet

Further expanding upon the application of the concept of U-Nets, Obiols-Sales et al. (2020) uses it as a surrogate model to improve the computation speed of the simulation. The authors substitute the main bulk of the numerical computation of the CFD simulation with results of the U-Net. They utilize a finite number of steps to initialize the simulation domain, in this way removing the domain knowledge from the network, and applying additional simulation steps to smooth out the results of the network. In this way the authors achieve results that are indistinguishable from the ground-truth, while still achieving a speed-up in comparison to a direct numerical simulation. Although the network does not receive any direct information about the domain, the

Table 5.2 DL Method Comparison

Network	Responsiveness	Precision	Reliability	Adaptability / Flexibility
U-Net	1000x	2.32%	-	o
DeepFluid	716x	4%	+	-
DeepCFD	51,400x	2%	-	+
CFDNet	7.4x	0%	-	+
PINN	instantaneous	2% Temperature 4.6% Velocity 6.1% Pressure	+	-
tempoGAN	8x	n/a	+	+
pix2pix	instantaneous	20%	-	+
AE-LSTM	n/a	0.01%	+	+

boundary, and initial conditions, using the initial simulation results of the CFD solver, allow for potential scalability of the problem.

PINN

Building on the original concept of PINNs, Laubscher and Rousseau (2021) focuses on the applicability of such networks for the prediction of 2D convective flow. The authors propose error functions for the momentum and energy residual, the boundary conditions, and the initial conditions. The authors achieve very precise results, noting that deeper versions of their proposed network architecture achieves significantly better results. As was pointed out in Section 4.3, and as it is also acknowledged by the authors, the PINN is tailored to specific problems, and thus cannot be extrapolated to unobserved problems.

tempoGAN

As discussed in Section 4.3, the proposed multi-discriminator based GAN, tempoGAN (Xie et al., 2018), upscales lower resolution turbulent simulations, without losing a large amount of detail. The approach evaluated the results only visually, and thus it is difficult to assess the numerical accuracy of the approach, but due to the 2 discriminators - one for spatial and one for temporal discrimination, the approach is capable of outputting multiple steps of a simulation. tempoGAN is trained on vector fields with the use of physics aware error functions, and is thus well suited for its reuse in unobserved scenarios, as long as the constraints of the training data are observed.

Pix2pix

Utilizing a more generalized GAN, Mokhtar et al. (2021) focuses on the applicability of neural networks in the prediction of pedestrian comfort in regard to wind speed. Although the authors achieve a significant speed-up in comparison to established approaches, the accuracy is drastically reduced. The results are nevertheless still useful, due to the fact that the absolute deviation from the ground truth is still low enough, and can provide the users with sufficient information in their design process.

AE-LSTM

Building upon the concept of AE and LSTM, Hasegawa et al. (2020) proposes a two tier neural network, that combines the capabilities of an AE to extract latent space feature vectors from a flow field in the encoder. The decoder then transforms the

latent space into the original flow field. This is combined with the ability of LSTM to learn temporal relations between the latent space of data points. Although the authors focus on only one scenario - flow in a channel around a bluff body, the network is capable of correctly predicting the flow around different geometries, that were not learned explicitly by the network.

5.2.3 Findings

The overview in Table 5.2 and the performed analysis highlights that there are only a few DL solutions that are capable of bridging the issues, that CFD solvers are facing. From this overview it is clear that DL solutions, that provide the highest precision are the ones utilizing an error function tailored to the simulation itself, i.e. physics informed. Similarly, the reliability of such solutions is not dependent on the DL model itself, but rather on how the data is structured as well as the training process. The importance of how the data is structured is also visible in the adaptability of such solutions - models that incorporate the domain, boundary conditions and geometries are capable of correctly predicting outcomes on unobserved scenarios.

5.3 Conclusion

As can be seen from the analysis of CFD Simulation tools (Section 5.1) the biggest trade off is between the speed of results and the ability to integrate the tool in other environments. As explained in Chapter 2 the biggest reason for the high computation times are, because urban wind simulations have larger than normal domains and require a high amount of cells in order to capture the amount of details required for sufficient analysis. The challenge of integrating such tools is compounded by the fact, that based on the scenario being simulated, pedestrian wind comfort, heat island analysis, structural integrity, etc. and the complexity of the geometry there is a vast amount of simulation parameters that need to be defined, from boundary and initial conditions, to meshing of the domain and defining turbulence models and solvers. And while there have been strides to bridge the gap of usability (SimScale GmbH, 2022; McLean, 2022; Fahssis et al., 2010) or their integration into existing established planning and engineering tools (Kastner and Dogan, 2021; DesignBuilder Software Limited, 2022) there is no solution currently available that can address all the aforementioned issues.

Section 5.2 showed that DL methods offer a way to simplify this issue. Such methods can be trained for specific scenarios, or be trained on more broad cases. The cur-

rent research has been predominantly focused on the feasibility of such approaches in predicting accurate results, in comparison to their ground truth counterparts. Such studies have shown a high level of accuracy and a significant speed-up in comparison with established tools. Nevertheless, it must be highlighted that such research has been done on simpler use cases, such as 2D flows around bluff objects, where steady laminar flow was predominantly observed. Such scenarios do not include the complexity added by turbulence, and while there is research present in this field as well (Bai et al., 2021; Xie et al., 2018; Mokhtar et al., 2021), the scope of such experiments is still relatively small and the results are inconclusive.

With the analysis of present CFD tools and advances in the field of DL, a clear research focus, not yet sufficiently explored in current literature is highlighted. The application of physics-informed DL methods to predict large scale CFD simulations in order to obtain highly accurate results, in comparison to established tools, while being capable of providing them reliably.

6 Concept

The analysis of Chapter 5 showed that there is no current approach to integrate CFD simulations in the early stages of design that fulfills all the requirements defined in Chapter 1 - reliability, flexibility and adaptability, user-friendliness, interoperability and responsiveness. The evaluation of DL-CFD methods highlighted that the current state-of-the-art solutions, although focused on smaller scales and simpler use cases, have the potential of bridging the underlying challenges for the integration of CFD simulations as DDS in the early stages of design. This Chapter outlines a proposed solution for the creation of a surrogate model on the basis of a novel DL approach, based on recent advancements in the field. This dissertation proposes the utilization of a Diffusion ANN - **DiffusioFlow** - to address the outstanding challenges that have been observed through the analysis in Section 5.2. This approach will take into consideration not only the requirements outlined in Chapter 5, but will also address the challenges of creating a working DL surrogate model for simulations from the very beginning.

6.1 Concept Outline

The concept outlines the basis of a digital pipeline, with which a user is capable of defining all the parameters for producing a trained neural network model, capable of predicting simulation results in unobserved use cases of the same type. As discussed in Chapter 5, such a pipeline needs to fulfill the requirement of 'User-Friendliness' and thus a level of abstraction of more complex concepts needs to be made. The pipeline needs to cover not only the process of selecting and training an appropriate ANN architecture, with all accompanying parameters, but also offer a way to generate the necessary training data with regard to the exact type of use case the user needs. The concept can thus be separated into three main components:

- The **Domain** of the use case
- The **Scenario** for the Domain
- The **ANN** itself

The proposed pipeline, the interaction between separate components and the expected output from each of them can be seen in Figure 6.1.

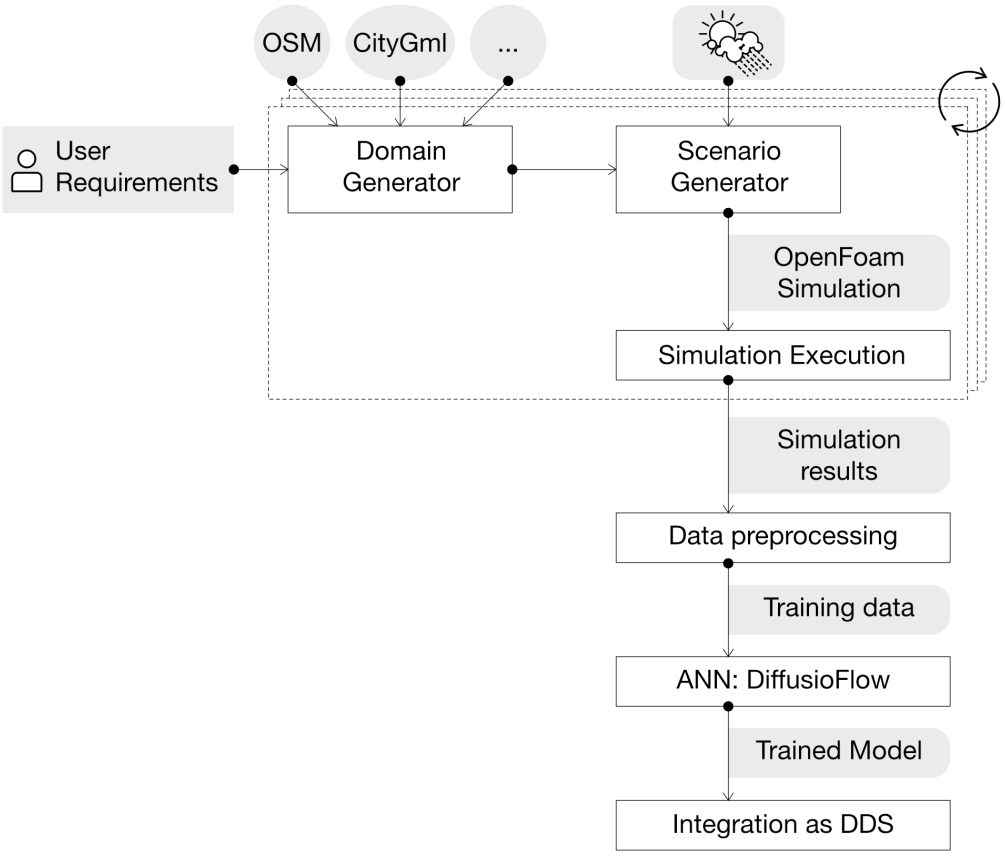


Figure 6.1 Overview of proposed pipeline

6.1.1 Domain

The definition of domain in the scope of this concept stands for not only the physical space that will be used by a simulation, and later server as the base upon which an ANN will perform its predictions. But it also encompasses the level of refinement of the core geometry and the use of any optional further components (see Section 2.4.2).

As has been already shown, CFD simulations can vary greatly in scope (Figure 2.4). This issue is further compounded by the simulation requirements set out in guidelines (Franke et al., 2007; Tominaga et al., 2008), in which it is stated, that for a fully formed flow to be observed, the given domain needs to be expanded further by specific minimum amount in each direction, based on the maximum height of buildings in the domain. The requirements must be automatically taken into consideration by the

pipeline, and the user needs to only specify a handful of simplified constraints with which the pipeline can derive the boundaries of the domain.

Furthermore, the aforementioned complexities of different geometries in the domain need to be addressed by the pipeline. The options for LoD and additional geometries, such as terrain and green spaces and vegetation, needs to be not only presented in an understandable fashion, but they need to be also internally constrained by the scope of domain itself - small details on facades or singularly modeled trees are not feasible to model in a simulation domain that is several square kilometers large. The pipeline needs to be capable of deriving such restrictions based on the domain size automatically.

6.1.2 Scenario

The Scenario defines how the domain is utilized - the different use case scenarios for CWE analysis may impose further restrictions on the simulation parameters, solvers used or the grid refinement.

Based on the selected type of Scenario, the pipeline must be able to introduce modifications to the results of the Domain. The Scenario can introduce more refinement requirements in areas of potential interest, e.g. for denser grids on the lower levels of the domain for pedestrian wind simulations. The areas of interest are further extracted from the Scenario - cross-sections for wind load on high rise buildings or 2D horizontal planes for comfort and wind tunnel analysis. These relevant subspaces are the base on which the ANN is further trained upon. These areas need to be automatically defined based on the underlying Scenario and Domain restrictions. Based only on a handful of parameters, all other requirements are extracted.

Furthermore, based on the Scenario, appropriate accurate turbulence solver needs to be selected. As can be seen from the literature review in Chapter 4 and the overview, done in Toparlar et al. (2017b), there are no clear-cut choices for turbulence solvers that perform consistently better than others - this needs to be addressed by the pipeline, by having an appropriate match for the different solvers to specific scenarios, and domains.

As has been discussed in Liu et al. (2017) grid density plays a pivotal role in the accuracy of the simulation results. An approach that finds an optimal balance of grid density, based on the scenario and domain size will be necessary.

6.1.3 DiffusioFlow

Due to the still rapidly developing field of Deep Learning CFD solutions, there is no clear optimal model to solve all the defined requirements in Chapter 5. From the definition of these requirements, the approach of using a Diffusion based ANN is appropriate. Such models are inspired by the thermodynamic process of diffusion and have a direct timestep integration. Originally used for the generative task of producing imagery from noise samples, such ANNs can be adjusted to predict the flow of a CFD simulation. Furthermore, the Diffusion ANN needs to not only be aware of the temporal aspect of a fluid simulation, but also about the geometry within the domain.

6.2 Pipeline Requirements

For the actual implementation of a prototypical pipeline, based on the outlined concept, concrete system requirements must be extracted for it. The presented pipeline can be divided into two major components, crucial for its performance - the data generation process and the ANN itself. Only through the correct fulfillment of the requirements, outlined in the following section, is it possible for the pipeline to be implemented.

Data Generation

The Data Generation components encapsulate the whole process of producing training data for the ANN component and consists of the following major requirements.

- Filtering of locations based on simulation requirements
- Automatic reconstruction of 3D geometry
- Automatic set-up of simulation parameters and environment

ANN - DiffusioFlow

The ANN component consists not only of the training of the network, but also involves the use of the final trained model and the processing of the output from the simulation.

- Execution and preprocessing of simulation data
- Adapting depth and complexity based on the Domain / Scenario
- Sufficient speed-up

The following sections will describe in detail how such requirements can be implemented in practice. The related prototypes have been developed through either student projects or research work. Due to the prototypical nature of the solution, a focus on a singular type of simulation scenario is made - specifically the pedestrian wind comfort analysis.

6.3 Data Generation

In order for the dissertation to explore the validity of the proposed ANN, a corresponding data generation process that can produce a consistent stream of information that can be utilized in the training process is required. A limitation of existing ANNs observed in Chapter 4 is that the simulation relevant data must be in a structured N dimensional layout, with a predefined data scheme, e.g. imagery with the same resolution and channels. This means that the Data Generation component needs to extract from the user requirements for the simulation appropriate consistent dimension sizes. Furthermore, the tool should not require from the user extensive knowledge in the simulation field domain, but rather extract the various Scenario and Domain parameters from context. The Data Generation component consists of two subcomponents, which are executed one after another - first the **Domain Generator** extracts the boundary conditions for the simulation and the geometry, while the **Scenario Generator** sets up the simulation parameters, executes the simulation and prepares the data for use by the ANN.

6.3.1 Domain Generator

The **Domain Generator** is a GUI based application that provides the user with a wide set of simplified options with which to generate a large amount of Domain Solutions, that share the same boundary conditions. Figure 6.2 provides an overview of the prototypical implementation.

In this application the user can select their data source of choice, e.g. CityGML (Kolbe, 2009) or OpenStreetMap (OpenStreetMap contributors, 2022). The options provided to the user are further split into two sections - one focused on the geometry and one on the boundary conditions of the simulation domain. Based on these required parameters, an additional set of inferred parameters are extracted on their basis. The GUI provides a large overview of areas matching the defined parameters is shown to the user as well. Once the user has defined their requirements through the given parameters they can execute the extraction of random areas that match their

criteria. The output of the application consists of two separate entities per area - the geometry, stored as a geometry file and an options file, containing simulation relevant parameters.

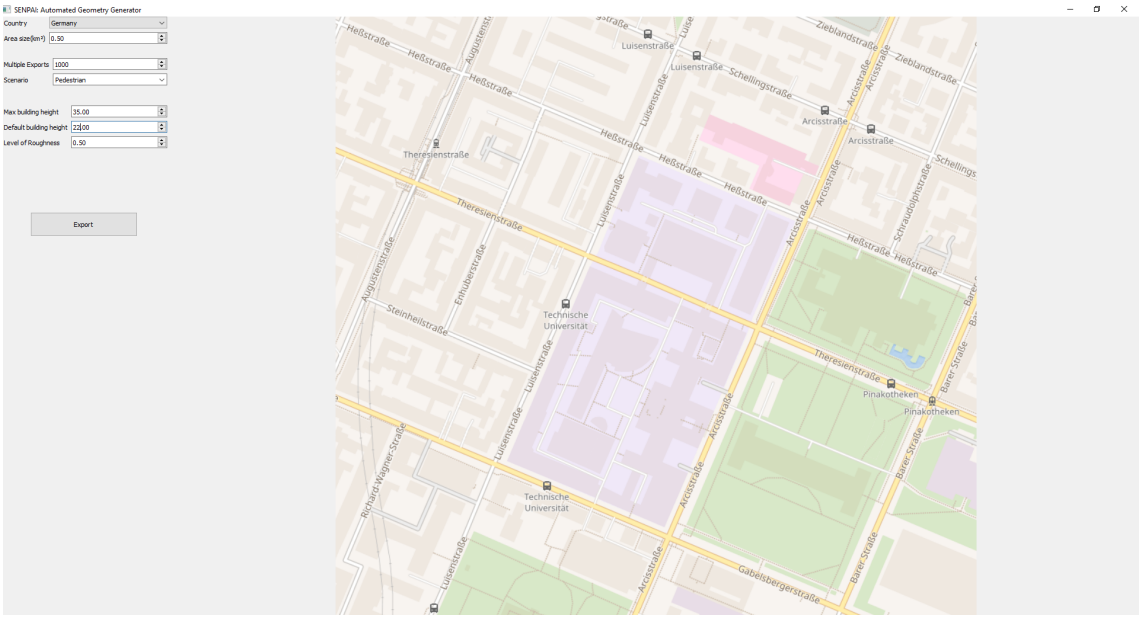


Figure 6.2 Domain Generator GUI Overview

Required Parameters

The required parameters, are the options provided to the user through the GUI. They can be roughly divided into two major categories - geometry and boundary conditions. The geometry parameters focus on the scale of the domain, such as building height, geographical region, and size of the observed domain. These parameters control the basic aspects of the Geometry and allow for a geometry that fits into the consistent static boundaries of the domain.

Due to the open-source nature of the data sources and the potential for vague or incomplete data to be present, a small set of default values for parameters, crucial to the simulation, are provided to the user. For the prototypical implementation of the solution, this is only the *default building height* (in meters). A further parameter, independent of the simulation scenario, is the *maximum building height* (in meters). This is a required parameter, since guidelines such as Franke et al. (2007) recommend that the simulation domain is expanded by an additional factor, based on the tallest building in the domain. A further scenario independent variable required for the user is the *area* of the simulation domain (in square kilometers). It can be either provided through the GUI option or through interactions with the map overlay. For the current

prototypical implementation, the aspect ratio of the area is locked as a 1:1 proportion. Due to the potentially large difference in building geometries, an option limiting the search window, based on *geographical location* is introduced.

Additional required parameters are related to simulation specific information. Such parameters involve the scenario itself, e.g. pollutant dispersal, heat island, pedestrian wind comfort, level of roughness, auxiliary geometry, e.g. green spaces, and small structures.

The definition of *scenario* is presented through a select menu. These options are based on current established validation scenarios in urban planning. In addition to the *geographical location* a *level of roughness* is presented to the user with a predefined value, for each available location. This value represents the extent to which smaller details in facades and ground terrain are represented. Higher numbers represent more prominent elements, which have a more notable effect on the flow, and lower numbers representing smaller features. It is an abstract value, presented to the user as a percentage value. Finally, a list of auxiliary geometries are given as an option to introduce additional complexity and detail to the simulation. Small structures and roof shapes introduce additional geometrical complexity to the domain, while green spaces require additional handling in the **Scenario Generator**.

Inferred Parameters

The inferred parameters are underlying parameters, that directly influence how the **Scenario Generator** creates and executes the simulations. These parameters build upon the required parameters directly, or have clear goals to optimize, and thus the user does not need to directly modify them. Additional decision based parameters are also included into this section.

Parameters that build upon the required parameters are options that are abstracted and embedded into the options file directly. Such parameter consists of the type of turbulence solver required, the domain boundaries, and the area of the simulation, that will be extracted for the training of the ANN. The *turbulence solvers*, as shown in Chapters 2 and 4 have various degrees of advantages in different simulation *scenarios*. Additionally, the *extracted area* depends on the simulation scenario, e.g. a pedestrian wind comfort analysis will create a horizontal slice in the lower levels of the built environment, while pollutant dispersal will measure the wind speed on a higher level. The *domain boundaries* are built upon the *maximum building height*. Following the guidelines of Franke et al. (2007); Tominaga et al. (2008) the simulation domain needs to be expanded by a scaled factor, based on the maximum height of a building.

Two crucial factors, that require optimization are the *total cell count* and the *vertical cell count*. Although the **Scenario Generator** performs additional optimizations to these values, an underlying value is necessary. As was shown in Section 4.1 the increase of both these values provides a steady increase in the precision of the simulation. The application first maximizes the *total cell count* based on the hardware capabilities of the device. Then based on this value the *vertical cell count* is optimized. Since $cells_{total} = vertical_{cells} * x_{cells} * z_{cells}$, where x_{cells} and z_{cells} represent the cell count in the two horizontal directions, an increase in the *vertical cell count* leads to a decrease in cell density in the other directions. For the prototypical implementation this value is selected as $vertical_{cells} = max\{16, 0.000016 * cells_{total}\}$. In this way an implicit priority is placed on the horizontal density, while retaining a detailed enough vertical axis.

Finally, due to the limitations as to how complex a query for a given area can be made, some additional filtering must be made during the search for appropriate areas to extract. Two further decision based parameters are introduced - *blockage ratio* and *full area*. The blockage ratio, as defined in Chapter 2, is defined to be at most 3%. The *blockage ratio* is dependent on the direction of the flow with relation to the geometry in the domain. Due to the fact that the **Scenario Generator** utilizes several directions for the wind, for each area, the *blockage ratio* is not directly computable. The prototypical application thus expands the *domain boundaries* by an additional factor to ensure that this value remains sufficiently low. Although the areas, presented to the user, fulfill the *area* requirement, it is possible, that several geometrical objects are not completely within the *area*. A tolerance value is introduced, that allows for such area to still be exported. The geometries that have parts outside the *area* are measured and if they are above the defined threshold, the area is discarded.

6.3.2 Scenario Generator

The **Scenario Generator** has as a main goal to process the information that the **Domain Generator** provides, create the appropriate set up for the simulation, execute it, and process the results. The simulation tool utilized in the prototype is OpenFOAM (OpenFOAM Foundation, 2022), due to its extensive knowledge base, support for a wide variety of solvers and easy exchangeability of the various components. More details about this prototype can be seen in Appendix A.3. The **Scenario Generator** processes the output of the **Domain Generator**, creating the necessary OpenFOAM data structures. Once all the relevant simulation structures are created, the simulations are executed. *Scenario* specific outputs are then generated. The core focus

areas of the **Scenario Generator** can be divided into simulation specific tasks and auxiliary tasks.

To introduce reliable and realistic simulations, the application utilizes an external solution to provide several location accurate wind direction and speeds for each area. In order for wind directions to be represented correctly, an approach similar to those used in wind tunnel simulations is utilized. Rather than defining separate parts of the boundary domain for each wind direction simulation, the geometrical model itself is rotated. This allows for only one side of the domain to be consistently defined as the inflow, the opposite side as the outflow. This leads to the requirement, that for all wind directions a separate grid needs to be generated.

Simulation Tasks

The simulation tasks focus on the generation of the simulation specific files and structures associated with the OpenFOAM environment. Some of these tasks are importing the already defined simulation parameters from the options file generated through the **Domain Generator**. Such a parameter is the *turbulence solver*, which is directly dependent on the *scenario* parameter. The Navier-Stoke Equations are solved through the SIMPLE algorithm in all *scenario*s. The *boundary domain* is reconstructed through several **Domain Generator** parameters - the *full area* and the *domain boundaries*. The combination of these parameters allows for the definition of the full domain. Finally, the *level of roughness* option is imported. This value is directly integrated into the wall functions for the geometry and the ground terrain.

Auxiliary Tasks

The auxiliary task focus on three main objectives - the set-up of the output, the definition of the variations of wind directions for each simulation, and the optimization of the simulation for the hardware environment. The *output sampling* is defined based on the provided *scenario* and the computed *boundary domain* from the simulation tasks. The OpenFOAM solution provides various types of geometry to perform uniform sampling - points, lines, planes, volumes. Based on the defined *scenario* a specific geometry is selected. This geometry is fitted then to the whole *boundary domain*, e.g. a plane through the hole simulation domain at 10 meter elevation above the ground for pedestrian wind comfort analysis. The *output sampling* of OpenFOAM requires further processing to be correctly transformed into the respective flow fields required by the ANN. An external API is utilized in combination with the *geographical location* to extract accurate wind patterns for the given geometry. Furthermore, the

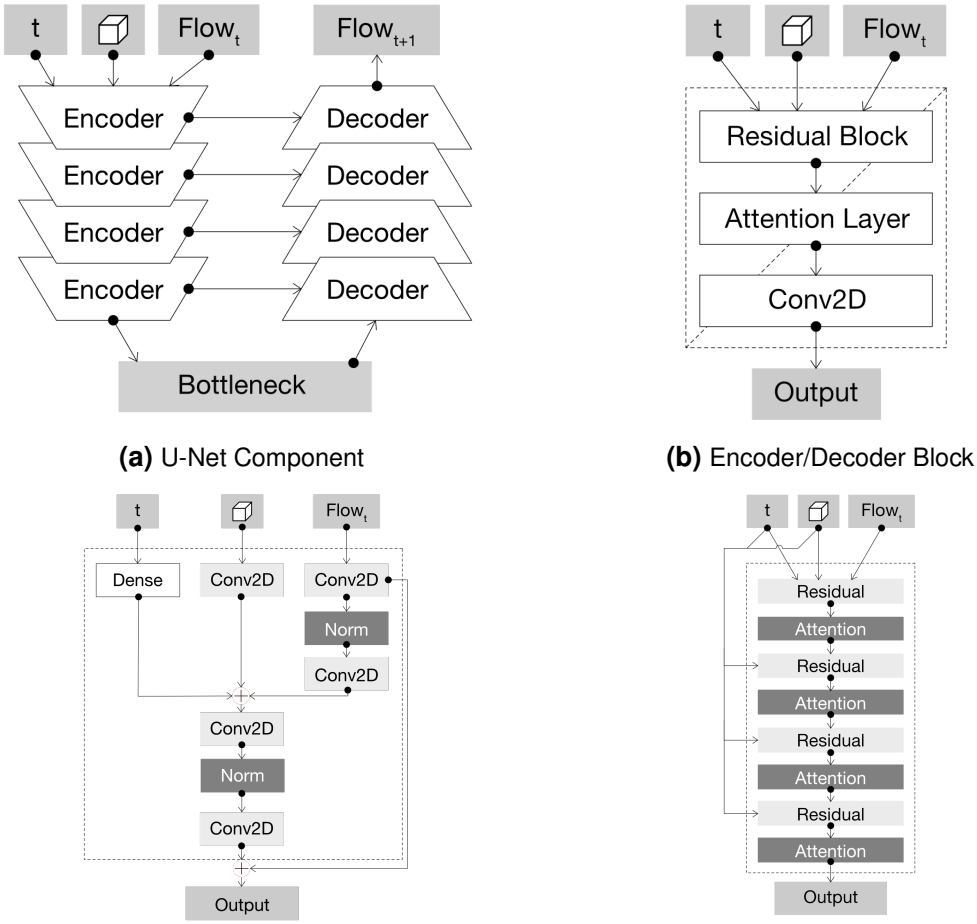
cell density is computed based on the *total cell count* and the *vertical cell count*. The application of a further meshing algorithm, through OpenFOAM, is utilized as well, to optimize the cell distribution within the simulation domain, i.e. higher density around geometries and lower density in free flow areas.

6.4 DiffusioFlow

Due to the generally young field of DL there is no definitive model, solution or approach that can perform the goals outlined in this dissertation. For this purpose a focus on a novel model and the underlying hyperparameter optimization of it is necessary (Yeh et al., 2021; Brodzicki et al., 2021). A further important component, that greatly influences the results of the network is how they learn from the training data. As shown with PINNs, error functions that are tailored to the data more closely, provide higher accuracy. The goal of the ANN, **DiffusioFlow**, is to accurately predict the next time step of a flow simulation. The full architecture of the model can be seen in Figure 6.3.

6.4.1 Data

A crucial factor in the construction of **DiffusioFlow** is the definition of the most relevant parameters required in the prediction of a fluid flow. Based on the state-of-the-art approaches, presented in Chapter 4, several core values are defined - velocity, pressure, geometry, and time (Figure 6.4). The expected input data of **DiffusioFlow** consists of 3 main components - the timestep, the geometry height map and the flow relevant fields for the given timestep. The timestep is represented as an integer value. The maximum value depends on the longest simulation utilized in training. This input is transformed using an embedding to match the resolution of the height map and the flow relevant fields. The height map for the geometry and the flow relevant fields are both represented through separate 2D Cartesian grids. For simulations, that have reached equilibrium before the defined maximal timestep, the final state of the simulation is extended for the missing time steps. This way **DiffusioFlow** has the capability of learning the concept of equilibrium states as well, while also providing the structured data approach required by the ANN. The height map, similar to the timestep, goes through a separate embedding. This embedded version is utilized in every component of the ANN. The flow relevant fields are represented through the velocity, where dependent on the simulation scenario, involves all 3 velocity components or just a combination of them. Included in the flow relevant values is the pressure value



(a) U-Net Component

(b) Encoder/Decoder Block

(c) Residual Block

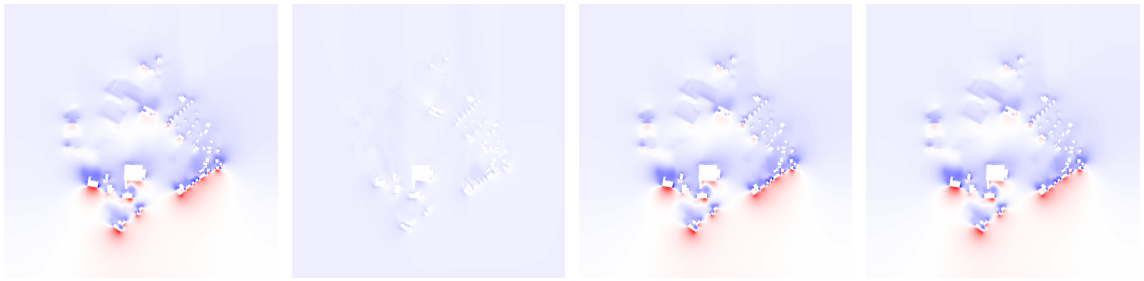
(d) Bottleneck

Figure 6.3 DiffusioFlow Architecture

as well. The flow fields and height map are also normalized to a value range between $[-1, 1]$. The output of the DL model is the predicted velocity fields, same as the input fields, but for one timestep further into the simulation.

6.4.2 Architecture

The backbone of **DiffusioFlow** is a U-Net Architecture. This form of ANN has consistently showcased to provide the most reliable results when compared to other approaches. The network consists of several Encoder/Decoder blocks that have a similar structure. As defined in Section 3.4.4 the amount of Encoders is equal to the Decoder count. Each Encoder/Decoder block (Figure 6.3b) is constructed in the same fashion. The block receives the embedded timestep and height map, and the flow fields, which have been processed through the previous encoder, decoder or



(a) Velocity in X **(b)** Velocity in Y **(c)** Velocity in Z **(d)** Pressure

(e) Height Map

Figure 6.4 Visual representation of the training data. Time input not included.

bottleneck component. The three inputs are passed through a Residual Block (Figure 6.3c)(He et al., 2016). The Residual Blocks, consist of four separate components:

- *Time*: A MLP layer that is utilized to learn from the embedded times
- *Geometry*: A Convolutional layer, with kernel size of 3, that processes the embedded height map
- *Flow Embedding*: A Convolutional layer, with kernel size of 1, that processes the flow fields and serves as an embedding for the *Flow*
- *Flow*: A set of Convolutional layers, with kernel size of 3, each followed by a Group Normalization Layers (Wu and He, 2018) is applied on top of the *Flow Embedding*

Finally, a concatenation between the final output of the *Flow* component with the *Flow Embedding* is performed. This introduces a skip connection, similar to the original Residual Block introduced in He et al. (2016).

The Residual Block is followed with an optional Attention Layer. This Attention Layer is only introduced in the Encoder/Decoder blocks, where the resolution of the

original data is low enough. This allows for better global coherence. Finally, a simple Convolutional Layer with kernel size of 3 is used to process the information. For the Encoder Blocks this layer is utilizing a stride of 2, thus learning how to efficiently down sample the image as well. For the Decoder Blocks the final Convolutional Layer is preceded with an Upsampling layer that increases the resolution of the latent space. As with traditional Encoder Blocks the depth of the feature space is increased, while the original resolution of the data are decreased. The Decoder performs the inverse task - reduces the latent space depth but increases the resolution of the data.

The amount of Encoder/Decoder pairs, referred to as the depth of the network, is dependent on the size of the input data and the scenario. The size of the flow field and height map define a minimum amount of such pairs. This aims to provide the network with a deep, low-resolution set of latent space data, while also insuring that the aforementioned Attention Layers are also utilized. Further Encoder/Decoder blocks is then dependent on the scenario that is being simulated.

At the lowest level of the ANN a Bottleneck component (Figure 6.3d) is introduced. This block consists of several Residual Blocks, followed by Attention Layers. This allows for further efficient processing of the data at a lower resolution, with higher dimensionality of the latent space.

After the final Decoder block a Group Normalization Layer is applied to its output. A final Convolutional Layer, with a kernel size of 3, processes the normalized result and outputs a 2D dataset with the same number of channels as the input flow fields. A $tanh$ activation function is utilized to map the output of this Convolutional Layer to the value range of $[-1, 1]$.

6.4.3 Learning

There are two main components that highlight the novel approach in predicting CWE simulations with the use of ANN - the sampling process utilized for the diffusion aspect of the network and the error function used to train the model.

The sampling process in traditional Diffusion ANN is represented through a form of noise sampling that can be utilized iteratively. This allows for the network to iteratively learn how to remove small amounts of noise from the original input. In **DiffusioFlow** this process is greatly altered. The training data is grouped per simulation as a singular instance of data with an inverted order, i.e. the final simulation step is the first step. This means that the representation of change between each reverse time step in a singular simulation, is the equivalent of the forward diffusion process in traditional Diffusion ANNs. This enables the network to learn the reverse diffusion process, i.e.

how exactly the flow fields change in normal forward time step. An example of the forward and reverse diffusion processes are presented in Figure 6.5.

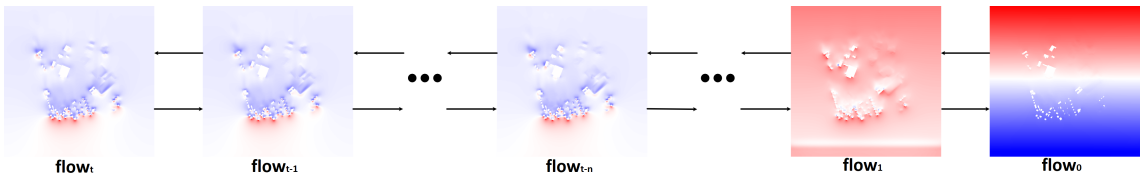


Figure 6.5 Forward and Reverse Diffusion Processes Example. The Forward process starts at the final timestep t and ends at timestep 0.

The importance of error functions cannot be sufficiently emphasized. They serve as the main driving force behind how an ANN learns, and thus it is crucial to find an appropriate function to fit the problem. As was shown in Section 5.2 there is no clear-cut choice of a loss function. The error functions can be further separated into two specific groups - the informed and uninformed error functions. Informed Error functions are structured around specific properties of the data itself, while uninformed error functions apply established approaches in evaluating differences between generic data. It has been shown through the state-of-the-art review (Chapter 4) and the in depth analysis of DL solutions (Section 5.2) that informed error functions ANN models achieve higher precision. Informed error functions are traditionally based on either existing uninformed error functions, such as the modified L_1 Norm in Ribeiro et al. (2020); Mokhtar et al. (2021) or modified version of the L_2 Norm in Xie et al. (2018). Some more complex approaches such as with PINNs, utilize the derivatives and other properties of the results of a network to compute several modified uniformed error functions, such as residual loss and boundary loss, and use their combined loss in the training of the Network (Laubscher and Rousseau, 2021; Laubscher, 2021). For the prototypical implementation the loss function for the AE in Hasegawa et al. (2020), which is a combination of the MSE and the gradient difference loss (Ehlert et al., 2019) is utilized. It is given as:

$$\mathbb{L} = \frac{1}{N_x}\frac{1}{N_y}\frac{1}{N_\phi}\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\sum_{k=1}^{N_\phi}(|(g(i,j,k) - g(i-1,j,k)) - (p(i,j,k) - p(i-1,j,k))| \\ +|(g(i,j,k) - g(i,j-1,k)) - (p(i,j,k) - p(i,j-1,k))|)$$

where $q()$ is the value for a given parameter k of the flow field for position i, j from the ground truth, and $p()$ is the value output from the ANN.

6.4.4 Hyperparameters

In addition to the definition of the Architecture and Error Functions of **DiffusioFlow** additional general hyperparameter optimization is also required. Due to the refined nature of available technical solutions for ANN several parameters are listed:

- **Epochs** - Epochs define how many times the ANN is trained on the whole data set, larger values provide better training results, but often lead to overfitting. This causes the Network to be capable of reproducing the training data perfectly, but be unsuited for unobserved values.
- **Depth** - The depth of the **DiffusioFlow** represents the amount of Encoder / Decoder blocks the U-Net has. Higher amount can obtain more abstract low resolution latent space, but is associated with longer training times.
- **Width** - Similar to depth, the width controls the complexity of each Encoder / Decoder block. This includes the amount of Residual Blocks and the utilization of Attention as well.

7 Validation

Through the presented concept in Chapter 6, a prototypical implementation of the core components has been completed - the **Domain Generator**, the **Scenario Generator**, and **DiffusioFlow**. The prototypes are tested in the order presented in Figure 6.1. A singular scenario, the pedestrian wind comfort, is used for the analysis and validation of each prototypical component. Each prototype is evaluated and analyzed based on the following separate criteria:

- The **Domain Generator** is evaluated on the success rate of finding appropriate geometry through the various data sources utilized. The performance of the export process is evaluated and the dependencies for the performance of the export are analyzed.

- The **Scenario Generator** is evaluated based on how well the abstraction of complex parameters has been achieved. The generalization of such parameters can lead to disruptions and unsuccessful execution of the CFD simulation. The success rate of the **Scenario Generator** is measured based the execution of each simulation, and the cause of the disruption. To successfully compare the ANN with established approaches the performance of the **Scenario Generator** is analyzed and underlying correlations between the geometry and simulation parameters is performed.

- The ANN **DiffusioFlow** aims to improve on the existing approaches based on the reliability, precision and responsiveness criteria, outline in Chapter 1. The prototypical implementation is evaluated based on the achieved speed-up in comparison to established method, utilized in the **Scenario Generator** as well as the accuracy and precision of these results.

This chapter defines the outlines for the validation and analysis process in detail. Furthermore, a detail analysis and validation of all 3 prototypical components is performed and presented.

7.1 Set-up

The creation of the evaluation process requires a consistent testing environment and clear validation and analysis targets. A focus testing on all three core applications, the **Domain Generator**, the **Scenario Generator**, and **DiffusioFlow** is done on the same machine, to insure consistent and comparable results. The applications were tested on an Ubuntu 22.04.2 LTS System, with an Intel Xeon E5-2687W CPU, 32 GB of Random Access Memory (RAM), and an NVIDIA GeForce GTX 780 GPU. For each evaluation, a separate scenario is constructed, tailored to the concrete case.

7.2 Domain Generator Evaluation

Due to the prototypical implementation of the solution, only a single scenario is made available for the validation - pedestrian wind comfort analysis. The validation and analysis of this component is focused on the following criterion: the time required to export a predefined amount of areas, based on a fix set of criteria. This criterion takes into consideration how long each singular area requires for the processing and exporting of the area. Furthermore, it evaluates how many areas are discarded during the process, due to the *full area* implicit parameter defined in Section 6.3.1. For the purpose of this evaluation, the values for the **Domain Generator** explicit parameter are defined in Table 7.1.

Parameter Name	Value
Default Building Height	$22m$
Maximum Building Height	$35m$
Area	$0.5km^2$
Geographical Location	Germany
Scenario	Pedestrian Wind Comfort
Export Target	50
Data Source	OpenStreetMap

Table 7.1 Domain Generator Explicit Parameters

Figure 7.1 outlines the total amount of areas that were evaluated for the export process. Due to the implicit parameters, such as *blockage ratio* and *full area*, that filter areas based on the geometry, there are unsuited areas. As can be seen from Figure 7.1, the additional expansion of the *domain boundaries* to accommodate for a lower *blockage ratio* and *full area* has lead to a high success rate in the exporting process.

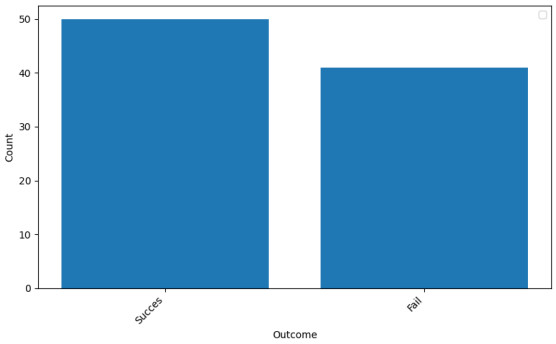
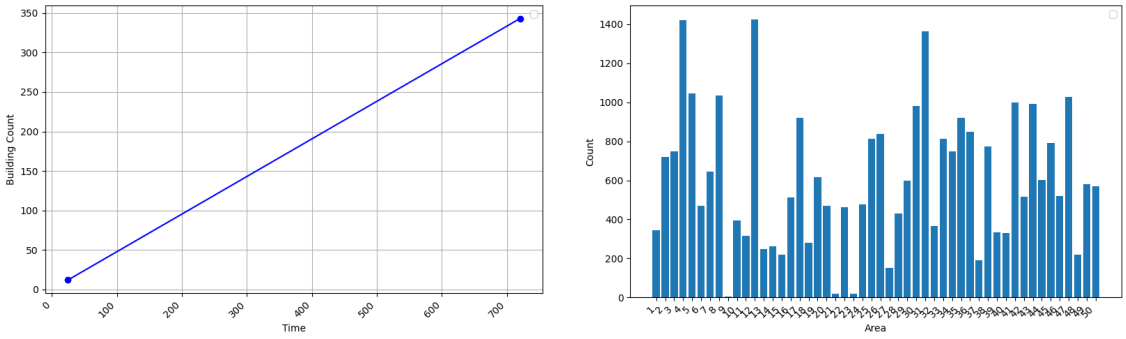


Figure 7.1 Domain Generation Success Rate

Figure 7.2a evaluates the export time of each successful area in relation the amount of separate geometries represented in the *data source*. Due to the open-source nature of the *data source*, the representation of various urban elements, such as buildings and green spaces, is inconsistent. This leads to the wide variation in export times - larger amount of smaller entities requires additional processing time, while areas with large singular objects are processed quicker. The varied amount of separate entities per successful are export can be viewed in Figure 7.2b.



(a) Export Time per Area

(b) Separate Building Entities per Area

Figure 7.2 Exported Area Overview

7.3 Scenario Generator Evaluation

The **Scenario Generator** builds upon the results of the **Domain Generator**. Due to the time-consuming process of performing CFD simulations, a small subset of the areas, extracted with the **Domain Generator** are used in the evaluation - 14 areas with 4 separate wind directions, for a total of 56 simulations. This validation is focused on two major components - the required time per simulation, to be fully executed, and the successful automation and abstraction of the CFD simulation parameters.

The time consists of two core components - the creation of an efficient grid for the simulation and the execution of the simulation itself. The success of the automation and abstraction is measured through the amount of simulation, that were successfully executed. Due to the singular available *scenario* in the **Domain Generator** and the underlying hardware solution, an overview of the concrete choice for the parameters outlined in Section 6.3.2 are presented in Table 7.2. Additionally, each simulation was executed with a parallelization using a multi-CPU approach, greatly increasing the computational speed. Each simulation is executed until it reaches an equilibrium state (Section 2.3.1) or reaches a defined amount of time steps.

Parameter Name	Value
Turbulence Solver	RANS $k - \epsilon$
Boundary Domain	$1.35km^2$
Output Sampling	Horizontal Domain Plane at $10m$
Cell Density	1.5 Million Cells
Maximum Time Steps	2000
Exported Time Steps	200

Table 7.2 Domain Generator Explicit Parameters

The main evaluation criteria of the **Scenario Generator** is the success with which the complex CFD modelling task has been abstracted. Figure 7.3 showcases the average success rate of such simulation. The analysis distinguishes between the reason for an unsuccessful simulation - simulation issue, that arose during the simulation, or a meshing issue, that arose during the creation of the grid for the simulation. As can be seen from Figure 7.3 only a third of simulations were executed successfully. The main source of these unsuccessful simulation were caused by issues, that arose during the execution of the simulation itself. Through an analysis of the auxiliary information from such simulations, an underlying cause can be established. The core reason is numerical instability with the utilization of the pressure solver. The exact issue to correct such issue requires further analysis, but two major sources of instability are:

1. the parallelization of the simulation - communication between the various processes introduces additional numerical impression
2. poor refinement of the grid - if grid cells have poor alignment or complex shapes are not handled by the meshing process correctly, numerical impressions can occur.

The process for refinement of the grid did not have sufficient hardware resources to be executed which resulted in the 15 meshing errors. This led to an abrupt termination

of the process, resulting in a domain that has no geometry. This leads to a simulation of an empty domain, making the results unsuited for the ANN.

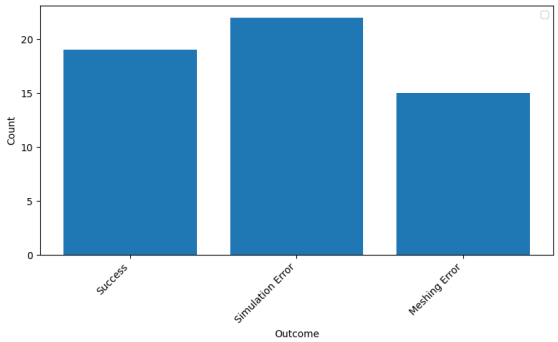


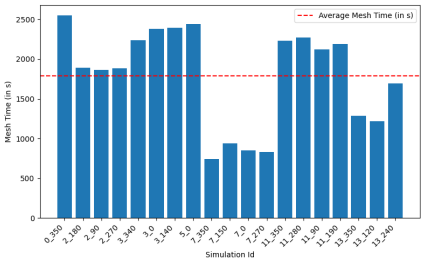
Figure 7.3 Simulation Success Rate

The evaluation of the required time per simulation is divided into 2 major components - the creation of the grid and the simulation itself. An overview of each separate area and their successful meshing and simulation components are presented in Figures 7.4a and 7.5a. As can be seen from the overview, the direction of the wind has a noticeable effect on the runtime. This is due to the way different wind directions are implemented, as outlined in Section 6.3.2. The grid generation consists of two parts, first a base Cartesian grid, aligned to the domain directions, is created. Based on this grid, a secondary algorithm refines each cell, introducing various levels of refinement around obstacles. It is thus clear that, if the repositioning of the geometry positions them to be at an acute angle against the domain directions, a more complex refinement is necessary.

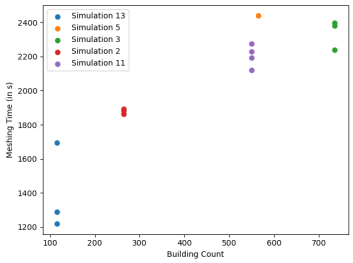
The total meshing and simulation times, shown in Figure 7.4a and Figure 7.5a, highlight a further underlying dependency for the runtime of the CFD simulation - the complexity of the geometry in the domain. Figures 7.4b and 7.5b highlight that a more crucial factor, that influences the runtime of both meshing and simulation time, is the amount of separate elements present in the domain. As can be seen from these results, there is a stronger correlation between the meshing and simulation process and the complexity of the environment.

7.4 DiffusioFlow Evaluation

The evaluation of the ANN **DiffusioFlow** is done utilizing the data provided through the successful simulations of the **Scenario Generator**. Due to the vast amount of parameters available for the optimization of the precision of the ANN, a predefined combination is selected, based on the literature review of the state-of-the-art solutions

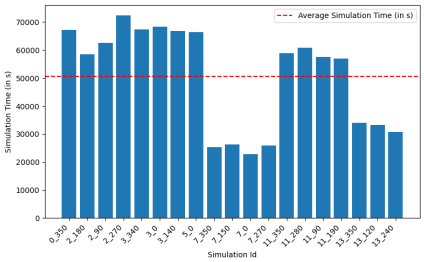


(a) Meshing Time for all Areas and Directions

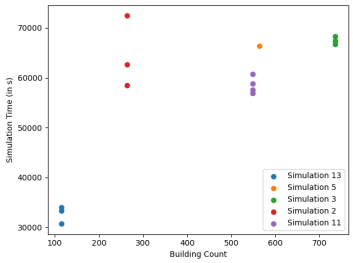


(b) Meshing Time and Geometry Count

Figure 7.4 Meshing Time Overview



(a) Simulation Time for all Areas and Directions



(b) Simulation Time and Geometry Count

Figure 7.5 Simulation Time Overview

from Chapter 4. An overview of the specific parameters are available in Table 7.3. As outlined in Chapter 6 the error function is defined as a combination of the gradient difference loss and the MSE. To evaluate the predictions of the ANN, the MSE metric is utilized.

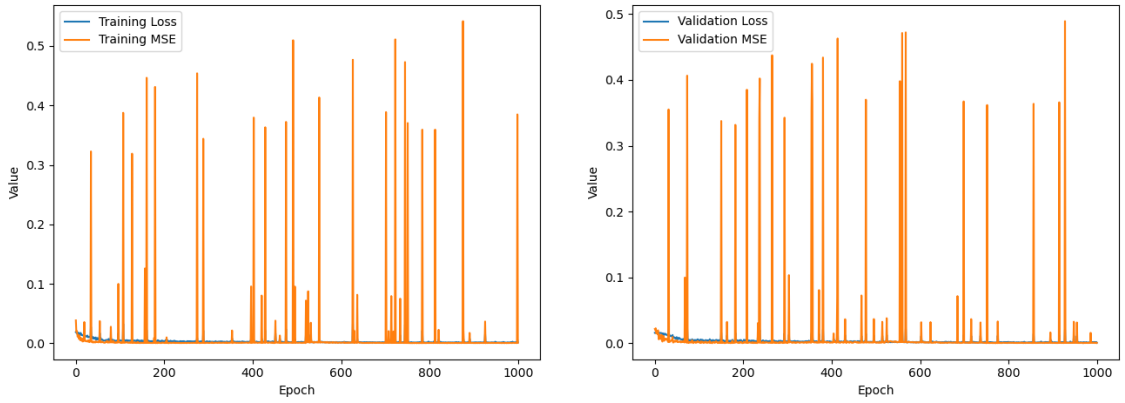
Parameter Name	Value
Flow Field Values	U_x, U_y, U_z, p
Flow Field and Height Map Resolution	512 by 512
U-Net Depth	4
Features per level	$[32, 64, 128, 256]$
Residual Blocks per level	2
Attention Layer per level	$[0, 0, 0, 1]$
Batch Size	2
Epochs	1000

Table 7.3 DiffusioFlow Parameters

The implemented prototype of the **DiffusioFlow** is evaluated based on three core criteria - the overall success of the training process on the data, the responsiveness, and the reliability and precision, with regard to the simulation results obtained from **Scenario Generator**. Due to the low amount of training data available, an additional

augmentation to the data is introduced, during the Diffusion Sampling Process. This allows for the increase of data by several factors, allowing for a more robust training.

Figure 7.6 highlights, that the chosen approach and provided data, allow for the ANN to reach a high level of accuracy with regard to the training and validation data. The observed peaks for the MSE metric point to the issue, that for given time steps the ANN did not achieve sufficient accuracy. The results of the validation can be further improved, with the introduction of a larger dataset and a higher amount of training Epochs. To accurately measure if the trained version of **DiffusioFlow** can improve upon the responsiveness of CFD simulations, while also maintaining a sufficiently high precision and reliability a comparison between the two approaches is performed.



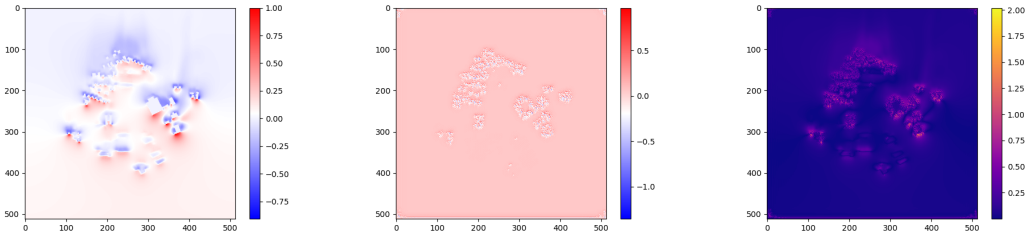
(a) Training progress

(b) Validation progress

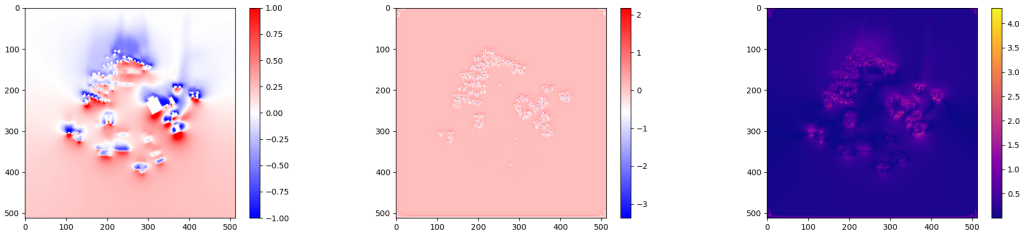
Figure 7.6 Training and Validation of **DiffusioFlow**

Figure 7.8 presents a comparison between the performance of the OpenFOAM solution, utilized in the **Scenario Generator**, and the predicted equivalent results of **DiffusioFlow**. For the OpenFOAM solution the total computational time, including meshing and simulation time, are considered. For the ANN the performance is measured based on the time it requires to predict the full set of time steps for each simulation. The generation of the first time step based on the geometry is taken into this value. Figure 7.8a shows that the increase in responsiveness, through the achieved speed-up, is sufficiently high - a factor of up to 70. A key insight obtained from this result is that although the OpenFOAM simulations have vastly differing times, dependent on the geometry in the environment, the results of **DiffusioFlow** are consistent. This allows for an exponential speed-up, dependent on the complexity of the domain.

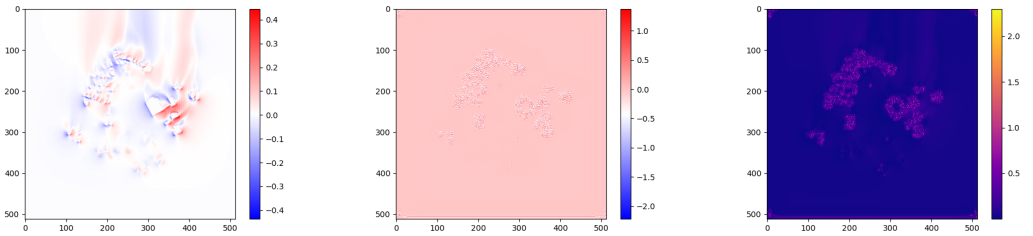
To measure the precision and reliability of the predicted results the MSE Metric is used to compare them with the results obtained through OpenFOAM. Each flow field component is compared - the velocity in all 3 directions and the pressure. As can be seen from Figure 7.8b the difference between both approaches is within a sufficient



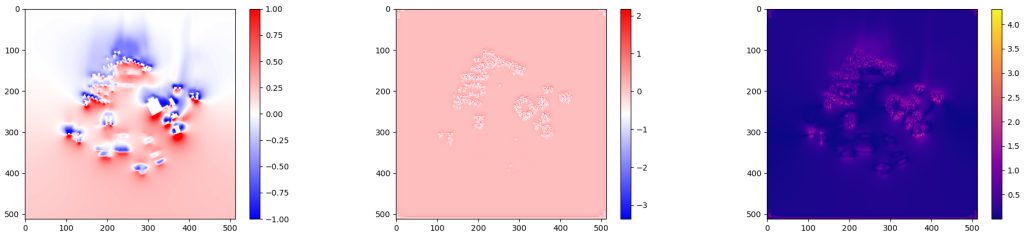
(a) Ground Truth for p **(b)** Prediction for p **(c)** Difference for p



(d) Ground Truth for U_x **(e)** Prediction for U_x **(f)** Difference for U_x



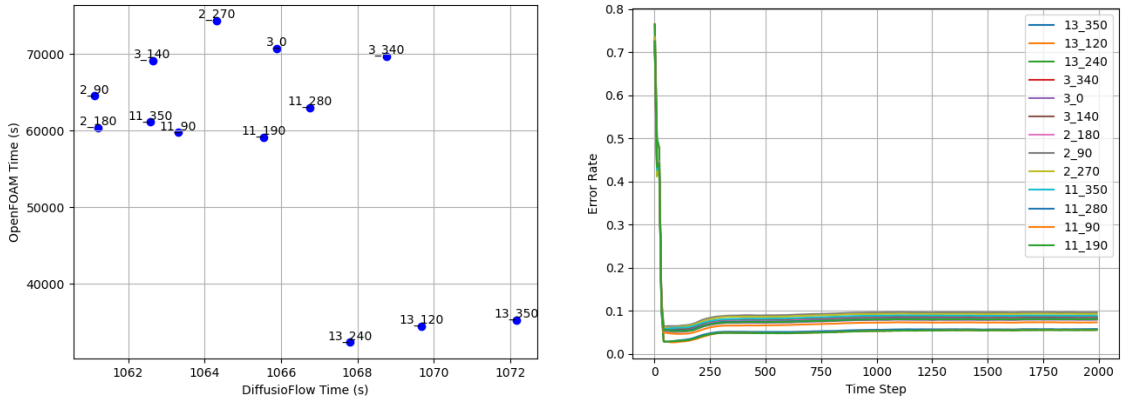
(g) Ground Truth for U_y **(h)** Prediction for U_y **(i)** Difference for U_y



(j) Ground Truth for U_z **(k)** Prediction for U_z **(l)** Difference for U_z

Figure 7.7 Visual Comparison between **DiffusioFlow** and OpenFOAM for Case 13 and Wind Direction 120

range. Additionally, Figure 7.7 highlights the areas in which **DiffusioFlow** achieved the lowest accuracy in comparison to OpenFOAM. As can be seen from this, although the total difference is around 10% per simulation step (see Figure 7.8b), the approach did not capture the motion of fluid accurately. This issues can be resolved with a higher amount of training data or a smaller fluid time step.



(a) Responsiveness comparison

(b) Reliability and Precision comparison

Figure 7.8 Performance Comparison between **DiffusioFlow** and OpenFOAM. Labels represent Simulation Area and Wind Direction

7.5 Conclusion

This chapter provides an overview of how well each of the prototypical implementations of the core concepts proposed in Chapter 6 performed. Due to the different goals that each prototype attempts to fulfill a different set of evaluation criteria for each was developed.

The **Domain Generator** has achieved a satisfactory performance, capable of extracting simulation areas with consistent performance that is dependent only on the complexity of the area. Nevertheless, the prototype requires further improvement - a more expanded set of different CWE scenario need to be integrated, and better handling of the geometry in areas that exceed the maximum area limit is required.

The success of the **Scenario Generator** needs to be further expanded upon. Nevertheless, key insights into the issues were obtained, highlighting a concrete area of improvement - the creation of a robust and precise simulation grid. On one hand, an improved meshing approach would reduce numerical instabilities in regions of domains that have complex geometries. On the other hand, a more memory-efficient grid creation will allow for its successful execution on a wider range of hardware setups.

Finally, **DiffusioFlow** showcased that the proposed hypothesis in Chapter 1 is achievable. The substantial increase in responsiveness, in comparison to the OpenFOAM solution, provides the possibility for better integration of CFD simulations into the early stages of design. Nevertheless, the precision and reliability of the proposed approach required further improvement. Due to the small data size, the ANN was not successful in capturing the correct fluid motion. The Diffusion DL methods are developed to iteratively modify the provided data by a small step. Based on this, a longer

training phase is required, and a potentially smaller time step of the fluid needs to be utilized in order to achieve more reliable and precise predictions.

8 Summary and Outlook

The integration of CFD / CWE simulation tools in the early stages of design is crucial for the planning process. Such solutions provide the user with objective feedback on a wide range of potential issues - from pedestrian wind comfort and pollutant dispersal, to heat island analysis and structural integrity of buildings. Utilizing such tools out of the box cause a negative disruption to the established design processes, characterize by its creative and iterative nature. This is caused by the simulation tools need for more detailed information about the design, often not available in the early stages. Furthermore, such methods often require expert knowledge to utilize correctly. The long computational times, associated with the complex and detailed simulation tools bring a further challenge for their integration into the early stages of design. Such tools have been shown to provide the planner with reliable and precise information, thus need for new approaches for their integration in the early design stages is necessary.

This dissertation presents the novel concept of utilizing state-of-the-art DL models in conjunction with reliable and precise CWE wind simulations for their integration as DDS tools in the early stages of design. Through the provided analysis clear requirements towards this hybrid approach have been defined. Furthermore, this dissertation takes into consideration the complete process of creating DL models suited for this task. Based on these observations, three core conceptual components were defined:

- **Domain Generator:** Abstracts the complex task of defining a CWE simulation for a specific task
- **Scenario Generator:** Utilizing the input from the **Domain Generator**, both the scenario relevant parameters and the boundary conditions, creates the corresponding simulation and all fine-tuned parameters
- **DiffusioFlow:** Based on the preprocessed output of the **Scenario Generator** the Diffusion ANN is trained, and the trained model is stored

The utilization of these components allows for the fulfillment of the requirements towards simulation and analysis tools for the successful integration as DDS solutions into the early stages of design. The greatly improved responsiveness and user-friendliness achieved through the use of a DL model allows for the seamless use

of CWE simulations in the early design stages. This proposed solution thus provides the following clear advantages:

- The early stages of design can be performed with better support, through the integration of a wide range of simulations as DDS tools, thus providing objective evaluations on the design without disruption.

- The flexible construction of the prototype pipeline allows for the user-friendly adaptation and creation of trained models for a wide range of various scenarios

- The utilization of state-of-the-art DL solutions enables the pipeline results, the trained **DiffusioFlow** model, to be easily and seamlessly embedded into existing design tools.

The concept and dissertation hypothesis were proven through the implementation of the crucial pipeline components. These subsystems represent prototypical solutions and thus require further refinement and analysis.

- Although it is possible to define the boundary conditions through the **Domain Generator** through user-friendly parameters, it utilized parameters for only one singular scenario. Due to the lack of CWE guidelines that focus on the optimization of such simulations for specific scenarios, further research and refinement is required.

- The **Scenario Generator** successfully creates the required simulation parameters and obtains the relevant data necessary for **DiffusioFlow**. The data utilized in the training and prediction is currently represented through planar (2D) vector fields of minimal information - velocity, pressure and geometry height. The simulation performed is always 3D, thus the third axis of the simulation is omitted from the data. The integration of this additional information would greatly improve accuracy of the simulation.

- Furthermore, the low success rate during the execution of the generated simulations from **Scenario Generator** requires further improvements. This can be achieved through the introduction of better refinement schemes for the grid of the simulations, which could improve upon the meshing and simulation errors that have occurred. A further improvement can be also achieved in the simplification of the geometry in the scene. Currently, separate buildings are represented as separate geometries. Allowing for geometries, that share components, such as walls, to be merged, thus simplifying the meshing process.

- **DiffusioFlow** successfully provides the user with reliable results. The overall precision of the ANN, represented through the differences between the ground truth and the prediction, although sufficiently low requires further improvement. The ANN did not capture accurately the fluid dynamic. The underlying reason for that requires further research. Due to the still rapidly evolving nature of the field of DL a thorough analysis and comparison against other existing solutions is required. Further, model optimization, through hyperparameter optimization, can also be beneficial for the precision of the ANN.

- Based on the current approach in modeling and developing DL models, such solutions provide only approximations to the ground truth. The integration of such solutions into design tools, needs to be performed in such a way, as to inform the user of the possibility, that the result may not be sufficiently precise.

Although this dissertation has introduced a concept, as well as a prototypical solution, further challenges, questions and potential focus areas have arisen. These topics were not directly addressed in this work. The following concepts for the improvement of this dissertation as well as possible approaches are presented:

- The urban environment has been reduced to only geometrical representations, without the consideration of semantic data, and with a low LoD. Such concepts have been traditionally omitted in urban scale wind studies, due to the costly nature of integrating more complex details into the simulation. If DL models offer a way to bridge the gap between such simulations and their seamless integration into the early stages of design, then a further focus on the importance of such parameters could bring CWE and their respective DL counterparts closer to on-site measurements.

- The embedding of a simple height map into the ANN, to represent the building geometry, provided a further improvement upon the results of the model. Further research can be conducted as to what further information can provide the network with useful insights into CWE simulations.

- Although DL models can provide with reliable and sufficiently precise result it is difficult to measure the uncertainty of a model once it is trained. If such tools are utilized in the early stages of design, where the underlying information is vague itself, an appropriate way of communicating the confidence of the results, and potential discrepancies to the ground truth, is required.

In summary, this dissertation has shown, that through the utilization of DL methods, such as the proposed novel **DiffusioFlow**, it is possible to achieve significantly more

reliable results in CWE simulations in comparison to established approaches. The approach requires further improvements to the achieved precision of the predictions, as is shown in Chapter 7. Nevertheless, such surrogate models, allow for the integration of objective analysis and simulation tools as DDS tools in the early stages of design.

A Prototypes

Within the scope of this dissertation, the concept is subdivided into the core concepts. These components have been implemented across various interdisciplinary projects, lectures and stand-alone research. A large part of the prototypes have been implemented under the Simulated Engineering in New Planning with Artificial Intelligence (SENPAI) project. It serves as the connector between the various components, providing them with specific context and tailoring them to the goals of this dissertation.

These solutions focused on the core aspects needed for bridging the gap of requirements between CWE simulations and their integration as DDS tools in the early stages of design. To facilitate such an integration with the use of DL the prototypes cover also the crucial areas with regard to the generation of reliable and well-structured data for the training of the DL models.

On the basis of these core aspects the main priority of the prototypes is the creation of a DL model, capable of predicting CWE simulations reliably. A secondary focus is placed on the creation of reliable, consistent and varied data for the DL model. Finally, an adaptable approach for tailoring the simulation parameters based on the geometry and specific simulation scenario is also prototypically implemented.

A.1 SENPAI // DiffusioFlow

Development:	Stand-alone Research
Current State:	Complete
Programming Language:	Python
Additional Software Libraries:	Tensorflow, Numpy

The main concept, **DiffusioFlow**, is implemented as a stand-alone project and utilizes all further prototypes. Implemented with TensorFlow, and utilizing NumPy for processing and loading the data, the DL model provides the user with several options for the fine-tuned generation. The full architecture of the model can be seen in Figure 6.3.

The expected input data of **DiffusioFlow** consists of 3 main components - the timestep, the height map and the flow relevant fields for the given timestep. The timestep is represented as an integer value. The maximum value depends on the longest simulation utilized in training. This input is transformed using an embedding to match the resolution of the height map and the flow relevant fields. The height map for the geometry and the flow relevant fields are both represented through separate 2D Cartesian grids. For simulations, that have reached equilibrium before the defined maximal timestep, the equilibrium state is extended for the missing time steps. This way **DiffusioFlow** has the capability of learning the concept of equilibrium states as well. The height map, similar to the timestep, goes through a separate embedding that is utilized in every component of the ANN. The flow relevant fields are represented through the velocity, where dependent on the simulation scenario, involves all velocity components in 3D or just a combination of them. Included in the flow relevant values is the pressure value as well. The output of the DL model is the predicted velocity fields with the same structure and dimensions, but for one timestep later.

A.2 SENPAI // OSM Crawler

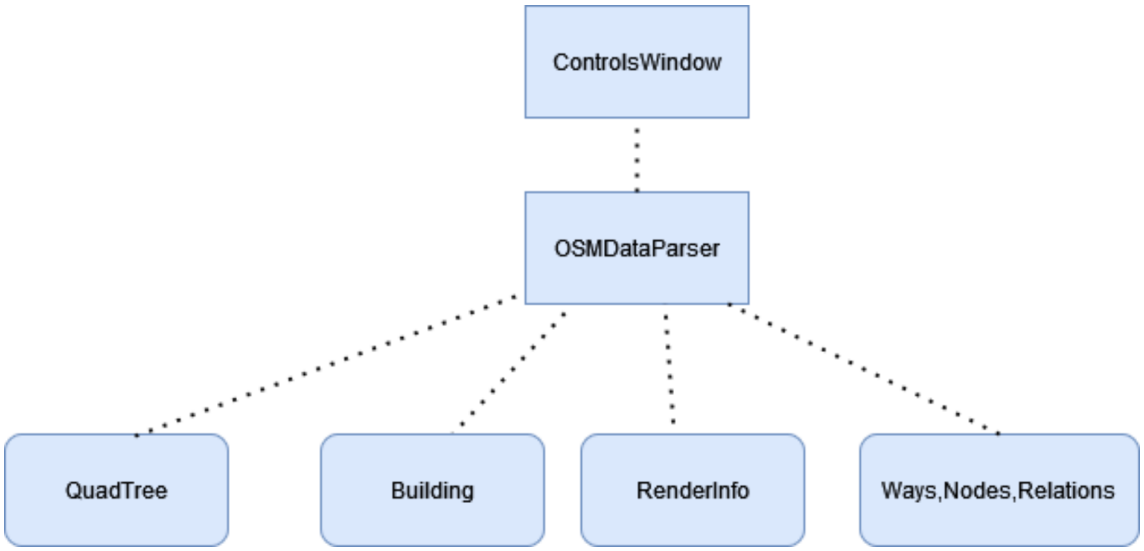


Figure A.1 SENPAI // OSM Crawler Software Architecture

Development:	Stand-alone Research & IDPs
Students:	Yunus Can Cakir (Cakir, 2022) & Jonas Herget (Herget, 2023) & Kadir Tandogan Tilki (Tilki, 2022)
Supervision:	Ivan Bratoev, M.Sc.; Dr.-Ing. Gerhard Schubert; Prof. Dr.-Ing. Frank Petzold
Current State:	Complete
Programming Language:	C++
Additional Software Libraries:	Qt

The prototype focuses on executing the requirements set out for the **Domain Generator**. It is a GUI based application that utilizes the Qt Library for visualization. This library offers native support for the integration of OpenStreetMap, allowing for easier visualization of the selected areas. The two major components of this prototype are:

- **Implicit Parameter Extraction**: based on the Explicit Parameters, controlled through the GUI, the complex implicit parameters are inferred.
- **Geometry Extraction**: Focuses on extracting the simplified geometrical representation of all areas. The filtering of areas, based on their *full area* and *blockage ratio* is performed here as well.

The extracted geometry with a combination of specific implicit and explicit parameters are stored as two separate files. The geometry of all elements in the domain

are stored as a Wavefront OBJ File, while the parameters are stored in a structured JSON File.

A.3 SENPAI // OpenFOAM Simulator

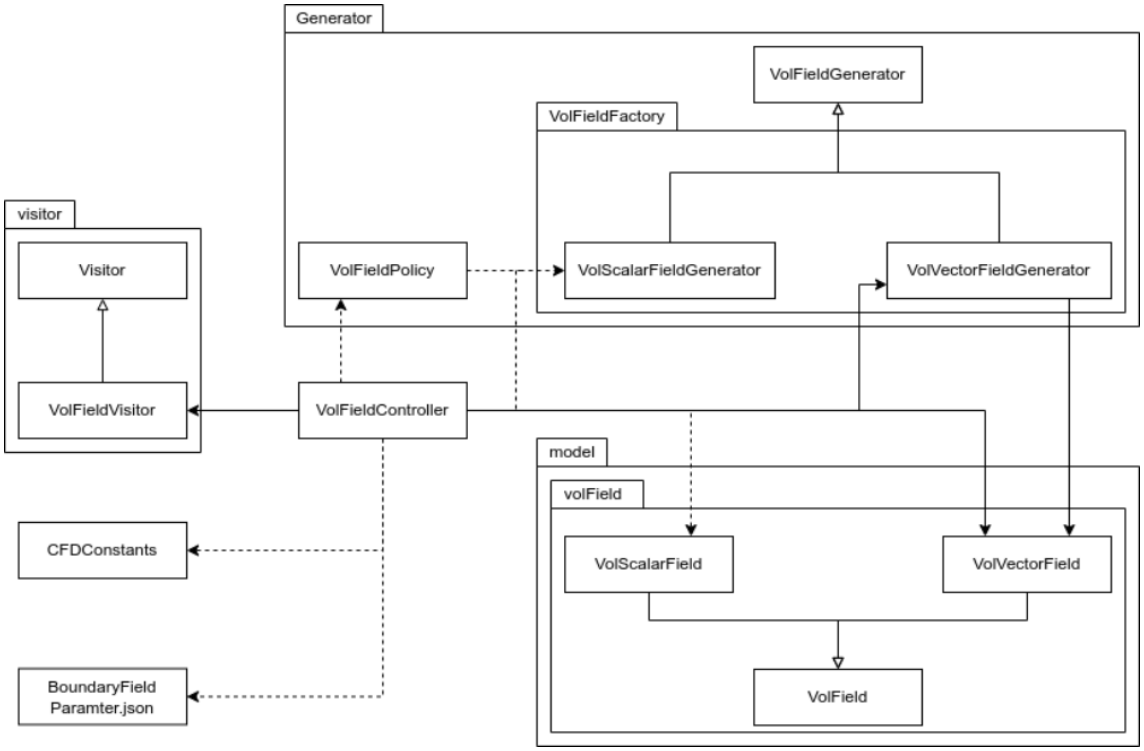


Figure A.2 SENPAI // OpenFOAM Simulator Software Architecture

Development:	Stand-alone Research & IDPs
Students:	Ivan Brkan (Brkan, 2022) & Jonas Herget (Herget, 2023)
Supervision:	Ivan Bratoev, M.Sc.; Prof. Dr.-Ing. Frank Petzold
Current State:	Complete
Programming Language:	Java

The prototype serves as the base for the implementation of the **Scenario Generator**. The prototype is implemented in Java, thus allowing for it to be executed on a wide range of hardware solutions. It is a Command-line Interface (CLI) solution. The OBJ File and JSON File of Appendix A.2 are required for the execution of the prototype. The prototype focuses on the following aspects:

- **Realistic Wind**: Based on the *geographical location* accurate weather data is extracted using the OpenWeather API. The last several days are aggregated, and the predominant directions are selected for the simulation.
- **Simulation Parameters**: Based on the stored parameters of the **Geometry Generator** the simulation specific parameters, such as *turbulence solver* are processed.

- **Auxiliary Parameters**: The definition of the simulation domain, as well as the orientation of the geometry and the type of information to be extracted is performed on both the realistic wind directions and the parameters defined in **Geometry Generator**.

The prototype can process a larger set of extracted areas and provides as output the correct layout for the OpenFOAM simulations. As discussed in Section 5.1 the OpenFOAM solution requires a predefined file and folder data structure for it to execute a simulation correctly. The prototype thus creates this structure and allows for its direct use.

A.4 SENPAI // Node Orientated Neural Network Environment

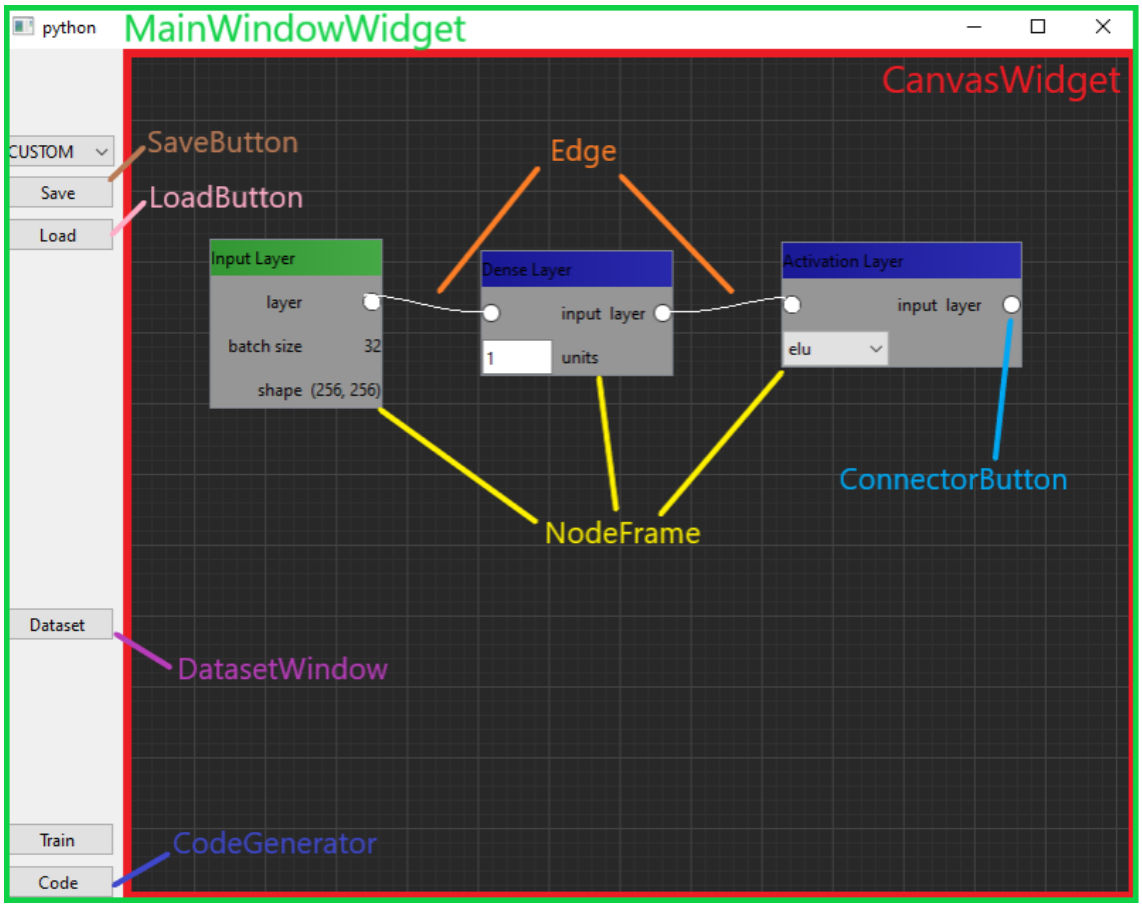


Figure A.3 SENPAI // N.O.N.N.E. GUI

Development:	Stand-alone Research & IDPs
Students:	Fabian Danisch & Eric Esch (Danisch and Esch, 2022)
Supervision:	Ivan Bratoev, M.Sc.; Alejandro Rueda, M.Sc.; Prof. Dr.-Ing. Frank Petzold
Current State:	Complete
Programming Language:	Python
Additional Libraries:	TensorFlow, Numpy, Qt

As part of the dissertation research, a focus was placed on the simplification in modelling of ANN. Such an approach would allow for a quicker iterative process while developing DL solutions for various problems. This prototype aims to introduce the concept of Visual Programming to the development of ANN. It is written in python,

to allow an easier integration with the various established libraries for DL. It has the following core components:

- **Node Representation**: Various Layers, typical for ANN are represented through simple GUI Elements such as Nodes. The left side of each Node represents the input, while the right side the output. The communication between Nodes is done through the Edges that connect the various input and outputs.
- **Dataset Loading**: A prototypical implementation for the handling of dataset is utilized to load images for various classification and segmentation tasks. The GUI offers a wide range of additional options, that facilitate data augmentation, and training and validation splits.
- **Code Generation**: Once the user has created the ANN they require the prototype allows for the export of the network as python code. This allows then for a direct execution of the concept.

An additional concept that have been considered during the implementation of the prototype include the visualization of the network in an abstract form, allowing for a more understandable overview. Furthermore, the possibility for training and utilizing ANN within the Visual Programming Environment was also considered. The prototype allows for the storing and loading of modelled ANN solutions as well.

B Glossary

B.1 Acronyms

ABL Atmospheric Boundary Layer. 32, 51

AE Autoencoder. 45–47, 49, 59–61, 70, 72, 88

AEC Architecture, Engineering and Construction. 51, 55, 57, 62

AI Artificial Intelligence. v, 1, 4–7, 33–35, 37, 44, 49

AIJ Architectural Institute of Japan. 53

ANN Artificial Neural Network. 34, 37–49, 55, 58, 62, 75–79, 81, 83–85, 87–89, 91, 95–97, 99, 101, 103, 106, 111, 112

API Application Programming Interface. 58, 68, 83

BEM Blade Element Momentum. 54, 55

BNN Bayesian Neural Network. 47–49, 56, 57

CBML Component-Based Machine Learning. 57

CDP Collaborative Design Platform. 4

CFD Computational Fluid Dynamics. 1–7, 9, 26, 29, 49, 51, 52, 54, 55, 59, 61–66, 68–70, 72–76, 78, 91, 93–95, 97, 99, 101

CLI Command-line Interface. 109

CNN Convolutional Neural Network. 44–46, 49, 59–62, 69

CPU Central Processing Units. 66, 92, 94

CWE Computational Wind Engineering. v, vii, 1–3, 5–7, 49, 51, 52, 55, 62, 63, 65, 66, 68, 77, 87, 99, 101–105

DDS Design Decision Support. 2, 4, 63, 64, 68, 75, 101, 102, 104, 105

DL Deep Learning. v, vii, 7, 33–35, 37, 38, 49, 51, 55–58, 60–63, 68, 69, 73–75, 84, 85, 88, 99, 101–103, 105, 106, 111, 112

DNS Direct Numerical Simulation. 25, 59, 61

DSGS Dynamic SGS. 25

ELBO Evidence Lower Bound. 48

GAN Generative Adversarial Network. 48, 49, 57, 61, 72

GELU Gaussian Error Linear Unit. 39

GPU Graphical Processing Units. 66, 70, 92

GUI Graphical User Interface. 3, 68, 79, 80, 107, 111, 112, 117

HVAC Heating, Ventilation and Air Conditioning. 66

LeakyReLU Leaky Rectified Linear Unit. 39

LES Large Eddy Simulation. 24, 26, 30, 31, 52–54, 59

LoD Level of Detail. 28, 57, 77, 103

LSTM Long Short-Term Memory. 61, 72, 73

MAE Mean Absolute Error. 42, 70

ML Machine Learning. 33–35, 37, 49, 55

MLP Multi-layer Perceptron. 44, 56–58, 86

MSE Mean Squared Error. 41, 70, 88, 96, 97

MSLE Mean Squared Logarithmic Error. 41

NLP Natural Language Processing. 46

PDE Partial Differential Equations. 58

PET Physiologically Equivalent Temperature. 56

PINN Physics-Informed Neural Networks. 58, 59, 72, 84, 88

PISO Pressure-Implicit with Splitting of Operators Algorithm. 17

RAM Random Access Memory. 92

RANS Reynolds-Averaged Navier-Stokes. 21, 23, 26, 30, 32, 52–55, 58, 59, 62, 94

ReLU Rectified Linear Unit. 39

RMS Root Mean Square. 25

RNN Recurrent Neural Network. 44, 47, 56

SDF Signed Distance Function. 60

SENPAI Simulated Engineering in New Planning with Artificial Intelligence. 105, 107, 109, 111, 117

SGS Sub-Grid-Scale. 24, 25

SIMPLE Semi-Implicit Method for Pressure Linked Equations. 16, 83

SIMPLEC SIMPLE Consistent. 17

SIMPLER SIMPLE Revised. 17

SMLR Step-Wise Multiple Linear Regression. 56

SVM Support Vector Machines. 34

U-Net U Shaped Neural Networks. 46, 49, 60, 70, 85, 89, 96

URANS Unstable RANS. 52

List of Figures

List of Tables

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Abualdenien, J. and Borrmann, A. (2022). Levels of detail, development, definition, and information need: a critical literature review. *Journal of Information Technology in Construction*, 27:363–392.

Adams, M. D. and Kanaroglou, P. S. (2016). Mapping real-time air pollution health risk for environmental management: Combining mobile and stationary air pollution monitoring with neural network models. *Journal of Environmental Management*, 168:133–141.

Aghamolaei, R., Fallahpour, M., and Mirzaei, P. A. (2021). Tempo-spatial thermal comfort analysis of urban heat island with coupling of cfd and building energy simulation. *Energy and Buildings*, 251:111317.

Allen, S. and Cahn, J. (1972). Ground state structures in ordered binary alloys with second neighbor interactions. *Acta Metallurgica*, 20(3):423–433.

Ansys (2022). Ansys Fluent | Fluid Simulation Software. `https://www.ansys.com/products/fluids/ansys-fluent`. Accessed: 2022-20-09.

Bahlali, M., Dupont, E., and Carissimo, B. (2018). A hybrid cfd rans/lagrangian approach to model atmospheric dispersion of pollutants in complex urban geometries. *International Journal of Environment and Pollution*, 64:74.

Bai, K., Wang, C., Desbrun, M., and Liu, X. (2021). Predicting high-resolution turbulence details in space and time. *ACM Trans. Graph.*, 40(6).

Balduzzi, F., Bigalli, S., and Bianchini, A. (2018). A hybrid BEM-CFD model for effective numerical siting analyses of wind turbines in the urban environment. *Journal of Physics: Conference Series*, 1037:072029.

Baskaran, A. and Stathopoulos, T. (1989). Computational evaluation of wind effects on buildings. *Building and Environment*, 24(4):325–333.

Basodi, S., Ji, C., Zhang, H., and Pan, Y. (2020). Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*, 3(3):196–207.

Blocken, B. (2014). 50 years of computational wind engineering: Past, present and future. *Journal of Wind Engineering and Industrial Aerodynamics*, 129:69–102.

Blocken, B., Janssen, W., and van Hooff, T. (2012). Cfd simulation for pedestrian wind comfort and wind safety in urban areas: General decision framework and case study for the eindhoven university campus. *Environmental Modelling & Software*, 30:15–34.

Blocken, B., Stathopoulos, T., and Carmeliet, J. (2007). Cfd simulation of the atmospheric boundary layer: wall function problems. *Atmospheric Environment*, 41(2):238–252.

Blocken, B., Stathopoulos, T., and van Beeck, J. (2016). Pedestrian-level wind conditions around buildings: Review of wind-tunnel and cfd techniques and their accuracy for wind comfort assessment. *Building and Environment*, 100:50–81.

Boussinesq, J. V. (1903). *Théorie analytique de la chaleur mise en harmonie avec la thermodynamique et avec la théorie mécanique de la lumière. Tome 2, Tome 2,*. Gauthier-Villars et C-ie, éditeurs, Paris.

Bratoev, I., Bonnet, C., Chokhachian, A., Schubert, G., Petzold, F., and Auer, T. (2017). Designing and evaluating district heating networks with simulation based urban planning. In *International Conference on Urban Comfort and Environmental Quality*, pages 167–174.

Bratoev, I., Schubert, G., Chokhachian, A., Bonnet, C., Petzold, F., and Auer, T. (2018). Interactive solar potential simulation for early stage urban planning. In *BAUSIM*.

Brkan, I. (2022). Senpai // urban wind simulation. Idp documentation, Technical University of Munich, Chair of Architectural Informatics.

Brodzicki, A., Piekarski, M., and Jaworek-Korjakowska, J. (2021). The whale optimization algorithm approach for deep neural networks. *Sensors*, 21(23).

Cakir, Y. C. (2022). Senpai // automated geometry generation. Idp documentation, Technical University of Munich, Chair of Architectural Informatics.

Calzolari, G. and Liu, W. (2021). Deep learning to replace, improve, or aid cfd analysis in built environment applications: A review. *Building and Environment*, 206:108315.

Cauchy, A.-L. (1827). De la pression ou tension dans un corps solide. *Exercices de mathématiques*, 2(S 42).

Cermak, J. E. (2003). Wind-tunnel development and trends in applications to civil engineering. *Journal of Wind Engineering and Industrial Aerodynamics*, 91(3):355–370.

Chang, F.-J., Chen, P.-A., Lu, Y.-R., Huang, E., and Chang, K.-Y. (2014). Real-time multi-step-ahead water level forecasting by recurrent neural networks for urban flood control. *Journal of Hydrology*, 517:836–846.

Charney, J. G., Fjörtoft, R., and Neumann, J. V. (1950). Numerical integration of the barotropic vorticity equation. *Tellus*, 2(4):237–254.

Chen, X., Zhao, X., Gong, Z., Zhang, J., Zhou, W., Chen, X., and Yao, W. (2021). A deep neural network surrogate modeling benchmark for temperature field prediction of heat source layout. *Science China Physics, Mechanics & Astronomy*, 64(11):1.

Chien, K.-Y. (1982). Predictions of channel and boundary-layer flows with a low-reynolds-number turbulence model. *AIAA Journal*, 20(1):33–38.

Clerk-Maxwell, J. (1869). Remarks on the mathematical classification of physical quantities,. *Proceedings of the London Mathematical Society*, s1-3(1):224–233.

Danisch, F. and Esch, E. (2022). Senpai // n.o.n.n.e. Idp documentation, Technical University of Munich, Chair of Architectural Informatics.

de Lagrange, J. L. (1773). Sur l'attraction des sphéroïdes elliptiques. *Nouv. Mém. Acad. royale Berlin année*, pages 619–649.

Deardorff, J. W. (1970). A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers. *Journal of Fluid Mechanics*, 41:453–480.

Derickson, R. G. and Meroney, R. N. (1977). A simplified physics airflow model for evaluating wind power sites in complex terrain. In *Summer Computer Simulation Conference*, pages 763–769.

DesignBuilder Software Limited (2022). DesignBuilder. `https://designbuilder.co.uk/`. Accessed: 2022-20-09.

Dong, C., Loy, C. C., He, K., and Tang, X. (2016). Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307.

Doormaal, J. P. V. and Raithby, G. D. (1984). Enhancements of the simple method for predicting incompressible fluid flows. *Numerical Heat Transfer*, 7(2):147–163.

Driller, D. and Kaltenbach, H.-J. (1999). Improvement of the convective outflow condition for turbulent flow simulations. In *APS Division of Fluid Dynamics Meeting Abstracts*, APS Meeting Abstracts, page JO.07.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12(null):2121–2159.

Ehlert, A., Nayeri, C. N., Morzynski, M., and Noack, B. R. (2019). Locally linear embedding for transient cylinder wakes. *arXiv preprint arXiv:1906.07822*.

Euler, L. (1757). Principes généraux du mouvement des fluides. *Mémoires de l'Académie des Sciences de Berlin*, pages 274–315.

Evans, M. W., Harlow, F. H., and Bromberg, E. (1957). The particle-in-cell method for hydrodynamic calculations. Technical report, LOS ALAMOS NATIONAL LAB NM.

Fahssis, K., Dupont, G., and Leyronnas, P. (2010). Urbawind, a computational fluid dynamics tool to predict wind resource in urban area. In *International Conference of Applied Energy, Conference paper*.

Ferziger, J. H., Perić, M., and Street, R. L. (2002). *Computational methods for fluid dynamics*, volume 3. Springer.

Förster, N., Bratoev, I., Fellner, J., Schubert, G., and Petzold, F. (2021). Designing crowd safety - agent-based pedestrian simulations in the early collaborative design stages. In *PROJECTIONS, Proceedings of the 26th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA)*, volume 2, pages 729–738.

Foundation, T. L. (2023). Open neural network exchange - onnx. https://onnx.ai/. Accessed: 19-07-2023.

Franke, J., Hellsten, A., Schlünzen, K., and Carissimo, B. (2007). Best practice guideline for the cfd simulation of flows in the urban environment-a summary. In *11th Conference on Harmonisation within Atmospheric Dispersion Modelling for Regulatory Purposes, Cambridge, UK, July 2007*. Cambridge Environmental Research Consultants.

FSD blueCAPE Lda (2022). blueCFD - Core Project. http://bluecfd.github.io/Core/. Accessed: 2022-20-09.

Gabriel, M., Fellner, J., Reitberger, R., Lang, W., and Petzold, F. (2021). Voxel based method for real-time calculation of urban shading studies. In *Proceedings of the 12th Symposium on Simulation for Architecture and Urban Design (SimAUD), Virtual Event, USC, Los Angeles, CA, USA*, pages 15–17.

Germano, M., Piomelli, U., Moin, P., and Cabot, W. H. (1991). A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics*, 3(7):1760–1765.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Gromke, C., Blocken, B., Janssen, W., Merema, B., van Hooff, T., and Timmermans, H. (2015). Cfd analysis of transpirational cooling by vegetation: Case study for specific meteorological conditions during a heat wave in arnhem, netherlands. *Building and Environment*, 83:11–26. Special Issue: Climate adaptation in cities.

Haghighat, E. and Juanes, R. (2021). Sciann: A keras/tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. *Computer Methods in Applied Mechanics and Engineering*, 373(7553):113552.

Hang, J., Sandberg, M., and Li, Y. (2009). Effect of urban morphology on wind condition in idealized city models. *Atmospheric Environment*, 43(4):869–878.

Hargreaves, D. and Wright, N. G. (2007). On the use of the k–ε model in commercial cfd software to model the neutral atmospheric boundary layer. *Journal of wind engineering and industrial aerodynamics*, 95(5):355–369.

Hasegawa, K., Fukami, K., Murata, T., and Fukagata, K. (2020). Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes. *Theoretical and Computational Fluid Dynamics*, 34(4):367–383.

Hassid, S. and Poreh, M. (1978). A turbulent energy dissipation model for flows with drag reduction. *Journal of Fluids Engineering*, 100:107–112.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Herget, J. (2023). Senpai // urban scale data pipeline. Idp documentation, Technical University of Munich, Chair of Architectural Informatics.

Hirt, C. W. and Cook, J. L. (1972). Calculating three-dimensional flows around structures and over rough terrain. *Journal of Computational Physics*, 10:324–340.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Hoffman, G. H. (1975). Improved form of the low reynolds number k- ϵ turbulence model. *The Physics of Fluids*, 18(3):309–312.

Houzeaux, G., Badia, R., Borrell, R., Dosimont, D., Ejarque, J., Garcia-Gasulla, M., and López, V. (2022). Dynamic resource allocation for efficient parallel cfd simulations. *Computers & Fluids*, 245:105577.

Hugging Face, I. (2023). Huggingface - the ai community building the future. `https://huggingface.co/`. Accessed: 19-07-2023.

Ishugah, T., Li, Y., Wang, R., and Kiplagat, J. (2014). Advances in wind energy resource exploitation in urban environment: A review. *Renewable and Sustainable Energy Reviews*, 37:613–626.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Issa, R. (1986). Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62(1):40–65.

Jiang, Y., Li, X., Luo, H., Yin, S., and Kaynak, O. (2022). Quo vadis artificial intelligence? *Discover Artificial Intelligence*, 2(1):4.

Jóczik, S., Zimborás, Z., Majoros, T., and Kiss, A. (2022). A cost-efficient approach towards computational fluid dynamics simulations on quantum devices. *Applied Sciences*, 12(6).

Kastner, P. and Dogan, T. (2021). Eddy3d: A toolkit for decoupled outdoor thermal comfort simulations in urban areas. *Building and Environment*, page 108639.

KC, A., Whale, J., and Urmee, T. (2019). Urban wind conditions and small wind turbines in the built environment: A review. *Renewable Energy*, 131:268–283.

Kempf, A., Klein, M., and Janicka, J. (2005). Efficient generation of initial- and inflow-conditions for transient turbulent flows in arbitrary geometries. *Flow, Turbulence and Combustion formerly: Applied Scientific Research*, 74(1):67–84.

Ketterer, C. and Matzarakis, A. (2016). Mapping the physiologically equivalent temperature in urban areas using artificial neural network. *Landscape and Urban Planning*, 150:1–9.

Khusni, U., Dewangkoro, H. I., and Arymurthy, A. M. (2020). Urban area change detection with combining CNN and RNN from sentinel-2 multispectral remote sensing data. In *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*. IEEE.

Kim, B., Azevedo, V. C., Thuerey, N., Kim, T., Gross, M., and Solenthaler, B. (2019). Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum*, 38(2):59–70.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kolbe, T. H. (2009). *Representing and Exchanging 3D City Models with CityGML*, pages 15–31. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kondo, K., Murakami, S., and Mochida, A. (1997). Generation of velocity fluctuations for inflow boundary condition of les. *Journal of Wind Engineering and Industrial Aerodynamics*, 67-68:51–64. Computational Wind Engineering.

Korteweg, D. J. and De Vries, G. (1895). Xli. on the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 39(240):422–443.

Laubscher, R. (2021). Simulation of multi-species flow and heat transfer using physics-informed neural networks. *Physics of Fluids*, 33(8):087101.

Laubscher, R. and Rousseau, P. (2021). Application of mixed-variable physics-informed neural networks to solve normalised momentum and energy transport equations for 2d internal convective flow.

Launder, B. and Spalding, D. (1974). The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 3(2):269–289.

Launder, B. E. and Sharma, B. I. (1974). Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters in heat and mass transfer*, 1(2):131–137.

Li, J., Delmas, A., Donn, M., and Willis, R. (2018). Validation and comparison of different cfd simulation software predictions of urban wind environment based on aij wind tunnel benchmarks. In *Proceedings of the Symposium on Simulation for Architecture and Urban Design*, SIMAUD '18, San Diego, CA, USA. Society for Computer Simulation International.

Lin, B., Li, X., Zhu, Y., and Qin, Y. (2008). Numerical simulation studies of the different vegetation patterns' effects on outdoor pedestrian thermal comfort. *Journal of Wind Engineering and Industrial Aerodynamics*, 96(10):1707–1718. 4th International Symposium on Computational Wind Engineering (CWE2006).

Liu, S., Pan, W., Zhang, H., Cheng, X., Long, Z., and Chen, Q. (2017). Cfd simulations of wind distribution in an urban community with a full-scale geometrical model. *Building and Environment*, 117:11–23.

Liu, Y. S., Yigitcanlar, T., Guaralda, M., Degirmenci, K., Liu, A., and Kane, M. (2022). Leveraging the opportunities of wind for cities through urban planning and design: A prisma review. *Sustainability*, 14(18).

Luo, S., Vellakal, M., Koric, S., Kindratenko, V., and Cui, J. (2020). Parameter identification of rans turbulence model using physics-embedded neural network. In Jagode, H., Anzt, H., Juckeland, G., and Ltaief, H., editors, *High Performance Computing*, pages 137–149, Cham. Springer International Publishing.

Macleamy, P. (2004). The future of the building industry. the effort curve.

Mason, P. and Callen, N. (1986). On the magnitude of the subgrid-scale eddy coefficient in large-eddy simulations of turbulent channel flow. *Journal of Fluid Mechanics*, 162:439–462.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

McLean, D. (2022). Integrated environmental solutions ltd. `https://www.iesve.com/`. Accessed: 2022-20-09.

Millar, J., Wayland, R., and Holgate, J. (2020). Hybrid rans-les simulation of a tall building in a complex urban area. In Hoarau, Y., Peng, S.-H., Schwamborn, D., Revell, A., and Mockett, C., editors, *Progress in Hybrid RANS-LES Modelling*, pages 379–388, Cham. Springer International Publishing.

Mochida, A. and Lun, I. Y. (2008). Prediction of wind environment and thermal comfort at pedestrian level in urban area. *Journal of Wind Engineering and Industrial Aerodynamics*, 96(10):1498–1527. 4th International Symposium on Computational Wind Engineering (CWE2006).

Mohan, S. and M.V.S.S, G. (2022). A brief review of recent developments in the integration of deep learning with GIS. *Geomatics and Environmental Engineering*, 16(2):21–38.

Mokhtar, S., Beveridge, M., Cao, Y., and Drori, I. (2021). Pedestrian wind factor estimation in complex urban environments. In Balasubramanian, V. N. and Tsang, I., editors, *Proceedings of The 13th Asian Conference on Machine Learning*, volume 157 of *Proceedings of Machine Learning Research*, pages 486–501. PMLR.

Morozova, N., Trias, F., Capdevila, R., Pérez-Segarra, C., and Oliva, A. (2020). On the feasibility of affordable high-fidelity cfd simulations for indoor environment design and control. *Building and Environment*, 184:107144.

Moshayedi, A. J., Roy, A. S., Kolahdooz, A., and Shuxin, Y. (2022). Deep learning application pros and cons over algorithm deep learning application pros and cons over algorithm. *EAI Endorsed Transactions on AI and Robotics*, 1(1).

Mughal, M. O., Kubilay, A., Fatichi, S., Meili, N., Carmeliet, J., Edwards, P., and Burlando, P. (2021). Detailed investigation of vegetation effects on microclimate by means of computational fluid dynamics (cfd) in a tropical urban environment. *Urban Climate*, 39:100939.

Murakami, S. (1998). Overview of turbulence models applied in cwe–1997. *Journal of Wind Engineering and Industrial Aerodynamics*, 74-76:1–24.

Nakayama, H., Tamura, T., and Okuda, Y. (2005). Les study of fluctuating dispersion of hazardous gas in urban canopy. *EACWE 4*, page 238.

Nakayama, Y., Izawa, K., Aoki, K., and Tippetts, J. (2018). *Introduction to Fluid Mechanics*. Elsevier Science & Technology, San Diego, UNITED KINGDOM.

NEN (2006). Wind danger in the built environment, NEN 8100 Dutch Standard.

Newell, A. and Simon, H. (1956). The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 2(3):61–79.

Obiols-Sales, O., Vishnu, A., Malaya, N., and Chandramowliswharan, A. (2020). CFDNet. In *Proceedings of the 34th ACM International Conference on Supercomputing*, New York, NY, USA. ACM.

Obrecht, C., Kuznik, F., Merlier, L., Roux, J.-J., and Tourancheau, B. (2014). Towards aeraulic simulations at urban scale using the lattice boltzmann method. *Environmental Fluid Mechanics*, 15(4):753–770.

OpenFOAM Foundation, T. (2022). Openfoam. `http://www.openfoam.org/`. Accessed: 2022-20-09.

OpenStreetMap contributors (2022). Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`. Accessed: 2022-18-12.

Orszag, S. A. (1970). Analytical theories of turbulence. *Journal of Fluid Mechanics*, 41(2):363–386.

Østergård, T., Jensen, R. L., and Maagaard, S. E. (2016). Building simulations supporting decision making in early design – a review. *Renewable and Sustainable Energy Reviews*, 61:187–201.

Patankar, S. and Spalding, D. (1972). A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806.

Patankar, S. V. (2018). *Numerical heat transfer and fluid flow*. CRC press.

Paterson, D. A. and Apelt, C. J. (1990). Simulation of flow past a cube in a turbulent boundary layer. *Journal of Wind Engineering and Industrial Aerodynamics*, 35:149–176.

Qin, H., Hong, B., Jiang, R., Yan, S., and Zhou, Y. (2019). The effect of vegetation enhancement on particulate pollution reduction: Cfd simulations in an urban park. *Forests*, 10(5).

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

Ramponi, R., Blocken, B., Laura, B., and Janssen, W. D. (2015). Cfd simulation of outdoor ventilation of generic urban configurations with different urban densities and equal and unequal street widths. *Building and Environment*, 92:152–166.

Reda, E., , Zulkifli, R., Harun, Z., , and and (2017a). Large eddy simulation of wind flow through an urban environment in its full-scale wind tunnel models. *JOURNAL OF MECHANICAL ENGINEERING AND SCIENCES*, 11(2):2665–2678.

Reda, E., Zulkifli, R., and Harun, Z. (2017b). Large eddy simulation of wind flow through an urban environment in its full-scale wind tunnel models. *Journal of Mechanical Engineering and Sciences*, 11(2):2665–2679.

Reynolds, O. (1883). XXIX. an experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels. *Philosophical Transactions of the Royal Society of London*, 174:935–982.

Ribeiro, M. D., Rehman, A., Ahmed, S., and Dengel, A. (2020). Deepcfd: Efficient steady-state laminar flow approximation with deep convolutional neural networks. *arXiv preprint arXiv:2004.08826*.

Richards, P. and Hoxey, R. (1993). Appropriate boundary conditions for computational wind engineering models using the k-ϵ turbulence model. *Journal of wind engineering and industrial aerodynamics*, 46:145–153.

Richardson, L. F. (2007). *Weather prediction by numerical process*. Cambridge university press.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

Russell, S. J. (2010). *Artificial intelligence a modern approach*. Pearson Education, Inc.

Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P., and Aroyo, L. M. (2021). "everyone wants to do the model work, not the data work": Data cascades in high-stakes ai. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA. Association for Computing Machinery.

Santiago, J.-L., Buccolieri, R., Rivas, E., Calvete-Sogo, H., Sanchez, B., Martilli, A., Alonso, R., Elustondo, D., Santamaría, J. M., and Martin, F. (2019). Cfd modelling of vegetation barrier effects on the reduction of traffic-related pollutant concentration in an avenue of pamplona, spain. *Sustainable Cities and Society*, 48:101559.

Schaeffer, J. (2013). *One jump ahead: challenging human supremacy in checkers*. Springer Science & Business Media.

Schrödinger, E. (1926). An undulatory theory of the mechanics of atoms and molecules. *Physical Review*, 28(6):1049–1070.

Schubert, G. (2021). *Interaction Forms for Digital Design*. PhD thesis, Technical University of Munich.

Schubiger, A., Barber, S., and Nordborg, H. (2020). Evaluation of the lattice boltzmann method for wind modelling in complex terrain. *Wind Energy Science*, 5(4):1507–1519.

Shirzadi, M., Naghashzadegan, M., and A. Mirzaei, P. (2018). Improving the cfd modelling of cross-ventilation in highly-packed urban areas. *Sustainable Cities and Society*, 37:451–465.

SimScale GmbH (2022). Simscale. `https://www.simscale.com/`. Accessed: 2022-20-09.

Singaravel, S., Suykens, J., and Geyer, P. (2018). Deep-learning neural-network architectures and methods: Using component-based models in building-design energy prediction. *Advanced Engineering Informatics*, 38:81–90.

Smagorinsky, J. (1953). The dynamical influence of large-scale heat sources and sinks on the quasi-stationary mean motions of the atmosphere. *Quarterly Journal of the Royal Meteorological Society*, 79(341):342–366.

Smagorinsky, J. (1958). ON THE NUMERICAL INTEGRATION OF THE PRIMITIVE EQUATIONS OF MOTION FOR BAROCLINIC FLOW IN a CLOSED REGION. *Monthly Weather Review*, 86(12):457–466.

Smagorinsky, J. (1963). General Circulation Experiments with the Primitive Equations. *Monthly Weather Review*, 91(3):99.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR.

Sola, A., Corchero, C., Salom, J., and Sanmarti, M. (2020). Multi-domain urban-scale energy modelling tools: A review. *Sustainable Cities and Society*, 54:101872.

Sommerfeld, A. (1908). Ein beitrag zur hydrodynamischen erklärung der turbulenten flüssigkeitsbewegungen. *International Congress of Mathematicians*, 3:116–124.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

Sonnenwald, F., Stovin, V., and Guymer, I. (2016). Feasibility of the porous zone approach to modelling vegetation in cfd. In *Hydrodynamic and Mass Transport at Freshwater Aquatic Interfaces: 34th International School of Hydraulics*, pages 63–75. Springer.

Steinmann, F. (2004). *Modellbildung und computergestütztes Modellieren in frühen Phasen des architektonischen Entwurfs*. doctoralthesis, Bauhaus-Universität Weimar.

Stokes, G. G. (2009). *On the Effect of the Internal Friction of Fluids on the Motion of Pendulums*, volume 3 of *Cambridge Library Collection - Mathematics*, page 1–10. Cambridge University Press.

Tarabishy, S., Psarras, S., Kosicki, M., and Tsigkari, M. (2020). Deep learning surrogate models for spatial and visual connectivity. *International Journal of Architectural Computing*, 18(1):53–66.

The Langley Research Center Turbulence Modeling Resource (2022). Turbulence modeling resource. `https://turbmodels.larc.nasa.gov/`. Accessed: 2022-20-09.

Thuerey, N., Weißenow, K., Prantl, L., and Hu, X. (2020). Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36.

Tilki, K. T. (2022). Cdp // step-up from 2d: 3d visualisation of osm data. Idp documentation, Technical University of Munich, Chair of Architectural Informatics.

Tominaga, Y., Mochida, A., Yoshie, R., Kataoka, H., Nozu, T., Yoshikawa, M., and Shirasawa, T. (2008). Aij guidelines for practical applications of cfd to pedestrian wind environment around buildings. *Journal of Wind Engineering and Industrial Aerodynamics*, 96:1749–1761.

Tominaga, Y. and Stathopoulos, T. (2013). Cfd simulation of near-field pollutant dispersion in the urban environment: A review of current modeling techniques. *Atmospheric Environment*, 79:716–730.

Toparlar, Y., Blocken, B., Maiheu, B., and van Heijst, G. (2017a). A review on the CFD analysis of urban microclimate. *Renewable and Sustainable Energy Reviews*, 80:1613–1640.

Toparlar, Y., Blocken, B., Maiheu, B., and van Heijst, G. (2017b). A review on the cfd analysis of urban microclimate. *Renewable and Sustainable Energy Reviews*, 80:1613–1640.

Tripathy, R. K. and Bilionis, I. (2018). Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of Computational Physics*, 375:565–588.

Turing, A. (1950). I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vazquez-Canteli, J., Demir, A. D., Brown, J., and Nagy, Z. (2019). Deep neural networks as surrogate models for urban energy simulations. *Journal of Physics: Conference Series*, 1343(1):012002.

Von Kármán, T. (1931). *Mechanical similitude and turbulence*. Number 611. National Advisory Committee for Aeronautics.

Von Kármán, T. (2004). *Aerodynamics: Selected topics in the light of their historical development*. Dover, Mineola (New York).

Wang, B., Sun, S., and Duan, M. (2018). Wind potential evaluation with urban morphology-a case study in beijing. *Energy Procedia*, 153:62–67.

Weerasuriya, A., Hu, Z., Zhang, X., Tse, K., Li, S., and Chan, P. (2018). New inflow boundary conditions for modeling twisted wind profiles in CFD simulation for evaluating the pedestrian-level wind field near an isolated building. *Building and Environment*, 132:303–318.

Westermann, P. and Evins, R. (2021). Using bayesian deep learning approaches for uncertainty-aware building energy surrogate models. *Energy and AI*, 3:100039.

Whitaker, S. (1986). Flow in porous media i: A theoretical derivation of darcy's law. *Transport in Porous Media*, 1(1):3–25.

Widl, E., Agugiaro, C., and Puerto, P. (2018). FIRST STEPS TOWARDS LINKING SEMANTIC 3d CITY MODELLING AND MULTI-DOMAIN CO-SIMULATION FOR ENERGY MODELLING AT URBAN SCALE. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4:227–234.

Wilcox, D. C. and Rubesin, M. W. (1980). Progress in turbulence modeling for complex flow fields including effects of compressibility. *NASA STI/Recon Technical Report N*, 80:20527.

Wu, Y. and He, K. (2018). Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

Xiangxue, W., Lunhui, X., and Kaixun, C. (2019). Data-driven short-term forecasting for urban road network traffic based on data processing and lstm-rnn. *Arabian Journal for Science and Engineering*, 44(4):3043–3060.

Xie, Y., Franz, E., Chu, M., and Thuerey, N. (2018). tempoGAN: A Temporally Coherent, Volumetric GAN for Super-resolution Fluid Flow. *ACM Transactions on Graphics (TOG)*, 37(4):95.

Yakhot, V., Orszag, S., Thangam, S., Gatski, T., and Speziale, C. (1992). Development of turbulence models for shear flows by a double expansion technique. *Physics of Fluids A: Fluid Dynamics*, 4(7):1510–1520.

Yeh, W., Lin, Y., Liang, Y., and Lai, C. (2021). Convolution neural network hyperparameter optimization using simplified swarm optimization. *CoRR*, abs/2103.03995.

Yousif, S. and Bolojan, D. (2021). Deep-performance - incorporating deep learning for automating building performance simulation in generative systems. In Globa, A., van Ameijde, J., Fingrut, A., Kim, N., Tian, T., and Lo, S., editors, *Proceedings of the 26th Conference on Computer Aided Architectural Design Research in Asia (CAADRIA)*, CAADRIA proceedings, pages 151–160. CAADRIA.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Zhong, J., Liu, J., Zhao, Y., Niu, J., and Carmeliet, J. (2022). Recent advances in modeling turbulent wind flow at pedestrian-level in the built environment. *Architectural Intelligence*, 1(1).