MACHINE LEARNING FOR CARBON FIBER REINFORCED POLYMER PRODUCTION

Simon Stieber

Dissertation
zur Erlangung des Doktorgrades
doctor rerum naturalium (Dr. rer. nat.)
der Fakultät für Angewandte Informatik
der Universität Augsburg, 2023

Universität
Augsburg
University

Erstgutachter:	Prof. Dr. Wolfgang Reif, Universität Augsburg
Zweitgutachter:	Prof. Dr. Bernhard Bauer, Universität Augsburg
Drittgutachter:	Assoz.Prof. Dr.mont. Ewald Fauster, Montanuniversität Leoben
Tag der mündlichen Prüfung:	1. Dezember 2023

Abstract

Fiber reinforced polymer composites offer a range of properties that are essential for many applications where the highest performance demands are placed on high-end components. These include targeted stiffness combined with lightness, and they are also corrosion resistant. As a result, they have become an integral part in the respective products for aerospace, automotive, construction and sports equipment, among others. However, these outstanding properties come at a price: the production of composite materials is significantly more complex and thus more expensive than, for example, metal or aluminum castings. The complexity is inter alia related to a lower degree of automation, which can be traced back to strongly fluctuating starting materials, especially in the case of textile semi-finished products, and the associated manual work steps.

This is the starting point of the present work. Different possibilities for the analysis and optimization of a process family for fiber composites are presented: Liquid Composite Molding. Machine Learning methods are used for this purpose. The following scientific contributions are made: Thus, (1) ways to reconstruct the flow front and detect defects at runtime based on convolutional neural networks are discussed. Furthermore, it is shown how future progress of a process could be based on the course of this injection process in order to be able to counteract if necessary. In addition, the (2) properties of the textile are generated by different variants of neural networks, CNNs, ConvLSTMs and Transformers, based on the progress of each injection process as 2D maps and stored as a digital twin. These maps can be used for post-processing verification of the components or as a quality feature for further processing or life cycle of the product. Finally, (3) Reinforcement Learning, and neural network methods are used to control variants of an injection process so that fewer defects occur. These applications show the possibilities of Machine Learning in the context of fiber composite production, and possibilities how to work (4) with deep neural networks despite data poverty are shown. This includes, above all, sim-to-real transfer learning, in which models are “pre-trained” on data from the simulation and then retrained or “fine-tuned” with a small amount of real data. Thus, deep networks can be preconditioned with large simulative data sets to be fine-tuned to the real case.

Kurzfassung

Faserverbundwerkstoffe bieten eine Reihe an Eigenschaften, die für viele Anwendungsbereiche, in denen höchste Ansprüche an die Leistung von High-end-Bauteilen gestellt werden, essenziell sind. Dazu gehören zielgerichtete Steifigkeit bei gleichzeitiger Leichtigkeit, außerdem sind sie korrosionsbeständig. Somit sind sie unter anderem in den Bereichen Luft- und Raumfahrt, Automotive, Bau- und Sportgeräte ein integraler Bestandteil in den jeweiligen Produkten geworden. Diese hervorragenden Eigenschaften haben jedoch ihren Preis: die Herstellung von Verbundwerkstoffen ist deutlich aufwendiger und somit teurer als z.B. Metall- oder Aluminiumguss. Dieser Aufwand hängt mit einem geringeren Automatisierungsgrad zusammen, der sich auf stark schwankende Ausgangsmaterialien, v.a. bei textilen Halbzeugen und damit zusammenhängenden händischen Arbeitsschritten zurückführen lässt.

An dieser Stelle setzt die vorliegende Arbeit an. Es werden verschiedene Möglichkeiten zur Analyse und zur Optimierung einer Prozessfamilie für Faserverbundwerkstoffe vorgestellt: der Flüssigharzimprägnierung. Hierbei werden Methoden des maschinellen Lernens verwendet. Folgende wissenschaftliche Beiträge werden geleistet: So werden (1) Möglichkeiten zur Rekonstruktion der Fließfront und der Detektion von Fehlstellen zur Laufzeit basierend auf Convolutional Neural Networks diskutiert. Des Weiteren wird gezeigt, wie auf der Basis des Verlaufs eines Einspritzvorgangs der künftige Fortschritt sein könnte, um bei Bedarf gegensteuern zu können. Zudem werden die (2) Eigenschaften des Textils durch verschiedene Varianten von neuronalen Netzen, CNNs, ConvLSTMs und Transformer, basierend auf dem Verlauf des jeweiligen Einspritzprozesses als 2D Karten erzeugt und als digitaler Zwilling gespeichert. Diese Karten können zur Nachüberprüfung der Bauteile oder als Qualitätsmerkmal für die weitere Verarbeitung oder den Lebenszyklus des Produkts genutzt werden. Abschließend werden mit Hilfe von (3) Methoden des bestärkenden Lernens und neuronaler Netze Varianten eines Einspritzprozesses geregelt, so dass weniger Fehlstellen auftreten. Diese Anwendungen zeigen die Möglichkeiten des maschinellen Lernens im Kontext der Faserverbundwerkstoffproduktion auf. Es werden zudem (4) Möglichkeiten aufgezeigt, wie trotz Datenarmut mit tiefen neuronalen Netzen gearbeitet werden kann. Hierzu zählt vor allem das Sim-to-Real Transfer Learning, bei dem Modelle auf Daten aus der Simulation "vortrainiert" werden und danach mit wenigen Echtdaten nachtrainiert bzw. "fein abgestimmt" werden. Somit lassen sich tiefe Netze mit großen simulativen Datensätzen vorkonditionieren, um dann auf den Echtfall fein angepasst werden zu können.

Acknowledgments

I would like to thank my advisor Wolfgang Reif for his support throughout the years of my work at the Institute for Software and Systems Engineering at the University of Augsburg. He helped focusing on the important aspects of my doctoral project and gave hope and perspective when there were unforeseen events. Alexander Schiendorfer gave me motivation to follow through on the route of this doctorate. In his postdoctoral role he helped me bringing together single strains of thoughts to knit them together towards useful scientific ideas. He helped shaping mere textual fragments towards well-crafted publications. My (former) colleagues, Niklas Schröter, Julia Krützmann and Carola Lenzen supported with deep technical discussions and helpful feedback. I would like to thank them for the cheerful work together over the many years at this chair. I want to thank all my student assistants, that carried a lot of work on their shoulders, namely Leonard Heber, who drove the work on Reinforcement Learning for RTM.

Ewald Fauster was a big inspiration regarding work ethic and technical accuracy, I always enjoyed working with him since it often felt like fast-paced game of ping pong, when we exchanged thoughts and gave each other input. He also introduced me to the world of fiber reinforced polymer and the engineering mastery behind it. I also would like to thank Marcel Bender for his corrections and also would like to thank everybody else that helped me crafting this thesis with their feedback. I want to thank Christof Obertscheider, who helped me implementing a simulation that made Reinforcement Learning possible. He also gave me general insights into numerical simulation.

I would like to thank my parents, Cordula and Rudolf, for always being supportive throughout my complete studies, in harder and in easier times. They taught me to be curious, but also supportive of others to make them reach their potential. Lastly, I want to thank my lovely wife, Melanie, for enduring me during the writing of this thesis and always carrying me through depths and enjoying the heights of this doctorate with her warmth and love.

Simon Stieber

Contents

Chapter 1

Introduction

Summary. This chapter introduces the challenges in manufacturing Carbon Fiber Reinforced Polymer (CFRP), a composite material widely used in various industries due to its desireable properties. In general, manufacturing of CFRP involves combining fibers and polymer resins and there are several methods for manufacturing CFRP, with Liquid Composite Molding (LCM) being one of the most prominent and the focus of this thesis. LCM involves injecting liquid resin into a mold containing dry fibers, offering advanced automation possibilities. However, it also poses challenges like precise control of resin flow and handling of process-induced errors. The manufacturing of CFRP is further complicated by the inherent characteristics of the base materials, including variability in the textile base material and the anisotropic nature of CFRP. These factors make CFRP manufacturing a challenging endeavor that requires careful consideration of process parameters and material behavior. The stages of CFRP manufacturing are briefly outlined to provide context for the work that will be discussed subsequently. These phases include design, simulation, manufacturing, and testing. The chapter concludes by discussing the usefulness of Machine Learning (ML) for LCM. ML methods are suitable for LCM due to the availability of data, the ability to label data, the repeatability of the process, and the complexity of the process. To close the chapter, the scientific contributions of this thesis are summarized.

1.1 Challenges in Manufacturing Carbon Fiber Reinforced Polymer

Carbon Fiber Reinforced Polymer (CFRP) is a composite material that has gained significant attention in various industries due to its high strength-to-weight ratio, corrosion resistance, and durability. These industries include inter alia the production of aircraft, aerospace, automotive, and sports equipment, but also construction [Neitzel et al., 2014; Toepker, 2011]. Lightweight CFRP products are also more energy efficient than their metal counterparts, which is important for several industries, where fuel consumption and CO_2 emissions are important factors [Heywood et al., 2015]. Whenever carbon fibers are involved, the composites are named CFRP. The more general term is Fiber Reinforced Polymer (FRP). The manufacturing of FRP involves the combination of fibers and polymer resins, resulting in a material that exhibits superior mechanical properties. The manufacturing process of FRP presents several challenges that need to be addressed to ensure the production of high-quality composites.

There is a variety of methods for manufacturing FRP, each with its own set of advantages and limitations. The choice of manufacturing process depends on factors such as the desired shape, size, and application of the FRP component. Some of the most common processes include hand lay-up, pultrusion, and LCM which includes Resin Transfer Molding (RTM), Vacuum Assisted Resin Infusion (VARI), and Vacuum Infusion. LCM always involves the placement of fibers in a mold, followed by the infusion of resin and curing under controlled conditions. RTM features a non-transparent mold and higher injection pressures. VARI is a variant of RTM that uses a vacuum to assist the distribution of the resin. Vacuum Infusion processes only rely on vacuum without applying pressure at the inlet and have one half mold made of foil, which is often transparent. To summarize, LCM processes feature a spectrum of different process configurations. They are one of the most prominent FRP component manufacturing methods and are the primary focus of this thesis from the process side. The resin flows through the fiber preform, impregnating the fibers and forming a composite material upon curing. LCM offers advantages such as reduced labor costs and better possibilities for automation, improved resin distribution, and the ability to produce complex geometries. However, LCM also presents challenges such as the need for precise control of resin flow, prevention of void formation, correct orientation of fibers, and handling of process-induced errors.

FRP manufacturing is further complicated by the inherent characteristics of the base materials. The permeability of the same base textile can vary up to 20 % [Tifkitsis and Skordos, 2020; May et al., 2019]. The anisotropic nature of FRP, which results from the alignment of fibers, leads to variations in material properties depending on the direction of loading. The correct curing process of the matrix is crucial as well: the interaction between the fibers and the resin matrix can give rise to defects such as delamination, porosity, and fiber waviness. These factors make the manufacturing process of FRP a challenging endeavor that requires careful consideration of process parameters and material behavior. The complications become even more palpable when comparing FRPs to metals, which are homogeneous materials. In the course of LCM processes, a wide range of inaccuracies can adversely affect the final product, leading to increased overall production costs [Sorg, 2014]. These issues are primarily due to high input variances associated with the preform.

1.2 High Variability Requires Adaptive Manufacturing Processes

Variability in the textile base material encompasses differences in material properties, such as natural or recycled fibers, variations across batches, handling effects on the fibrous preform, and process-related effects like race-tracking. To achieve quality specifications, such as complete mold filling or minimal void content, and to improve process efficiency, including reducing mold filling time or identifying the optimal curing temperature, the application of active control methods is essential.

To do so, monitoring LCM processes with sensors has become crucial. E.g. dry spots can irreparably invalidate the stability and stiffness properties required for the manufactured part. Process monitoring based on sensors applied to the mold significantly improves the quality assurance – called in-situ monitoring. These sensors are able to track the progression of the fluid and, consequently, can be used to predict *spatial* deviations from proper RTM runs, which feature an opaque mold. This analysis may indicate problems such as dry spots, providing diagnostic insight or even control actions to avoid rejects. For other LCM processes, that feature a transparent upper mold, Computer Vision (CV) methods play a big role in monitoring the flow front. When the system is equipped with a camera, images of the flow front progress can be automatically analyzed [Matsuzaki et al., 2013]. To supervise the curing state of the matrix, Dielectric Analysis (DEA) sensors [Faber et al., 2021] and ultrasonic sensors [Achzet et al., 2023] are suitable. To monitor the state of the textile, several research works have been published: we inferred the material properties from the injection process itself [Stieber et al., 2023b], while others measured fiber properties prior to and after the process [Heuer et al., 2015].

1.3 Stages of CFRP Production: Design, Simulation, Manufacturing and Testing

The production of CFRP parts is a complex process that involves several steps: most importantly, design, simulation, manufacturing and testing. The method presented in this work draws from all of these steps, but focuses on the manufacturing process.

Design Phase The design phase is the initial phase in the manufacturing of (C)FRP products. During this phase, specific software is used to create detailed models of the product. Computer-aided Design (CAD) models are used to visualize and analyze the product, and to evaluate its performance and manufacturing feasibility with a mechanical simulation on the basis of CAD. During the design phase, a variety of factors is taken into account, including: the material properties of the FRP, the manufacturing processes used to produce the product [Denkena et al., 2014; Sundin and Björkman, 2016], and the performance requirements of the end user. This information is then utilized to improve the design of the product, and to ensure that it meets the requirements of the application. For example, the CAD model can be used to evaluate the strength and stiffness of the product in conjunction with Finite Element Analysis (FEA) software in the simulation phase, and to optimize the design to maximize these properties [Qi et al., 2019]. The manufacturability of the product can also be evaluated with this model, and the design can be optimized to ease manufacturing and minimize costs [Denkena et al., 2014]. By carefully considering the material properties, manufacturing processes, and performance requirements, the design of the product is optimized to meet the needs of the end user. In this work, the CAD designs and models are used in simulations to produce large amounts of data that are used to train ML models.

Simulation Phase Following the design of the component, the simulation phase is the second stage. During this phase, FEA software is used to analyze and optimize the performance of the product. FEA is a numerical method that allows to simulate the behavior of the product under various loads and conditions, and evaluate its performance and reliability. During the simulation phase, the CAD model of the product is taken as input to the FEA software. They define the boundary conditions, loads, and material properties of the product, and run the simulation to evaluate its behavior. The FEA software then solves a system of differential equations that describes the behavior of the product to be manufactured, and gives a detailed analysis of its performance. For example, FEA is used to evaluate the stress and strain distribution in the product, and to identify areas where the design may lead to failure of the component [Qi et al., 2019; Higuchi et al., 2020]. FEA can also be utilized to evaluate the natural frequencies and modes of the product, and to optimize the design to reduce unwanted vibrations or resonances [Zhang et al., 2020]. Another important use case for FEA in the context of this work is the simulation of the fluid flow during the manufacturing process [Obertscheider and Fauster, 2023; ESI Group, 2023]. Here, the simulation does not calculate the behavior of the product but rather the behavior of the fluid flow in the manufacturing process. This type of simulation is greatly leveraged to produce vast amounts of data necessary for large ML models and is discussed in more detail in Section 3.2. To summarize, by carefully analyzing the behavior of the product under various loads and conditions, the design can be improved. It can be ensured that the final product meets the performance requirements of the end user. Additionally, massive amounts of data can be produced, which is crucial for ML.

Manufacturing Phase In the third stage, concerning manufacturing, the product is fabricated using a variety of techniques. These techniques involve the use of specialized equipment and processes to manufacture the product with the correct dimensions, tolerances, and material properties. During this phase, process control software is run to monitor and control the production of the product [Neitzel et al., 2014]. This software allows to collect data from the manufacturing process, such as temperature, pressure, and flow rate, and use this data to optimize the production [Stieber et al., 2020]. For an exemplary machinery and sensor layout see Figure 2.5 and Figure 2.4 for the corresponding data architecture.

The approach in this thesis is to collect this data and use it to train and evaluate the ML models. When considering the state of the art in process control, we have to differentiate between industrial and academic state of the art. Within industry, LCM, especially RTM is carried out with preset parameters to the best of knowledge, without interference or control within the process. [Neitzel et al., 2014] In the academic world, there have been several attempts to monitor the resin injection in LCM processes, and to adjust the injection rate and pressure by applying process control software to ensure that the product is manufactured with the correct material properties. We conducted a literature survey [Fauster et al., 2023] on that topic, which will be discussed in more depth in Subsection 2.3.3. The cure kinetics of the product can also be monitored with such a software, and the temperature and pressure of the mold or the injection pressure can be adjusted to ensure that the product is fully cured and has the desired mechanical properties. The process data collected from the machines, controls, and specific sensors can be utilized for ML. A system that is based on Reinforcement Learning (RL) and simulation data to control the resin injection in RTM processes is evaluated in Chapter 7.

Testing and Inspection Phase Finally, the product undergoes testing and inspection to verify that it meets the performance requirements and quality standards. The testing phase is the final phase in the life-cycle of FRP products. The tests may include physical testing, such as tensile testing or fatigue testing, as well as visual inspection and non-destructive testing. This testing often also happens multiple times as part of a larger production line, as control gates whether single parts are ok, need rework or go to scrap [Sorg, 2014]. For example, tensile testing machines are used to apply tensile loads to the product, and measure its response in terms of strain and stress [Miyano et al., 2000]. Fatigue testing machines are utilized to apply cyclic loads to the product, and measure its response in terms of fatigue life and crack growth. Additionally, the quality and integrity of the product can be measured using visual inspection and non-destructive testing techniques [Towsyfan et al., 2020]. Visual inspection involves the use of cameras, microscopes and other imaging devices to examine the surface and internal structure of the product, and identify defects such as voids, delaminations, or inclusions. Non-destructive testing involves the use of techniques such as X-ray, ultrasound, or eddy current testing to evaluate the internal structure of the product without damaging it. In-situ sensor signals can be utilized to reconstruct process features and detect anomalies or problems (such as dry spots within LCM) within a process cycle [Stieber et al., 2021b]. In general, all the data gathered during the testing phase can be used to collect labeling information for ML models, which are mainly trained in a supervised manner.

To summarize, the life-cycle of FRP products involves a combination of design, simulation, manufacturing, and testing to produce high-performance, lightweight products that meet the performance and quality requirements of the end user. All of the phases can be utilized for either collecting data for or to apply (trained) ML models.

1.4 Usefulness of Machine Learning for Liquid Composite Molding

To be able to apply ML to a process, several preliminaries have to be fulfilled.

1. There must be a case for ML, i.e. the process has to be too complex to be solved by classical methods or training an ML model is cheaper than manually adapting a conventional solution.
2. Data of the process has to be available.
3. For supervised learning, the data has to be labeled, i.e. the data has to be associated with a label.
4. The process has to be repeatable and the data has to be reproducible.

All these requirements are fulfilled in the LCM of CFRP products. As described before, data collection (2) and labeling (3), which is a crucial step in the ML process, is done inherently during the manufacturing process of CFRP products. This is due to the fact that the manufacturing process is automated and data is collected from sensors and other devices and labels are gathered from the destructive or non-destructive testing, which makes it also inherently repeatable (4). The most important point of the preliminaries listed above is (1), making sure there is a use for ML methods for the LCM process. This is the case, as the LCM process is highly complex and involves many different steps, which makes it hard to solve with classical methods. Its complexity stems from the inherent fuzziness of textiles, that show diverse behavior in different manufacturing batches, but also from textile piece to textile piece, as described above. This makes it difficult to predict the behavior of the material, which is crucial for the LCM process and hard to set up this manufacturing process with a fixed set of parameters. For other manufacturing processes, e.g. casting metal, the material properties are well known and within a smaller corridor, which makes it easier to use fixed parameters [Sorg, 2014]. For LCM, ML methods are the key to predict the behavior of the material or the process itself and optimize it accordingly. Another aspect that is crucial for the application of ML methods is their speed. If a fluid simulation is accurate enough, it could also be used as a model of the process to analyze it or to use it for control. The problem that fluid simulations face is that they get very slow with increasing accuracy. ML models, on the other hand, especially Neural Networks (NNs) can keep a high inference speed for a limited set of parameters that they were trained on. This opens up the possibility for them to act as a fast surrogate model trained on fluid simulation models.

1.5 Scientific Contribution

We discuss the application of ML methods on manufacturing CFRP, a topic that has had some but not as much exposure to this type of data-driven methodology as other domains such as automated driving or language translation. Sorg [2014] presented a data-driven method to optimize the manufacturing of automotive CFRP roofs and Mendikute [Mendikute et al., 2021, 2022, 2023] described methods to use ML for impact performance prediction on CFRP. More precisely, the scientific contribution of this work is as follows:

- To begin with, an introduction to (C)FRP, the manufacturing methods we focus on, and the reasoning why ML methods are not only suitable, but sensible to use for a family of manufacturing processes that has to adapt to strong deviations in the textile base material, is given in Chapter 2. In Chapter 3, the theoretical foundations of the methods in use are laid out: ML, fluid simulations, and the definition of an ML-based digital twin. A more hands-on introduction into the basis of the process, the materials and data sets that were specifically produced and analyzed for this thesis, is given in Chapter 4. To show the possibilities of ML for CFRP, we delve into three different applications in the following three Chapters: 5, 6, and 7.
- One central approach of this work is to use both simulated data and data coming from the real process for *sim-to-real transfer learning*, an approach that was initially introduced in robotics [Peng et al., 2018; Zhao et al., 2020] and also had applications in processes related to LCM: injection molding [Tercan et al., 2018]. Due to limited access to real data on one hand and abundant data from simulation on the other, this approach of first training a model on a big simulation dataset and then retrain it with only a small amount of real data is worthwhile to adapt the model to the real world and thus bridge the simulation-reality gap.
- The first application of ML for CFRP is the *analysis of LCM processes* with the goal of detecting *dry spots*, based on sensory data. This was first published by Stieber et al. [2021b] with simulated data, achieving an accuracy of 91.68 %. This approach, described in detail in Chapter 5, was expanded to use real data for sim-to-real transfer learning as well. Furthermore, we predicted the future fluid flow within the LCM process based on the course of the current injection, again with both simulated and real data and with the approach of learning based on simulation first and then on real data.
- The second application is an *ML-driven digital twin of product properties*, illustrated in Chapter 6. Published by Stieber et al. [2021a, 2023b], this approach tests the capabilities of reconstructing 2D maps of textile material parameters from different ML models. These textile parameters are crucial indicators of the quality of the finished product and the 2D maps serve as the digital twin of the product properties. For this endeavor, we not only implemented sim-to-real transfer learning, but also conducted an experimental campaign on the uses of this approach, i.e. we tested how much (expensive) data from reality is necessary for which level of prediction quality given larger simulation datasets for pre-training. The best metrics on pixelwise accuracy of 95.93 % and Intersection over

Union (IoU) of 0.5031 were achieved when using the best model with sim-to-real transfer learning on 80 real samples.

- Lastly we tested the possibilities of *controlling LCM processes with RL methods.* After reviewing the research landscape for LCM control [Fauster et al., 2023], and finding only control engineering approaches, we conducted experiments on actively controlling the fluid flow by changing the input pressure at different inlets with an RL agent, published by Stieber et al. [2023a]. This approach has the advantage of not depending on an explicit environment model, but learning from rewards coming from a reward function. It can also handle high-dimensional input, e.g. images or multiple sensor signals at once. Our best model managed to reduce the amount of dry spots in a certain test set from 54 % to 27 %.

The contributions of this thesis are the result of research published in seven related and peer-reviewed papers the author of this thesis co-authored, six of which constitute the backbone of this dissertation. Two additional papers are still under peer-review at the time of writing. The research was funded partially by the Bavarian State Ministry of Economic Affairs and Media, Energy and Technology (StMWi) within the project MAI CC4 CosiMo (Composites for sustainable Mobility).

Chapter Summary and Outlook
This chapter introduced the challenges in manufacturing CFRP, a composite material widely used in various industries due to its desirable properties. There are several methods for manufacturing CFRP, with LCM being one of the most prominent and the focus of this thesis, which always involve combining fibers and polymer resins. LCM does so by injecting liquid resin into a mold containing dry fibers, offering great automation possibilities. However, it also presents challenges like precise control of resin flow and handling of process-induced errors. The manufacturing of CFRP is further complicated by the inherent characteristics of the base materials, including variability in the textile base material and the anisotropic nature of CFRP. These factors make CFRP manufacturing a challenging endeavor that requires careful consideration of process parameters and material behavior. The production of CFRP products involves several steps, including design, simulation, manufacturing, and testing. The chapter concludes by discussing the usefulness of ML for LCM. ML methods are suitable for LCM due to the availability of data, and the ability to label it, the repeatability of the process, and the complexity of the process. The scientific contributions of this thesis were summarized at the end of the chapter and will be unfolded in the following.

Chapter 2

LCM Monitoring and Control

Summary. In Chapter 2, a detailed analysis of FRP manufacturing processes is presented, with a spotlight on LCM. The discussion kicks off with an exploration of how FRP components are generally manufactured: by combining a polymer matrix with load bearing reinforcing fibers. This process is highly adaptable, allowing the properties of FRP composites to be tailored to fit a wide range of applications based on the choice of resin, reinforcement, and manufacturing technique. Going into detail on LCM, the process family's specifics are examined. A critical aspect of producing high-quality parts, the importance of achieving a uniform resin flow throughout the entire mold, is highlighted. The pitfalls of non-uniform flow are also addressed, with a focus on "dry spots", defects that can compromise the structural integrity of the finished part. The discussion then shifts to the materials used in FRP and LCM, providing an introduction to the material science behind FRP. This includes an explanation of the properties of the materials themselves, from single fibers to preforms and resins or matrices for FRP, as well as a discussion on general material behavior, encompassing the concepts of isotropy and anisotropy. Highlighting the challenges in LCM that necessitate adaptiveness, the text motivates the transition from static to adaptive processes. In-situ monitoring is introduced as a strategy that employs sensors to track fluid flow and predict real-time deviations. This automated process monitoring significantly enhances the quality assurance of the LCM process. The text concludes with a discussion on the optimization of LCM processes, distinguishing between passive and active methods. While passive methods optimize various process parameters in advance, active methods control certain parameters based on real-time measurements. The potential of Artificial Intelligence (AI)-based methods in enhancing control techniques for LCM is underscored, highlighting approaches in optimizing manufacturing processes. These potentials regarding both analysis and control of LCM processes are evaluated in the later application chapters of this thesis: 5, 6 and 7.

Publication. The concepts and results presented in this chapter are published partly in [Stieber et al., 2020; Wilfert et al., 2021; Fauster et al., 2023].

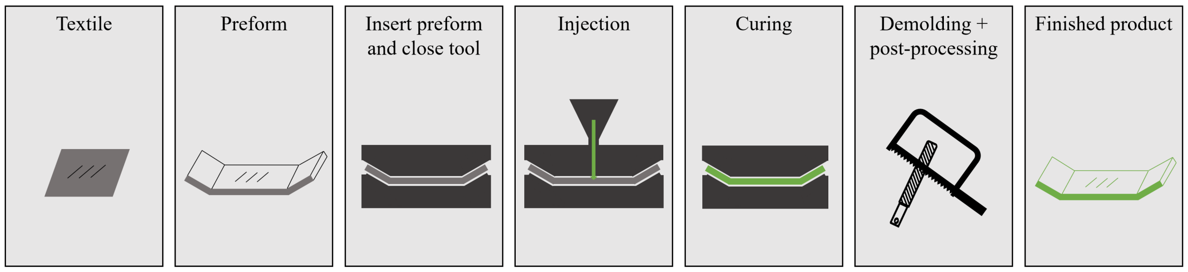


Figure 2.1: Overview of the RTM process, after Neitzel et al. [2014]

2.1 Manufacturing Processes for Fiber Reinforced Polymer Components

To be able to discuss LCM in general and the monitoring and control aspects in particular, some preliminary remarks regarding the manufacturing of FRP have to be introduced.

FRP composites are made by combining a polymer resin matrix with strong reinforcing fibers. The properties of FRP composites can be tailored to fit nearly any type of application, depending upon the choice of resin, reinforcement, and the way the product is manufactured. The different processes can cover different ranges of production volume, targeted price, part complexity and size. [Neitzel et al., 2014]

These different process types include the following: For low production volumes, *Hand Layup* [Elkington et al., 2015] can be used, it is the simplest method of fabricating FRP: fiber and resin are manually transferred into a mold, which is very labor intensive and thus used for low volume production of e.g. FRP boats, surfboards or wind turbine blades. *Filament Winding* is a process where continuous fiber filaments are impregnated with resin and wound around a rotating mandrel in a predetermined pattern. This process is used for producing cylindrical or spherical parts with high strength and stiffness, such as pipes, tanks, and pressure vessels with medium to high volumina. *Pultrusion* is a continuous process where fiber rovings are pulled through a resin bath and then through a heated die that shapes and cures the composite. This process is used for producing long and constant cross-section profiles, such as rods, bars, beams, and channels. For these specific types of components, high production volumes can be achieved. *Liquid Composite Molding* techniques are some of the most well-known and cost-effective out-of-autoclave manufacturing methods for CFRP components production for complex shapes parts. They are the main focus of this work and are described in more detail in the following.

2.1.1 LCM as a Family of Processes, RTM as Part of LCM

LCM methods include RTM [Babb et al., 1998] and Vacuum Infusion, inter alia[Neitzel et al., 2014; Rajak et al., 2021]. In the RTM procedure, overpressure is used to inject the matrix material into the mold. Once the fiber reinforcement is placed within the mold cavity, it is sealed, and a reactive mixture of resin, hardener, and occasionally additives, is injected into the cavity. This creates a “flow front” which differentiates the saturated from unsaturated material, depicted in both scenarios in Figure 2.2. The reinforcement structure gets impregnated by the mixture, which subsequently cures, typically under elevated temperature (refer Figure 2.1). The typical cycle times for a RTM process vary between 5 to 25 minutes, contingent on the

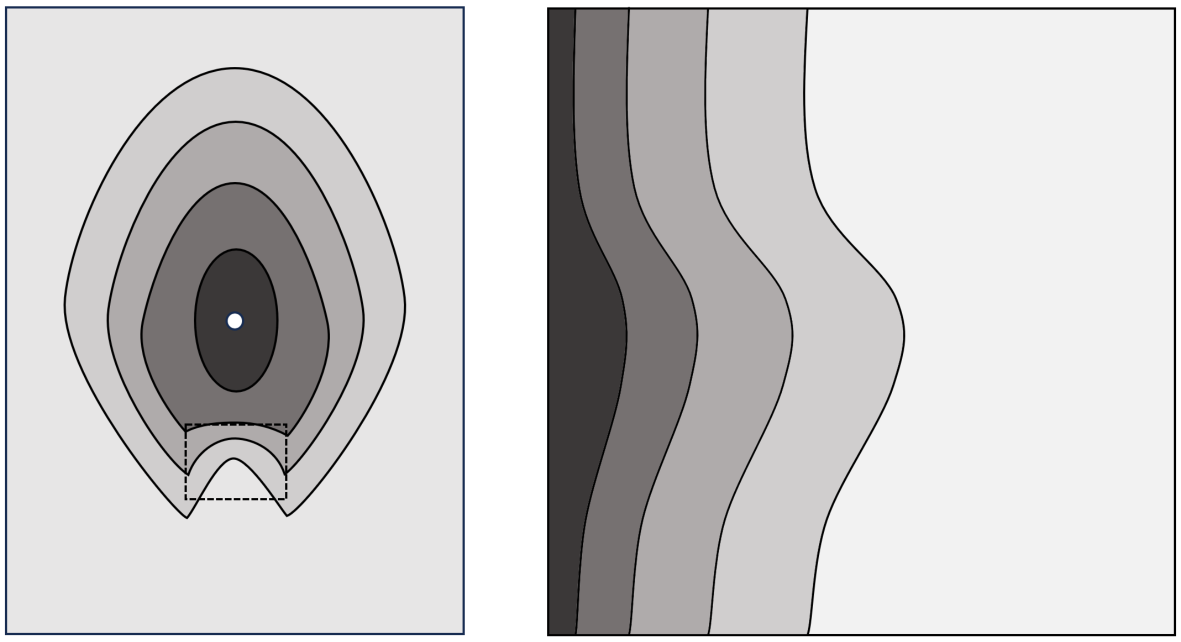


Figure 2.2: Schematic flows of both case studies, both including deviations from the perfect flow front. Left: Case study A: Schematic flow front progress with one central injection port (or sprue), including deviation from perfect flow front on the bottom, resulting in dry spot. Right: Case study B: injection from the side including deviation of the flow front: faster progress in the middle.

part's size and complexity [Neitzel et al., 2014]. For simpler shapes, the injection and the curing process, consume the majority of the time. With a single mold, series sizes can reach up to 50, 000 units annually [Neitzel et al., 2014]. Due to this high output, the initial investment in the process machinery and tooling can be distributed over many components and thus the process becomes economical. The RTM process allows for the creation of components with intricate geometries. The procedure enables the fabrication of dimensionally accurate molded parts, reducing the requirement for further post-processing. Within the RTM process, there exist several subprocesses, such as High-Pressure RTM — where exceptionally high injection pressures ($\approx$ 200 bar [Rosenberg, 2018]) are used — and Compression RTM, where the mold is not fully closed initially, leaving a gap for the matrix to spread. Consequently, during compression, the textile only has to permeate in the z-direction. Both these subprocesses reduce cycle times, but add to the process complexity. Another variant is the thermoplastic T-RTM process, which uses thermoplastic matrix instead of a thermoset.

During Vacuum Infusion [Hammami and Gebart, 2000], a resin also has to pass through a preform, with the difference to RTM that a vacuum is applied at the outlets instead of a higher pressure at the inlet to force the resin into the textile for RTM. It is the most straightforward method to produce continuous CFRP components. This technique relies on a predominantly single-sided mold and a vacuum film, which acts as the upper mold. A vacuum pump and a resin storage tank are the only requirements, keeping initial investment costs to a minimum. At the beginning of the production process, akin to all LCM processes, a semi-finished, textile product is positioned inside the mold, which is then sealed and vacuum is applied. The textile compresses due to the difference between ambient pressure and the pressure beneath the vacuum foil. Upon opening the injection line, the matrix saturates the fiber material, enabled by the vacuum. The vacuum permits a maximum pressure differential of less or equal to 1 bar. As a result, the flow path and the size of the components are inherently limited. Moreover, one process can be quite time-consuming. The vacuum injection process, however, is particularly well-suited for fabricating components with intricate geometry. The typical series size is not fixed, with quantities ranging from a handful of units per year to around 3000 units annually. [Neitzel et al., 2014] Both process types are similar to a certain extent, as described above, there are also hybrid process forms between both: RTM with added vacuum for better fluid flow and lower chances of trapped air bubbles, but also vacuum with slight overpressure at the injection point.

Another relevant procedure that is not used for producing finished FRP components but for testing material properties beforehand, are permeability measurements, described in more depth in Subsection 2.2.5. This process is similar to the injection phase of LCM since a fluid is injected into a textile to measure the advancement of the flow front and hence calculate the permeability in different dimension. In Vacuum Infusion, the upper mold is a foil, which is transparent in many cases and thus the flow front is directly visible. For this thesis a 2D permeameter was utilized to produce a real world dataset.

The resin can be injected in various ways, through different inlet configurations. For this work, we focus on LCM and in particular on RTM. We discuss two different types of injection scenarios (cf. Figure 2.2), described in detail in Section 4.1: both feature a rectangular mold. "Case study A" has a central injection point, thus the resin has to flow to all four sides. "Case study B" features a linear injection on one side and the fluid has to flow to the opposing side.

2.2 FRP Introduction

To be able to discuss the LCM processes, an introduction to the material science behind FRP in general is necessary. That is why in the following the materials themselves, from single fibers to preforms and resins or matrices for FRP are introduced. Also, general material behavior, i.e. the (an-)isotropy of textiles is discussed. Lastly, the step from static to adaptive processes that is vital to make FRP production more efficient is motivated.

2.2.1 Reinforcing Fibers

Reinforcing fibers can be classified according to various criteria: Structure (inorganic or organic), by manufacture or production (artificial or natural) or by properties (high-strength or high-stiffness) [Neitzel et al., 2014]. The three types of fibers that were used in the experiments for this thesis are presented briefly in the following.

Glass fibers hold great significance as a primary reinforcing fiber in many technical applications due to their remarkable cost-efficiency. Glass fibers are created by melting various oxides that act as network formers and network converters. To prevent crystallization, the latter are rapidly cooled. Multiple methods exist for its production, resulting in different types of fiberglass possessing distinct qualities.

Carbon fibers, renowned for their exceptional mechanical properties, have gained prominence across numerous fields of application. Alongside their mechanical prowess, they exhibit high thermal and electrical conductivity. Carbon fibers are technical fibers that are either manufactured or treated within a temperature range of 1000 to 2000 °C, for graphite fibers these temperatures go up to 2500 °C. They possess a carbon content ranging from 92 to 99.90 %

by weight. The base material for carbon fibers are so-called precursors, i.e. organic fibers or pitch. Nowadays Polyacrylnitril (PAN)-variants are quasi-standard as precursor material. As for glass fiber, there are different types and qualities for carbon fibers.

Natural fibers encompass a diverse range of materials derived from animals (e.g., wool), plants, or minerals (e.g., basalt). Among them, plant fibers play a crucial role in the composites industry. They can be categorized into bast fibers obtained from plant stems (such as flax, hemp, kenaf, and jute) and hard fibers sourced from leaves, fruits, and other plant parts (such as sisal or coconut). The current focus on utilizing these fibers as reinforcement materials stems from two key factors. Firstly, natural fibers offer significant potential for lightweight applications due to their low density, typically around 1.5 g/cm^3. Secondly, there is a growing environmental consciousness as renewable raw materials like plant fibers do not contribute to global warming through CO_2 emissions. This heightened environmental awareness and the accompanying marketing potential of "natural" composites has contributed to the increased interest in employing natural fibers as reinforcement materials. Natural fibers vary in length and diameter since they are a natural product. This leads to several issues for production — strong changes in the base material — and also reduces the possible applications of natural fiber based composites: they can be mainly used for applications with reduced mechanical requirements. [Neitzel et al., 2014]

2.2.2 Matrix Systems

The fibers provide strength and stiffness, while the matrix binds the fibers together and transfers stresses between them. The matrix in FRP is most commonly thermoset or thermoplastic polymers.

Thermosets are the most common matrix systems in the industry [Neitzel et al., 2014]. They are a type of polymer that undergo an irreversible chemical reaction when cured or hardened. This makes it harder to recycle them. They have a three-dimensional network of bonds, which provides good thermal stability and chemical resistance. Common thermosets used in FRP include epoxy, polyester, and vinyl ester resins. These materials are usually provided as a liquid that can impregnate the fibers, then cured at room temperature or elevated temperatures. The curing process creates a hard, rigid matrix that securely bonds the fibers together.

Thermoplastics are polymers that can be melted and reshaped multiple times without significantly degrading their properties. This makes thermoplastic FRPs potentially recyclable, which is a significant advantage over thermosets. However, using thermoplastics as a matrix can be more challenging due to the higher processing temperatures required and the difficulty in impregnating the fibers. Common thermoplastics used in FRP include polypropylene, polyethylene, and nylon.

Caprolactam is a monomer used in thermoplastic polymer production. It was the matrix of choice for project CosiMo (cf. Subsection 2.3.4). It is interesting for FRP manufacturing because it is a liquid at room temperature and has low viscosity, making it relatively easy to impregnate the fibers. In the "anionic polymerization" process, ϵ-caprolactam is transformed into Nylon-6 [Kurt, 2021] inside the mold, impregnating the fibers and forming the matrix in a single step. This process, called *in-situ polymerization*, can be advantageous because it does not require high pressures or temperatures, unlike many other thermoplastic processing methods.

For this thesis, we mostly assumed fluids with constant viscosity. For the real data from the permeameter, a certain plant oil was used and for the simulations, constant viscosities were assumed. More details are given in the respective Chapters 5 to 7.

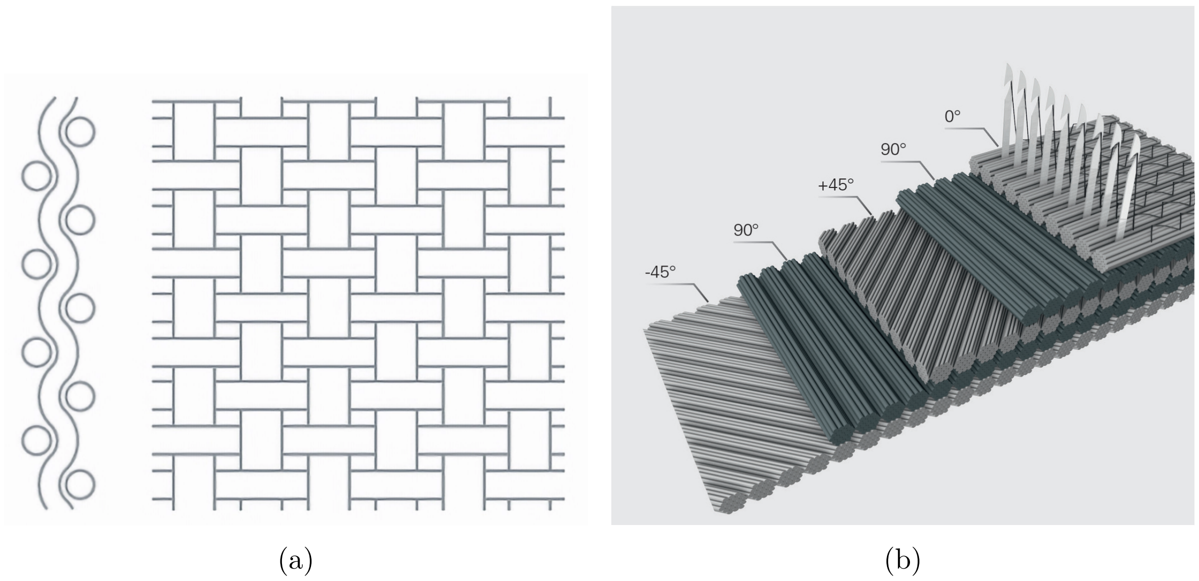


Figure 2.3: Difference between wovens and NCF: (a) woven with plain weave showing strong undulations (visible clearly on left part of figure). (b) NCF with different directionalities — in 45 ° steps — and only small reinforcement threads in black, visible on the far right. Images from [Hacotech GmbH, 2023]

2.2.3 Textiles

Based on the single fibers described in Subsection 2.2.1, semi-finished textile products need to be assembled for further processing. There are several possible textile structures to be produced, of which three that are the most important and used within the context of this thesis. They are described in the following: woven fabrics on one hand, and veils, fleece, and non-crimp fabrics (NCF), on the other, which belong to non-wovens.

Wovens — Conventional weaving technology allows for the transformation of continuous reinforcement fibers into flat semi-finished products. This same weaving technique is utilized in various textile applications, including clothing production. Different types of weaving methods are employed, chosen based on the specific requirements of the application and the type of fiber being processed. Here, the fibers are oriented, which leads to better mechanical properties in the finished product. One issue that wovens face is undulation (or crimping) coming from the interweaving of the fiber bundles, which leads to worse directionality of the fibers (cf. Figure 2.3 (a)). Wovens have, depending on the type of weaving technique, good stability and are easy to handle.

Veils, which are random fiber mats and belong to the group of *non-wovens*, are highly prevalent semi-finished textile products and find extensive usage, particularly in glass mat thermoplastics (GMT) and sheet molding compounds (SMC). They are also employed in LCM, as they readily absorb resin and facilitate the flow of materials. They were also the main textile in project CosiMo (cf. Subsection 2.3.4). Additionally, random fiber mats are utilized as cover layers during component manufacturing to enhance surface quality. In these applications, typical fiber volume content, also represented as φ_f (FVC) ranges from 30 to 40 %. As the name suggests, the fibers in these mats are placed randomly and do not follow certain directions opposed to the other types of semi-finished products presented in the following.

NCF distinguish themselves from woven fabrics by the absence of undulations in the fiber bundles when they are laid down. This unique characteristic enables optimal utilization of the mechanical properties of the reinforcing fibers, as they can be directly loaded along the direction of the fibers, that are not crimped. Moreover, NCF provide opportunities for incorporating various fiber orientations in all directions (cf. Figure 2.3 (b)). The number of individually layered surfaces in NCF depends on the available laying units, resulting in the production of multiaxial fabrics or NCF. Fabrics, in general, offer excellent versatility in adjusting fiber quantity and orientation, making them highly suitable for high-performance applications.

2.2.4 Preforms and Ply Stacking Sequences

The property profile of an FRP component is heavily influenced by its fiber reinforcement structure. The provision of the fiber framework, or *preform*, essentially dictates the process steps' selection and sequence, and, in the end, impacts production time and costs. A preform refers to a non-impregnated fiber structure that has already been positioned and fixed according to the desired fiber content and orientation of the component. This is subsequently impregnated with a matrix in a processing procedure, thereby transforming it into a consolidated FRP component.

The configuration of the preform can be saved in a so-called *ply stacking sequence*, which is also referred to as "ply book", especially in a simulation context [Beyrle et al., 2017]. The ply stacking sequence refers to the order and orientation of the plies — layers of reinforcement including different textiles, but also styrofoam or honeycomb reinforcements for sandwich components — within the mold. This sequence can significantly affect the mechanical properties [Ogunleye et al., 2022], manufacturability, and performance of the final part. There are several aspects to consider: orientation of the fibers, the balance and symmetry, sequencing, shear strength and thickness, which all can heavily influence the form of the flow front.

All these factors combined give a detailed plan on the expected material properties of the preform and subsequently the finished component. In this thesis, different assumptions about the level of information regarding the material properties of the preform are made. For some use cases, this information is taken as input: prediction of future flow fronts (Section 5.2) and also for some observation types in Chapter 7 on control through RL. The idea behind having the information available only on occasion comes from the fact that the planned material properties from ply stacking sequences are generally a good starting point for a model, but can differ from the actual material properties due to handling errors, that can result in e.g., crimps or areas of fringed textile. This can result in incomplete information about the actual properties, which is the motivation to measure them in-situ or during the process as shown in Chapter 6. Because when the actual properties are known, conclusion about the mechanical properties of the finished part and thus the overall quality can be drawn.

2.2.5 Darcy's Law and Permeability - Physical Basics of Impregnation

Following the description of individual fabric types, textiles, preforms, and the matrix material that flows through them during LCM processes, the physical principles of fluid flow through porous media are described. The process of impregnation involves saturating a dry fiber structure with a matrix material. The flow through the fiber preform can be likened to the flow of an incompressible fluid through a porous medium.

The governing equations for fluid flow are derived for an infinitesimal fluid particle [Versteeg and Malalasekera, 2007]. A fluid particle is a volume of constant mass which is traveling with the fluid. Incompressible flow is described by the continuity equation (2.1) and the Navier-Stokes equation (2.2):

$$\nabla \cdot \mathbf{v} = 0 \tag{2.1}$$

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla P + \eta \nabla^2 \mathbf{v} + S \tag{2.2}$$

In this context, $\nabla\cdot$ is symbolic of the divergence operator, while the velocity vector of the fluid is depicted by $\mathbf{v}$. ρ is the mass density of the fluid. In many cases, especially in low-speed flows, it can be considered constant. $\frac{D\mathbf{v}}{Dt}$ is the partial derivative of the velocity vector $\mathbf{v}$ with respect to time. $\rho\frac{D\mathbf{v}}{Dt}$ then represents the rate of change of momentum per unit volume. ∇P is the gradient of the pressure field P. η represents the dynamic viscosity of the fluid, which quantifies its resistance to shear flow. $\nabla^2 v$ is the Laplacian of the velocity field, which is the divergence of the gradient of the velocity vector field. The continuity equation states that the mass of a fluid in an infinitesimal volume which is fixed in space is conserved, i.e. can change only by inflow and outflow through the boundaries of the infinitesimal volume. In many cases, especially in low-speed flows, fluids can be considered incompressible. For incompressible fluids with constant mass density ρ it reduces to divergence of the velocity vector field equal to zero.

The Navier-Stokes equation is a statement of Newton's second law applied to fluid dynamics. It says that the rate of change of momentum in a fluid particle equals the sum of forces on the fluid particle. The left-hand side describes the change of momentum per unit volume. The terms on the left-hand side are forces per unit volume resulting from pressure gradient and viscous dissipation in the fluid. The negative sign in front of the pressure gradient indicates that the pressure force acts against the direction of increasing pressure. Viscous forces are proportional to the dynamic viscosity and the Laplacian of the velocity vector field.

For flow through porous media the velocity is the superficial velocity and the source term S has the form

$$S = \frac{-\eta}{K}\mathbf{v} \tag{2.3}$$

The superficial velocity is related to the physical velocity via the porosity of the porous medium. This is a result of applying the volume control method [Slattery, 1981]. This method takes into account the average length expansion of a control volume. On one hand, the control volume must be chosen sufficiently small to obtain the necessary level of information about the process being described. On the other hand, it should provide a reasonable simplification with respect to the geometrically determined micromechanical characteristics of the process. Assuming that the flow velocity of the matrix within the reinforcement structure is relatively small, the inertial forces in (2.2), i.e., the left side of the equation, can be neglected. This and other assumptions, that are made regarding the flow through textiles, are summarized in Table 2.1. As a result, (2.2) simplifies to the well-known form called the Stokes equation [Versteeg and Malalasekera, 2007]:

$$0 = -\nabla P + \eta \nabla^2 \mathbf{v} \tag{2.4}$$

Table 2.1: Scope of Darcy's Law, adapted from [Neitzel et al., 2014]

Boundary condition	**Physical Meaning**
Steady flow	Reynolds number < 1; Neglect of d'Alembert's inertial forces of the fluid
Newtonian behavior of the fluid	Viscosity is independent of the shear rate in the fluid
Incompressible fluid	Constant density
Isothermal process	Constant viscosity
Mechanically rigid fiber preform	Permeability is a constant
No capillary action in the fiber structure	Only driving force is the pressure gradient

Isolating the velocity vector gives

$$\mathbf{v} = -\frac{K}{\eta}\nabla P \tag{2.5}$$

with volume-average flow velocity $\mathbf{v}$, fluid viscosity η, permeability tensor K, and pressure gradient ∇P. The proportionality between flow velocity and the driving force caused by the pressure gradient, as depicted in (2.5), was empirically determined by Darcy in 1856 and is known as Darcy's law. He investigated the flow of groundwater in sandy soil layers [Darcy, 1856] and derived the proportionality described in (2.5). The material constant representing the permeability of the porous medium is denoted as *permeability* and is assigned the symbol K in (2.5) and is a tensorial quantity. The simulation of fluid flow for LCM relies heavily on Darcy's law and is described more thoroughly in Section 3.2 under the assumptions listed in Table 2.1.

The planar, anisotropic permeability tensor K is modeled as follows for 2-dimensional flow (which is the major relevant flow for this thesis):

$$K_{2D,aniso} = \begin{bmatrix} k_x & k_{xy} \\ k_{xy} & k_y \end{bmatrix} \tag{2.6}$$

k_x and k_y represent the permeability in the x and y directions, respectively, whereas k_{xy} describes how the flow in one major direction is dependent on a pressure gradient in the other major direction.

Permeability Measurements Following the definition and mathematical explanation of permeability, this paragraph will explore the experimental assessment of permeability, and explain the relevance for this thesis. There exist disparities between theoretical and practical considerations in permeability determination, with Darcy's law often accepted as a quick, simple, reliable, and adequately precise computational method in practice. [Neitzel et al., 2014] Different methodologies can be categorized by their dimensionality: 1D, 2D, and 3D. For this study, the 2D measurement is of paramount importance, as it was used to gather the majority of the empirical data to be utilized in the following Chapters 5 and 6. In 1D permeability measurement, it's presumed that the principal axes directions of the given fiber semi-finished product are pre-known, allowing the textile to be arranged for feasible 1D tests along both principal axes, one after another. A defined number of individual layers is cut out along one main axis of the fiber semi-finished product and positioned into a prepared mold for this purpose. Typically, this mold comprises a metal base plate, a cavity frame, and a transparent top plate. The cavity frame enables the setting of a specific cavity height and corresponding FVC. The transparent top plate is crucial for accurately determining the flow front progress visually at any given moment. In this kind of measurement, permeability is calculated from the flow front progress. Given that both the injection pressure and the ambient pressure within the mold are known, permeability can be mathematically determined.

While 1D measurements are straightforward to implement and evaluate mathematically, they encounter two main issues: race-tracking and fiber wash-out. The flow front often leads along the mold's edges in 1D permeability measurements, a phenomenon naturally occurring in other injection processes as well. This occurrence, referred to as the "race-tracking" effect [Bickerton et al., 1998], will be discussed in more detail in Subsection 4.5.1. Another potential effect is the "fiber wash-out", which refers to the displacement of fiber bundles within the mold when the FVC and consequently the compaction pressure is too low and the resin flow is too strong acting on the fiber bundles from the side. To circumvent the edge effects seen in 1D permeability measurement, the 2D permeability measurement was devised. Its primary distinction from the 1D permeability measurement is that the 2D measurement does not use a line gate for injection but rather injects from the center of the fiber semi-finished product via a point gate. This eradicates the race-tracking (at least at the cavity frame's edge) as it reduces the fiber wash-out issues found in 1D permeability measurement and the need to measure in three directions to ascertain the in-plane permeabilities and the orientation angle β. This β refers to the angle at which fibers or filaments are oriented relative to a reference direction in a fabric or the later composite material.

This reduces the experimental effort for the overall in-plane permeability determination from three measurements in the 1D experiment to one measurement in the 2D experiment. An accurate understanding of the flow front position over time is also necessary for 2D permeability measurement. In Chapter 5 and Chapter 6, we employ data from a 2D optical permeameter (cf. Figure 4.4). Here, the flow front is visually tracked, and permeability is determined along the two principal axes.

For completeness, the 3D permeability measurement is mentioned here in short. It is considerably more complex, leading to its limited use in the academic and industrial world. There are several methods to measure out-of plane permeability and an International Benchmark Exercise (IBE) regarding these measurement methods has been conducted by Yong et al. [2021]. For the means of this thesis, these did not offer additional value and were to complex.

2.2.6 Flow in Isotropic and Anisotropic Preforms

In general, isotropy means uniformity in all orientations. For FRP this is particularly interesting regarding the permeability K, since it can have different values in different directions opposed to FVC, which has only value for a certain area. When considering the flow though isotropic textiles, e.g. veils tend to show quasi-isotropic behavior, these material properties are directly related to each other. For instance, increasing the FVC results in lower in-plane permeability, and these changes are equivalent in both principal directions.

Anisotropic FRP show physical properties that are directionally dependent. FRP, especially when reinforced with carbon fibers are used for high strain, high tech applications. One feature that makes their application so interesting, when knowing specific directional loads, is their anisotropy: FRP can be engineered to have high strength and stiffness in specific directions, making them ideal for applications where loads or stresses are primarily applied in such ways, e.g. aircraft, cars, or spacecraft. The anisotropic nature of FRP allows for tailoring of material properties to meet specific requirements. For example, the fiber orientation, type of fiber, and type of resin can all be adjusted to create a composite with the desired strength, stiffness, weight, thermal expansion, and other properties. Modelling the flow through anisotropic materials in a realistic way is challenging and subject to several influences, which are described in Subsection 4.5.1.

2.3 From Static to Adaptive Processes

To advance LCM, to make it more efficient, ecological and usable for a wider range of products, the processes need to be become adaptive. In the following, first the challenges for LCM that make adaptiveness necessary are presented, then the sensory and monitoring means are briefly introduced to finally give a short introduction to the optimization of LCM.

2.3.1 Challenges in LCM

Given the intricacies that LCM processes offer, as introduced beforehand, it does not surprise that there are several challenges for these manufacturing processes. [Neitzel et al., 2014]

Resin Flow Achieving uniform resin flow throughout the entire mold is crucial for producing high-quality parts. Non-uniform flow can lead to areas of incomplete resin impregnation, resulting in defects known as “dry spots”. These can compromise the structural integrity of the finished part. The detection and furthermore the avoidance of these spots is one of the main topics in this thesis, covered in Chapter 5 and Chapter 7.

Resin Cure The degree and rate of resin curing, is another significant challenge. Inadequate or uneven curing can result in parts with poor mechanical properties or dimensional stability. We did not cover this topic in our works, since the temperature and cure cannot be controlled quickly during the process and needs to be determined beforehand [Kurt, 2021].

Fiber Orientation As described before, the position and orientation of the fibers is thoroughly planned to achieve maximum performance. Misalignment — e.g., from handling errors — can lead to decreased strength and stiffness in the composite part. The challenge is to maintain the fiber orientation during the resin injection and curing process. We managed to determine material properties in-situ during the process, which correlates with the fiber orientation (cf. Chapter 6).

During the course of LCM processes, a multitude of inaccuracies can compromise the end product, thereby escalating the total production cost [Sorg, 2014; Heuer et al., 2015; Liu et al., 1996]. Such issues predominantly arise from high input variances associated with the preform.

2.3.2 In-situ Monitoring and Analysis

To overcome these challenges, monitoring LCM processes with sensors would be crucial. In some cases, dry spots can irreparably invalidate the stability and stiffness properties required for the manufactured part. In others, they can be repaired manually. Either way, automated process monitoring based on sensors applied to the mold would significantly improve the quality assurance – called in-situ monitoring. These sensors (e.g., pressure, temperature, DEA or ultrasound [Achzet et al., 2022]) are able to track the flow front of the fluid and, consequently, predict *spatial* deviations from proper RTM runs. This may indicate problems such as dry spots or voids (i.e., enclosures where air is trapped), providing diagnostic insight or even control actions to avoid rejects. For other LCM processes, that feature a transparent upper mold, CV methods play a big role in monitoring the flow front. When the system is equipped with a camera, images of the flow front progress can be automatically analyzed [Matsuzaki et al., 2011, 2013]. This is also valid for optical permeameter experiments [Fauster et al., 2018, 2019]. To monitor the curing state of the matrix, especially DEA sensors [Faber et al., 2021], but also ultrasonic sensors [Achzet et al., 2023] are suitable. To monitor the state of the textile, several research works have been published: Stieber et al. [2023b] inferred the material properties from the injection process itself, as shown in Chapter 6. There are other possibilities to measure fiber properties a priori and a posteriori [Heuer et al., 2015].

2.3.3 Optimization and Control

As described before, the manufacturing process of FRP using LCM encompasses a variety of variability sources. These include differences in material properties in the base material e.g., natural or recycled fiber, across batches, handling impacts on the fibrous preform, and process effects such as race-tracking. This necessitates the application of active control methods to meet part quality specifications, such as thorough mold filling or minimal void content, as well as to enhance process efficiency, including minimizing mold filling time or determining optimal curing temperature.

In a literature review [Fauster et al., 2023], we provide a comprehensive overview of the existing scientific landscape related to passive and active control techniques for LCM, with a special emphasis on AI-based methods. Previous work on the optimization of the LCM processes can be divided into passive and active methods. Passive methods are used to optimize various process parameters in advance, with no further influence being exerted during ongoing production, they focus on the design stage of the process. Many authors used genetic algorithms to minimize the filling time [Landi et al., 2020; Okabe et al., 2017; Jiang et al., 2001] or prevent dry spots [Oya et al., 2020; Okabe et al., 2017; Jiang et al., 2002]. Since the focus of this thesis lays on actively optimizing LCM processes, we delve into further detail here.

The active, or online, methods are applied during the process to control certain parameters based on real-time measurements. Thus, unforeseen disturbances can be reacted to. This review identified three primary groups of research addressing aspects like process design, mold filling, and thermoset curing. We elaborated on specific examples and provided further insights for each of these categories. For active control, which is relevant for Chapter 7, the following early works were very interesting: Demirci and Coulter [1994a,b, 1996]; Demirci et al. [1997] focused on controlling the flow pattern by manipulating the mass flow rate using a forward model and NN. Nielsen and Pitchumani [2000, 2001, 2002b,a] aimed at controlling the filling time and volume flow rate using NN-based process.

2.3.4 Research Project CosiMo

The main initial inspiration to apply ML methods to an RTM process came from research project CosiMo[1]. The project's objective was to develop manufacturing procedures aimed at bulk fabrication of composite components, intended for utilization in the car and airplane manufacturing sectors. One main goal was to leverage ML techniques and smart tools to enable effective and sustainable production [Stieber et al., 2020]. With the use of a NN, process quality was meant to be monitored and anomalies such as dry spots were intended to be detected.

Figure 2.4 shows the actual architecture of data acquisition in the project is shown. There were different sensory and manufacturing systems — the RTM system itself with the press from Wickert, the T-RTM system from Krauss-Maffei, DEA sensors from Netzsch, conventional pressure and temperature sensors, and acoustic sensor technology from the University of Augsburg — that were brought together in a commercial data aggregation tool, the IBA PDA. Eventually the data was stored in a data server from IBA. It was planned the carry back control signals from the models that were trained on the data, but this was not carried out. The models that were developed were trained and tested on recorded data (cf. Subsection 4.1.1).

Figure 2.5 shows the project machinery. The picture on the left shows the T-RTM unit, the press and the tool including sensors. On the top right, a close-up of the plate tool is shown and on the bottom right, the locations of the individual sensors is indicated.

Within CosiMo, several innovations around the RTM process were tested: the textile was a (non-woven) veil, initially planned to be from recycled and natural fibers, introducing a high degree of randomness in the starting material. The matrix was a three component caprolactam that polymerized in-situ. Aside from conventional sensors, DEA sensors and experimental ultrasonic sensors where used in-situ. The data aggregation, analysis and model training were also not of the shelf, but part of the research questions of that project. We utilized the simulation PAM-RTM [ESI Group, 2023] to leverage sim-to-real transfer learning because there was only a small number of real data samples from about 100 cycles within the project. In Subsection 4.1.1, the evaluation of these samples is described in short.

Chapter Summary and Outlook

As this chapter concludes, it provides an outlook on the potential applications and advancements in the field of FRP manufacturing processes, particularly in the context of LCM. In an auxiliary project that optimized the setup of a pultrusion process using genetic programming [Wilfert et al., 2021], we demonstrated the potential of advanced optimization techniques in improving FRP processes, outside the LCM scope. Looking ahead, and with the basics around materials, process, and its optimization in mind, this thesis identifies three main application scenarios that will be explored in the later chapters. The first scenario, discussed in Chapter 5, focuses on the in-process analysis and fault detection of LCM. The second scenario, presented in Chapter 6, explores the development of a digital twin of material properties. The third and final scenario, detailed in Chapter 7, delves into the active control of the LCM process using RL. In summary, the chapter provides a comprehensive overview of FRP manufacturing processes and in LCM in particular, and sets the stage for the exploration of advanced techniques and applications in the subsequent chapters of the thesis.

[1]Project MAI CC4 CosiMo (Composites for Sustainable Mobility), funded by the Bavarian State Ministry of Economic Affairs and Media, Energy and Technology (StMWi)

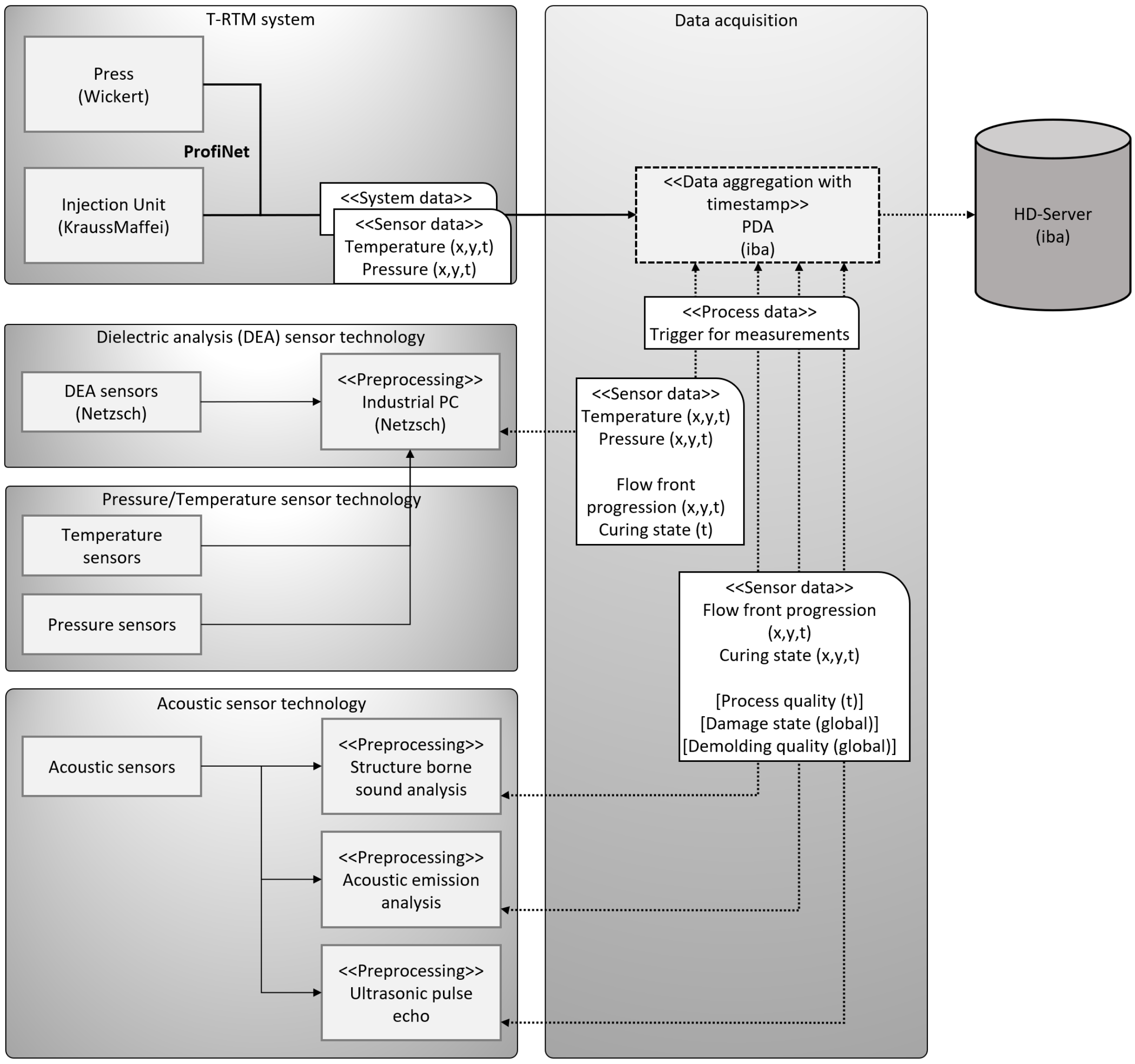

T-RTM system
Press (Wickert)
ProfiNet
Injection Unit (KraussMaffei)
<<System data>>
<<Sensor data>> Temperature (x,y,t) Pressure (x,y,t)
Data acquisition
<<Data aggregation with timestamp>> PDA (iba)
HD-Server (iba)
<<Process data>> Trigger for measurements
Dielectric analysis (DEA) sensor technology
DEA sensors (Netzsch)
<<Preprocessing>> Industrial PC (Netzsch)
<<Sensor data>> Temperature (x,y,t) Pressure (x,y,t) Flow front progression (x,y,t) Curing state (t)
Pressure/Temperature sensor technology
Temperature sensors
Pressure sensors
<<Sensor data>> Flow front progression (x,y,t) Curing state (x,y,t) [Process quality (t)] [Damage state (global)] [Demolding quality (global)]
Acoustic sensor technology
Acoustic sensors
<<Preprocessing>> Structure borne sound analysis
<<Preprocessing>> Acoustic emission analysis
<<Preprocessing>> Ultrasonic pulse echo

Figure 2.4: Data Architecture in Project CosiMo

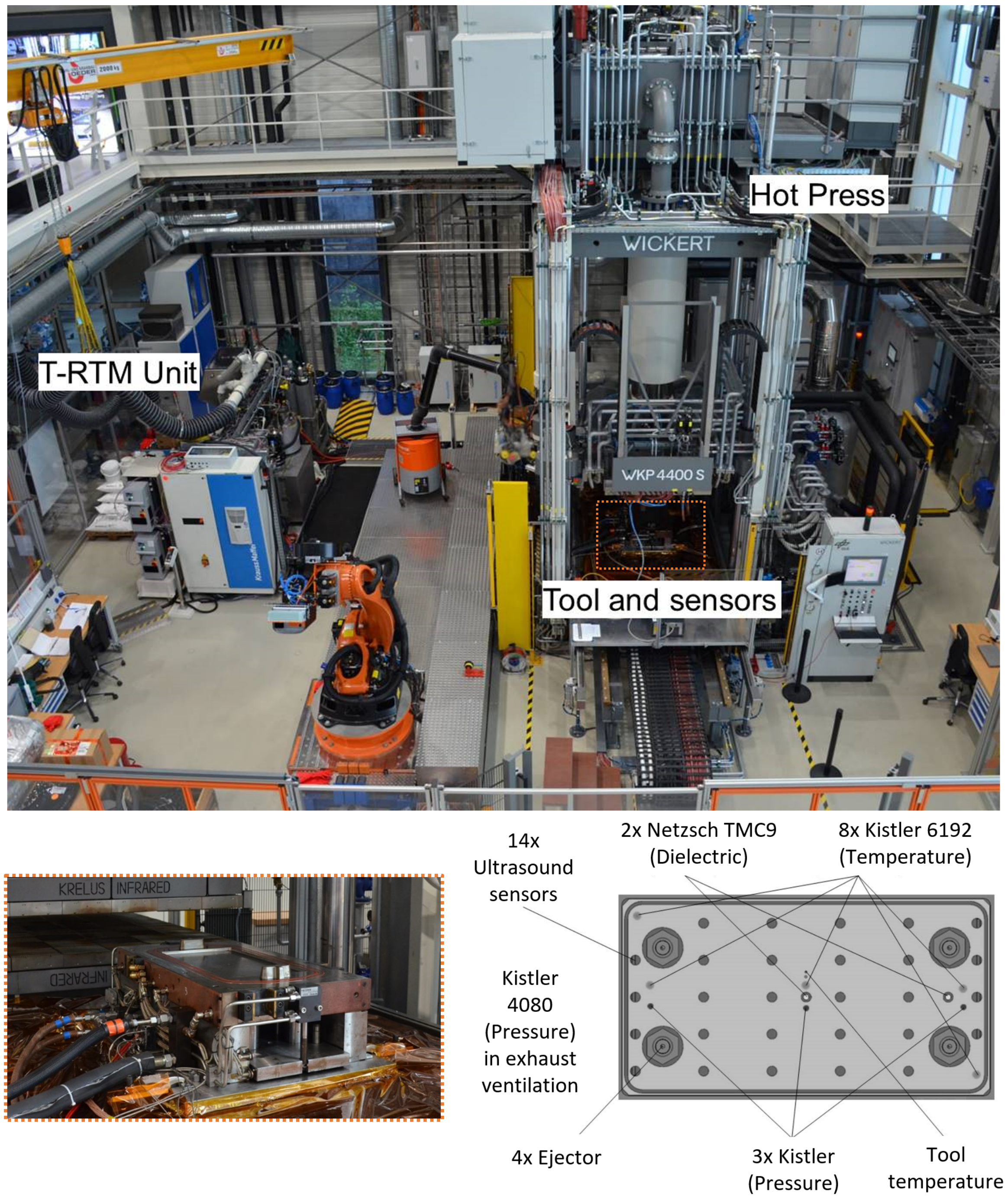


Figure 2.5: RTM machinery and sensors in Project CosiMo: overview on top, bottom left: view of mold equipped with sensors and bottom right: sensor layout in the mold.

Chapter 3

Preliminaries

Summary. In this preliminary chapter, the foundational concepts and methodologies that are employed in the later parts of this thesis are established. We commence with a succinct overview of Machine Learning, with a particular focus on the fundamentals of supervised learning. This sets the stage for the detailed introduction of several neural network architectures that are key to this thesis, which include Convolutional Neural Networks (CNNs), Long Short Term Memorys (LSTMs), Convolutional Long Short Term Memorys (ConvLSTMs) and Transformer models. We then shift our attention to transfer learning, specifically focusing on the transition from simulation to reality. This process involves adapting models trained in a simulated environment for use in real-world situations, which is integral to the areas of application covered in this work. Furthermore, we provide an introduction to Reinforcement Learning, which is necessary for the control of the process described in Chapter 7. Following this, we dive into the principles of LCM simulation. We give an introduction to the basics of fluid simulation, present an overview of various simulation tools, and discuss certain peculiarities in simulation that prove significant in later parts of this thesis. An overview of the two types of simulation environments used in our subsequent applications consolidates this discussion. Finally, the chapter concludes with an exploration of the "digital twin" concept. We offer an introduction to this idea, describe our interpretation of the digital twin, and demonstrate the vital link between ML and simulations within the framework of the digital twin. This understanding is crucial, as the concept of the digital twin plays a pivotal role in the research and applications presented in the subsequent chapters. Throughout this chapter, we aim to equip the reader with a solid foundational understanding of these preliminary topics, enabling a comprehensive grasp of the methodologies and complex systems presented later in this thesis.

Publication. The concepts from this chapter are published in [Stieber, 2018; Stieber et al., 2023a; Fauster et al., 2023; Obertscheider et al., 2023]

3.1 Machine Learning

Machine Learning (ML) is a subset of AI that provides systems the ability to automatically learn and improve from experience without being explicitly programmed [Bishop and Nasrabadi, 2006]. This field focuses on the development of algorithms and statistical models that enable computers to perform specific tasks effectively by leveraging patterns and inference instead of explicit instructions, which is normally the case for programming. At its core, ML is about understanding data and statistics, where the learning process is categorized based on the amount and type of supervision during training. It primarily includes three types of learning methods: supervised learning, unsupervised learning, and RL.

The foundation of ML is built upon theories and concepts from mathematics, statistics, computer science, information theory, and more, which enables the development of various learning algorithms. The use of ML has been growing at an unprecedented rate [Senn-Kalb and Mehta, 2023]. It is used in a vast range of applications including but not limited to recommendation systems, image recognition, Natural Language Processing (NLP), medical diagnosis, financial market analysis, and as in the case of this thesis, industrial applications. As we dive deeper into the realm of ML, we begin to uncover more specific models and techniques. In the context of this preliminary chapter, the focus will be on supervised learning and specific NN architectures, exploring their structures, advantages, and real-world applications. We will also explore the concept of transfer learning, particularly from simulation to reality, a key area of interest in the current ML research landscape. It is especially valuable in domains that do not offer excessive amounts of high-quality data, such as industrial production. The following sections will provide a comprehensive overview of these models and techniques, illuminating their significance in the wider scope of ML and their relevance to the presented research.

3.1.1 Supervised Learning

Supervised learning, a fundamental method of ML, involves learning a function that maps an input to an output based on exemplary input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, every example is a pair containing an input object (typically a vector) and a desired output value. The supervised learning algorithm analyzes the training data and produces an inferred function, which is used for mapping new, unseen examples. This process continues until the model achieves a desirable level of performance. The goal is to learn the mapping function $f : \mathcal{X} \to \mathcal{Y}$ from input space $\mathcal{X}$ to output space $\mathcal{Y}$ such that the prediction error is minimized for unseen data.

There are two major types of problems that supervised learning addresses: regression and classification. Regression problems involve predicting continuous output values (for example, predicting a certain quality criterion of a process, e.g. the surface roughness after a drilling process [Agarwal et al., 2022]), whereas classification problems involve predicting discrete output values (for example, distinguishing between good and bad manufactured parts [Stieber et al., 2021b]). Supervised learning relies heavily on the quality and quantity of the input data, as the accuracy of its output often hinges on the comprehensiveness of its training. It has been applied successfully to a wide range of applications that offer a great amount of data, from CV tasks such as image classification and object detection, to NLP tasks such as sentiment analysis and machine translation. Supervised learning has several advantages: it offers predictive accuracy when the models are trained with enough data and it is also versatile, since it can be applied to different problems.

Mathematical Concepts First, a brief overview on the mathematical concepts behind supervised learning is given.

1. The *hypothesis function*, $h_{\boldsymbol{\theta}}(x)$, represents what the model has learned from the input data. For example, in simple linear regression, it can be represented as

$$h_{\boldsymbol{\theta}}(x) = \theta_0 + \theta_1 x \tag{3.1}$$

, where $\boldsymbol{\theta} = [\theta_0, \theta_1]$ are parameters the model learns.

2. The *loss function*, $J(\boldsymbol{\theta})$, measures the error between the predicted and actual values. For regression tasks, Mean Squared Error (MSE) is often employed:

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - h_{\boldsymbol{\theta}}(x^{(i)}))^2 \tag{3.2}$$

N is the number of samples. For classification tasks, the Cross-Entropy Loss is a common loss function. In binary classification, it is given by:

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}[y^{(i)}\log(h_{\boldsymbol{\theta}}(x^{(i)})) + (1 - y^{(i)})\log(1 - h_{\boldsymbol{\theta}}(x^{(i)}))] \tag{3.3}$$

And for multi-class classification:

$$J(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_j^{(i)}\log(h_{\boldsymbol{\theta}}(x_j^{(i)})) \tag{3.4}$$

where K is the number of classes, $y_j^{(i)}$ is the j-th class label for the i-th instance, and $h_{\boldsymbol{\theta}}(x_j^{(i)})$ is the model's predicted probability for the i-th instance belonging to the j-th class.

3. *Gradient Descent* is an optimization algorithm used to minimize the loss function. The parameters are updated as follows:

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\boldsymbol{\theta}) \tag{3.5}$$

where α is the learning rate.

Despite its advantages, supervised learning also has limitations. The requirement for labeled data can be a significant barrier, as labeling data is often expensive and time-consuming [Fredriksson et al., 2020]. This challenge will later be discussed in the section on transfer learning (cf. Subsection 3.1.2). Moreover, supervised learning models can overfit to the training data, i.e., they can learn the noise in the training data to the extent that it negatively impacts the performance of the model on new data. These concepts are fundamental to various supervised learning algorithms, such as Linear Regression, Logistic Regression, Support Vector Machines, Neural Networks, etc. As we delve deeper into ML models, we will see supervised learning as the foundation for many powerful neural network models including CNNs, LSTMs, and Transformers, which are primarily used for this work.

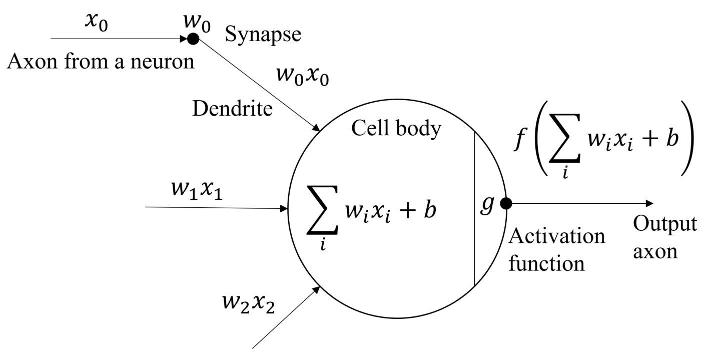


Figure 3.1: Mathematical model of neuron, based on [Li, 2023]

3.1.1.1 Artificial Neural Network

Neural Networks (NNs), also known as Artificial Neural Networks (ANNs), serve as the backbone of many modern ML models. Originally inspired by the structure and functioning of biological neural networks, ANNs are computing systems that learn to perform tasks by considering examples, generally without task-specific programming. They're designed to cluster and classify patterns.

The basic computational unit of an NN is the neuron, or node. Each node takes in inputs, applies a function (often nonlinear) to these inputs, and produces an output. In the mathematical model (see Figure 3.1) the signals (e.g. x_0, x_1, x_2), that are traveling along the axons are multiplied with weights (e.g. w_0, w_1, w_2) and bias b is added. These weights are adaptable and control the influence of every input of a neuron. All weighted input signals of the neuron are summed up in this basic model and if they pass a certain threshold, the neuron is activated and thus emits another signal—it fires. This sum can exchanged with another function, it is called the *activation function g*.

The term "network" arises due to the architectural representation as an interconnected web of numerous functions. These networks utilize a directed acyclic graph to visually depict the composition of the functions. For instance, let's consider three functions, f_1, f_2, and f_3, organized in a sequential chain to form:

$$f(x) = f_3(f_2(f_1(x)))$$

Here, f_1 is referred to as the first function or "layer" of the network, f_2 as the second, and so forth. The total number of functions, or layers, in the chain determines the network's depth, which is where the term "Deep Learning" originates. The final layer of the network is named the "output layer". The objective during the training phase of a neural network is to adapt $f(x)$ so it aligns closely with $f^*(x)$. The training dataset provides noisy, approximate instances of the function $f^*(x)$ at different data points.

Each data point or sample x is paired with a corresponding label $y \approx f^*(x)$. These training examples explicitly define the objective of the output layer for each input x, which is to produce an output that is nearly equivalent to y. However, the actions of the other layers are not directly defined by the training data. The learning algorithm is responsible for determining how these layers are used to produce the desired output, but the training data does not specify explicitly what each individual layer should do.

Instead, the learning algorithm is tasked with figuring out the optimal utilization of these layers to best approximate the function f^*. As the desired outputs for these layers are not explicitly stated in the training data, they are known as "hidden layers". [Goodfellow et al., 2016]

The layers of an NN often correspond to different levels of abstraction. For example, in a Deep Learning model trained on images, the initial layers might learn to represent simple shapes or edges, the middle layers might learn to represent more complex structures like parts of faces or objects, and the final layers might learn to represent whole objects or scenes. Mathematically, each layer in an NN is typically represented as a matrix of weights. If we denote the weight matrix for layer i as $W^{[i]}$, the bias vector as $b^{[i]}$, and the activation function as $g^{[i]}$, then the output $x^{[i]}$ of layer i can be represented as follows:

$$x^{[i]} = g^{[i]}(W^{[i]}a^{[i-1]} + b^{[i]}) \tag{3.6}$$

where $a^{[i-1]}$ is the output of the previous layer (or the input to the network when $i = 1$). The internal learned representations are not easily interpretable, making NNs somewhat of a *black box* approach: (Deep) NNs, consist of potentially hundreds of layers, each with many neurons. Each neuron applies a nonlinear transformation to its inputs. The interaction of these many nonlinear transformations leads to a highly complex overall mapping from inputs to outputs. This complexity makes the process difficult to interpret intuitively. In an NN, the input and output layers are visible to the user. Data is fed into the input layer, and what comes out of the output layer can be examined. What happens in between, in the hidden layers, is less accessible. Each hidden layer is transforming the data into a higher level of abstraction. While we can inspect the weights and biases of the hidden layers, understanding what these abstract representations actually mean is often nontrivial. Other ML models, like decision trees or linear regression, make their decisions based on clear, understandable rules or patterns in the input data. In contrast, NNs learn patterns in a different, more complex way, encoding the patterns in high-dimensional spaces that are difficult for humans to comprehend.

Nonetheless, NNs are some of the most powerful ML models. That strength comes from connecting neurons in these layers, and iteratively updating the weights of these connections based on the error in the network's predictions. This process is known as training the NN.

Network Training The process of training an NN works as follows [Goodfellow et al., 2016]:

1. *Initialization*: To begin with, the network's weights are initialized, often randomly. These weights, denoted $\mathbf{W}$, are the parameters that the network will learn during training.
2. *Forward pass*: For each input $\mathbf{x}$ in the dataset, a feedforward operation to get the network's output $\mathbf{y}$ is executed. The feedforward operation involves passing $\mathbf{x}$ through the network, layer by layer. At each layer i,

   $$z_i = \mathbf{W_i} \cdot \mathbf{x_i} + \mathbf{b_i} \tag{3.7}$$

   is calculated (where $\cdot$ denotes the dot product, $\mathbf{W_i}$ are the weights for layer i, $\mathbf{x_i}$ is the input to layer i, and $\mathbf{b_i}$ is the bias term for layer i), and then pass z_i through the layer's activation function to get

   $$\mathbf{x_{i+1}} = g(z_i) \tag{3.8}$$

   This continues until the output layer is reached.

3. *Loss computation*: After obtaining the output $\mathbf{o}$ for an input $\mathbf{x}$, we compare $\mathbf{o}$ with the true output (or label) $\mathbf{y}$ for that input (from the training data) to compute the loss. The loss function (e.g. MSE), denoted as $L(\mathbf{o}, \mathbf{y})$, quantifies the difference between the predicted and true outputs.
4. *Backpropagation*: The backpropagation algorithm [Linnainmaa, 1976] is used to compute the gradient of the loss function with respect to the weights. The gradient, denoted as $\nabla_{\mathbf{W}} L$, is a vector that points in the direction of steepest increase in the loss function, so for the descent we move in the opposite direction.
5. *Weight update*: We then update the weights using the gradient descent update rule:

$$\mathbf{W} \leftarrow \mathbf{W} - \alpha \nabla_{\mathbf{W}} L \tag{3.9}$$

   , where α is the learning rate, a hyperparameter that controls how big a step we take in the direction of steepest decrease.
6. Repeat steps 2-5: We repeat the feedforward, loss computation, backpropagation, and weight update steps for a certain number of iterations (or *epochs*, denoting a passthrough of the complete dataset) or until the loss falls below a certain threshold.

NNs have been used in a variety of applications, and their popularity has surged with the rise of Deep Learning: training NNs with many layers (hence they are *deep*) on large datasets. Despite their versatility, designing and training NNs can be a complex process. Selecting the appropriate architecture, activation function, learning rate, and other parameters requires a good understanding of the problem at hand. Additionally, the training process can be computationally intensive, depending on the dataset and model size.

In the subsequent sections, we will delve deeper into specific types of neural network architectures like CNNs, LSTMs, ConvLSTMs, Transformer Models, and Vision Transformers, that are some of the most powerful and widely-used types of NNs. They all face similar issues: they require significant (computational) resources to train, especially for larger inputs and they need a large amount of data to be trained on.

3.1.1.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) [LeCun et al., 1989] are a class of deep, feed-forward NNs, most commonly applied to analyzing visual data. They were designed to mimic the connectivity pattern of neurons in the human brain and were inspired by the organization of the visual cortex.

The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a transformed speech signal). This is achieved with the use of a convolution operation and local connections and receptive fields and tied weights followed by some form of pooling which results in translation invariant features. Within a CNN, each neuron in a convolutional layer is connected to only a small region of the input, which is referred to as the neuron's receptive field. Each neuron in a convolutional layer uses the same weights and bias. This means that the number of parameters in each convolutional layer is the same as the number of elements in the receptive field (plus one for the bias), regardless of the size of the input. Pooling means performing a down-sampling operation along the spatial dimensions (width, height), reducing the computational complexity and allowing for assumptions to be made about features contained in the sub-regions binned. The local receptive fields, the weight sharing and the pooling operation make CNNs very efficient because these

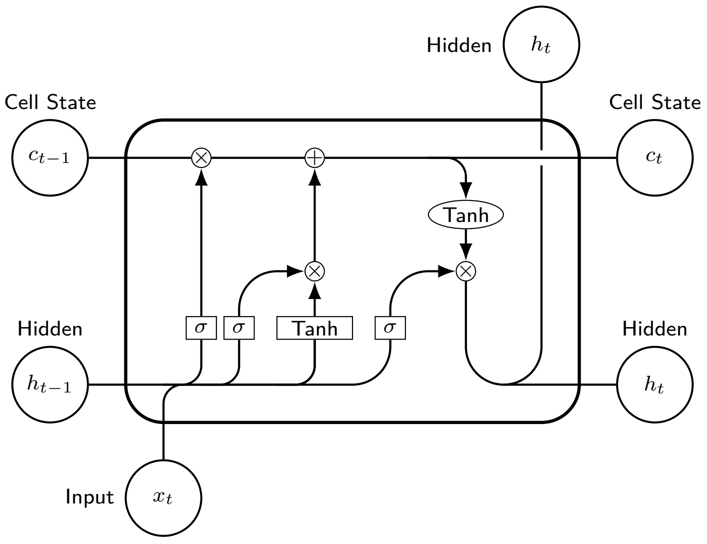


Figure 3.2: Long Short Term Memory

features all contribute heavily to CNNs being more parameter-efficient than fully-connected NNs. CNNs have been tremendous in the field of image recognition [Voulodimos et al., 2018] and were the first application of deep NN on large datasets, e.g. ImageNet [Deng et al., 2009] that lead to the recent general AI hype, initiated by the landslide win of a CNN at the ImageNet Challenge [Russakovsky et al., 2015]: AlexNet [Krizhevsky et al., 2012] in 2012.

3.1.1.3 Long Short Term Memory

Long Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997], is a type of Recurrent Neural Network (RNN) designed to learn from experience to classify, process, and predict time series given time lags of unknown size and duration between important events.

While conventional RNNs have "short-term memory" in that they use hidden states to process sequences of inputs, they are prone to forget information in the long run, especially as the length of the input sequence increases. This is due to an issue called the vanishing gradient problem [Hochreiter, 1998], where the contribution of information decays exponentially over time. LSTMs solve this problem by introducing a way to carry information across many time-steps without altering it, effectively providing the network with a form of "long-term memory". The key innovation of LSTMs is the cell state c_t, a horizontal line running through the top of the LSTM's diagrammatic representation shown in Figure 3.2. The cell state carries information from earlier time steps to later ones, and gating units ($\times$, $+$) regulate the flow of information into and out of the cell state, enabling the LSTM to remember or forget information. Furthermore, an LSTM features two different activation function, sigmoid (σ) and hyperbolic tangent (tanh). LSTMs have become a fundamental building block for many Deep Learning applications. They have been used to achieve state-of-the-art results in a variety of applications, including machine translation, speech recognition, and time-series prediction. Furthermore, they are the basis for more complex types of RNNs such as ConvLSTM Networks, which we will discuss next.

3.1.1.4 Convolutional Long Short Term Memory

Convolutional Long Short Term Memory (ConvLSTM) [Shi et al., 2015] is a type of RNN that combines the spatial feature extraction capabilities of a CNN with the sequence modeling capabilities of an LSTM. Traditional LSTMs operate on input data where the relationships between elements are defined by their relative positions, such as time-series data or text. However, for many types of data, including images and video, relationships between elements can be defined spatially as well. This is the specific niche of ConvLSTMs. In a ConvLSTM, the input-to-state and state-to-state transitions are both convolutional. This means that ConvLSTMs not only have the capability of handling the temporal properties of data like LSTMs but can also handle spatial data. The convolutional layers in a ConvLSTM allow it to capture spatial correlations in the input data, while the LSTM layers capture temporal correlations. ConvLSTMs have proven particularly effective in applications that involve spatio-temporal data, such as video processing [Grauman et al., 2022] and prediction, weather forecasting [Shi et al., 2015], and traffic prediction [Ali et al., 2022], and can have a great impact in the industrial domain, as well. They have shown a strong ability to handle the challenges posed by these types of data, where both spatial and temporal dynamics are crucial.

3.1.1.5 Transformer

Transformer models are a type of architecture introduced in a seminal paper, by Vaswani et al. [2017]. This paper proposed a new simple network architecture, the Transformer, based solely on attention mechanisms and completely dispensing with recurrence and convolutions and was originally tailored towards machine translation. From that starting point it was adapted to many other applications, far beyond NLP. Before going into detail on the Transformer, attention itself needs to be explained.

Attention To explain attention, some intuition around the origin of the wording is given first and then the technical background is explained. Attention is a mechanism that allows a model to focus on specific parts of the input data that are most relevant to the task at hand. It is inspired by the way humans pay attention to certain parts of a scene or text while ignoring others. Here follows a simple example to illustrate this concept: When a human reads a book, and comes across a sentence that mentions a character named "Alice". If they have read about Alice earlier in the book, they will automatically recall information about her to help them understand the current sentence. In this case, their attention is focused on the relevant parts of the text that provide information about Alice. In ML, attention works in a similar way. When processing input data, the model assigns different *attention scores* to different parts of the data. These scores indicate how much attention the model should pay to each part of the data. The model then uses these scores to weigh the input data and focus on the most relevant parts. More specifically, attention mechanisms are often used in sequence-to-sequence models, where the input is a sequence of data (e.g., a sentence), and the output is also a sequence of data (e.g., a translated sentence, cf. Figure 3.4). To explain that with an example, a quick introduction to how machine translation works with an encoder/decoder model is given:

- The encoder is the part of the model that is processing the input sequence and the decoder is the part that is generating the output sequence.
- The Query (Q) represents the current word or piece of information of focus. To clarify, it is like posing the question, "What should I pay attention to when considering this?"

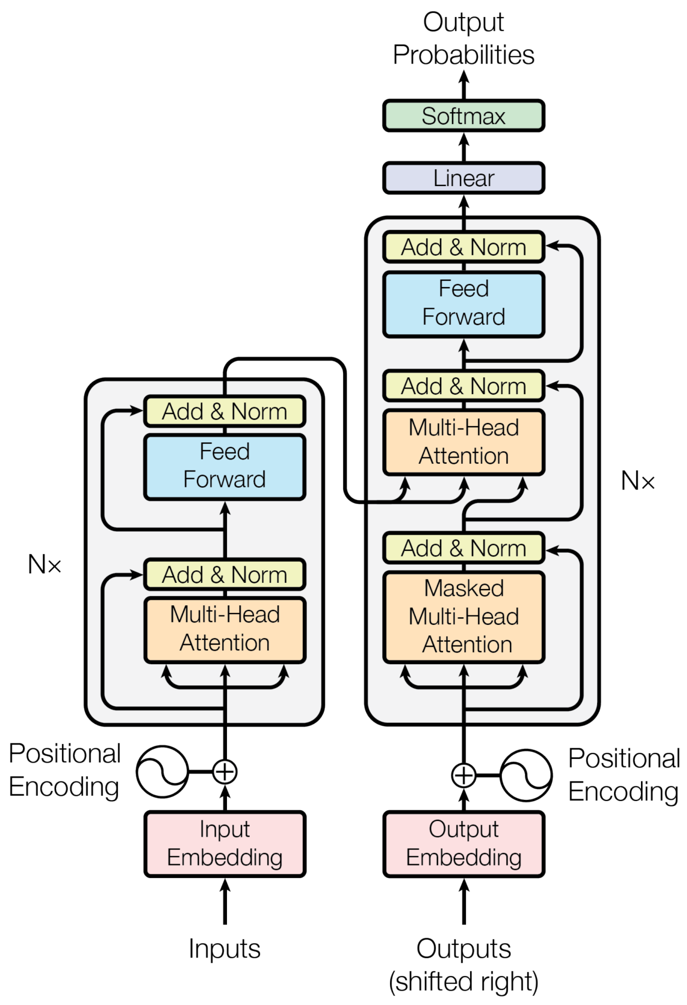


Figure 3.3: Transformer architecture, from [Vaswani et al., 2017]

- The Keys (K) are the elements the Query is compared against: they are "reference points" in the data.
- Each Key has an associated Value (V).

Once the model determines which keys are most relevant to the Query, the corresponding Values are used to produce the final output. When translating the sentence "Il fait beau" from French to English, which means "It is sunny" (cf. Figure 3.4), the encoder does the following: It reads the French sentence and creates a representation for each word. For simplicity, let's assume the following representations: Il: K_1, V_1, fait: K_2, V_2, and beau: K_3, V_3.

The decoder starts producing the English translation: it tries to predict the next word after 'It'. 'It' becomes the Query. To decide how to translate the rest of the sentence, the decoder compares the Query against all the Keys from the encoder to decide which French words ('Il', 'fait', 'beau') are most relevant. The comparison gives us attention weights. Assuming the highest weight is on 'fait', it then takes the Values associated with these relevant words to produce the output. In this case, the value K_2, V_2 associated with 'fait' would be used to produce the next word in English. Generally and technically, attention mechanism works by

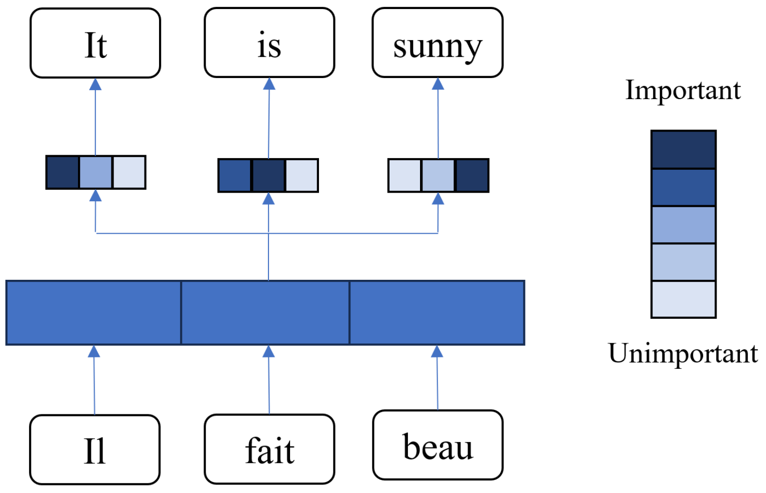


Figure 3.4: Attention in translation task: Weights are assigned to input words at each step of the translation

computing a weighted sum of the input sequence elements, where the weights are determined by the attention scores:

1. The *input sequence* is represented as a set of vectors, where each vector corresponds to an element of the sequence (e.g., a word in a sentence).
2. The attention scores are calculated by the model based on the *Query (Q), Key (K), and Value (V)* vectors. These vectors are obtained by multiplying the input vector with three different weight matrices (W_Q, W_K, and W_V).

   $$Q = X \cdot W_Q$$
   $$K = Y \cdot W_K$$
   $$V = Z \cdot W_V$$

   where X, Y, and Z are the different input vectors, and W_Q, W_K, and W_V are the weight matrices for the Query, Key, and Value vectors, respectively. For general attention, the weight matrices are multiplied with different vectors X, Y and Z, not just the input vector X as within self-attention.
3. The model calculates *scores* by taking the dot product of the Query vector of a particular element with the Key vectors of all other elements in the input sequence. The scores indicate how much attention each element should pay to every other element.

   $$S = Q \cdot K^T$$

   where S is the matrix of scores, Q is the matrix of Query vectors, and K^T is the transpose of the matrix of Key vectors.
4. *Softmax Normalization*: The scores are then passed through a softmax function to convert them into values that sum to 1. This ensures that the model assigns higher attention to more relevant elements.

   $$A = softmax(S)$$

where A is the matrix of *weights*, and S is the matrix of attention scores.

5. *Weighted Sum*: The model computes a weighted sum of the value vectors. This weighted sum represents the context vector for each element in the input sequence.

   $$C = A \cdot V$$

   where C is the matrix of context vectors, A is the matrix of weights, and V is the matrix of Value vectors.

6. *Output*: The context vectors are then used as input to the next layer of the model or for making predictions.

The attention mechanism allows the model to focus on the most relevant parts of the input sequence when generating the output sequence. It is a key component of many state-of-the-art ML models, particularly in the field of natural language processing.

Self-attention The Transformer model's core idea is the *self-attention* mechanism, which weighs the importance of different input elements in order to best derive meaningful context and is a special type of attention.

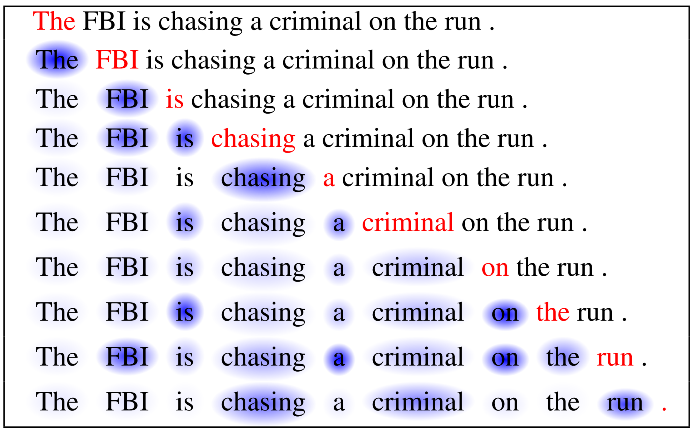
In a general attention mechanism, the Query, Key, and Value vectors come from different sources (see above). In a model for machine translation, the Query vectors might come from the decoder while the Key and Value vectors come from the encoder, as described above. This allows the decoder to "pay attention" to different parts of the input sequence when generating each word of the output sequence. In self-attention, the Query, Key, and Value vectors all come from the same source, which is the input sequence itself. This allows the model to "pay attention" to different parts of the input sequence when processing each element of the sequence (cf. Figure 3.5). Self-attention is used to capture relationships and dependencies between elements within the same sequence. For instance, consider the sentence: "The FBI is chasing a criminal on the run."

If the focus lays on the word "chasing" (the Query), self-attention might help the model relate 'chasing' to 'FBI' by weighing the Key of 'FBI' higher and fetching its Value. This gives the model contextual understanding that it's the 'FBI' that is 'chasing'.

To summarize, the difference between attention and self-attention is the source of the Query, Key, and Value vectors. In attention, the vectors come from different sources, while in self-attention, the vectors all come from the same source. One of the significant advantages of the self-attention mechanism is that it does not require any predefined temporal or spatial relationships between elements in the sequence. This makes it very flexible and applicable to a wide range of tasks, including tasks with complex interdependencies between elements.

Transformer models often consist of an encoder and a decoder. The encoder processes the input data and the decoder generates the output. Both the encoder and decoder are composed of multiple identical layers, each containing two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network (cf. Figure 3.3). The Multi-head Attention mechanism operates by executing multiple attention processes concurrently. These parallel attention outputs are then merged together and undergo a linear transformation to achieve the desired dimensionality. Essentially, having multiple attention heads enables the model to focus on various aspects of the sequence in distinct ways, such as emphasizing either long-term or short-term dependencies. [Vaswani et al., 2017]

One of the most remarkable aspects of Transformer models is their ability to handle long-term dependencies in the data. They can effectively relate information from distant parts of



The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .

Figure 3.5: Example for self-attention: Red indicates the current focused word and the intensity of blue represents the degree of attention, from Cheng et al. [2016]

the sequence when making predictions, a feature that makes them particularly useful for NLP tasks. Nonetheless, they are also position invariant. Transformers take the input as a set, rather than a sequence. That is why so-called *positional encoding* needs to be introduced to have that sequential information available for e.g. NLP, but also for CV tasks as in the Vision Transformer (ViT), presented in Subsection 3.1.1.6.

On parallelism of Transformers In RNNs, computations within a layer are dependent on previous computations within that layer, creating a sort of sequential dependency. This consecutive nature of computation limits parallelism, as subsequent calculations must wait for the preceding ones to complete before they can be executed. Transformers solely use the self-attention mechanism and feed-forward networks, both of which are highly parallelizable. Each output element in self-attention is calculated from a weighted sum of all input elements. These calculations are independent and can be done simultaneously. The same applies to feed-forward networks where each node's computation in a layer is independent of the other nodes in the same layer. RNNs, including LSTMs, are executed sequentially and the computations within layers depend on computations from before. That is why their computation can be hardly parallelized. Transformers on the other hand are built to be executed in parallel. In the self-attention mechanism, the attention scores for each Query-Key pair are computed independently of one another. As a result, all of these scores across all positions can be computed in parallel. Furthermore, these scores are used to create a weighted sum of Value vectors, which can also be calculated simultaneously for all positions. This ability to handle multiple positions at once contributes to the high degree of parallelism in Transformers. Transformers typically use layer normalization, a process that standardizes the inputs across the mini-batch for each layer. Because layer normalization operates independently on each feature, it can be computed in parallel, further contributing to the parallelizability of Transformers. Transformers were introduced at a time when the computational capabilities, especially of Graphics Processing Units (GPUs), that are inherently good at parallel computation, were soaring.

Additionally, very large corpora of data are openly available: e.g., for Generative Pre-trained Transformer (GPT) 3, that uses only the decoder part of the transformer architecture, the datasets in use were published: Common Crawl, Webtext2, Books1, Books2 and

Wikipedia [Brown et al., 2020], summing up to 499 billion tokens total. This massive increase in parallel compute, this architecture tailored towards parallelism, and very large datasets made the huge breakthroughs and massive pre-trained networks, such as the GPT [Brown et al., 2020; OpenAI, 2023] architectures, possible. Another famous implementation of the encoder part of the Transformer model is the BERT (Bidirectional Encoder Representations from Transformers) model [Devlin et al., 2018], which set new records for performance in a variety of NLP tasks. However, Transformer models have since been applied to a wide range of tasks beyond language processing, including image and video processing, and have inspired a whole new class of models called ViTs, which we will discuss next.

3.1.1.6 Vision Transformer

The ViT model [Dosovitskiy et al., 2020] is a recent development in the field of CV, which applies the Transformer model to image classification tasks. The key idea behind ViTs is to treat an image as a sequence of patches and to apply Transformer models to this sequence, just as they would be applied to a sequence of words in a text.

Firstly, an input image is split into small patches, each of which is treated as an equivalent to a word or token in a text input for a language model. Each patch is then linearly transformed (flattened and projected into a higher-dimensional space) to create a sequence of embedded vectors. A learnable position embedding is added to each patch embedding to retain positional information that was lost during the patching process. This sequence, which is different to the set, that is employed normally, of embedded vectors is then input into a standard Transformer encoder. The Transformer treats each patch as an individual token and processes them all concurrently in a multi-headed self-attention operation. This allows the model to consider each patch in the context of all other patches, enabling the efficient capture of long-range dependencies between patches located far apart. The Vision Transformer has shown impressive results, matching or even exceeding the performance of CNNs on large-scale image classification tasks, like ImageNet. One major advantage of ViT over traditional CNNs is its scalability - ViTs tend to perform better with increased data and computational resources, which is not always the case with CNNs. On the other hand, CNNs usually perform better on smaller datasets than ViTs.

3.1.2 Transfer Learning from Simulation to Reality

Transfer learning is a popular strategy in ML and Deep Learning where a model pre-trained on one task is reused as a starting point for a different but related task. This approach leverages the fact that many features (like edge detection in image processing or sentence structure in NLP) are general across multiple tasks [Tan et al., 2018]. By using pre-trained models, it is possible to leverage the patterns and knowledge already learned from vast amounts of data, thereby reducing the computational resources and time needed to train a high-performing model from scratch. It also offers a solution when the target task has limited data available, which would usually make training a Deep Learning model challenging. *Fine-tuning* is used as a synonym to transfer learning. Once a pre-trained model has been deployed on a new task, the model's parameters are often further optimized or "fine-tuned" on the new task's data. Fine-tuning involves continuing the backpropagation process to adjust the weights of the original pre-trained network, allowing the model to adapt to the new task. There are different strategies for fine-tuning. Some may choose to fine-tune all layers in the network, while others may choose to keep some of the earlier layers fixed (due to their more generic feature extractors) and only fine-tune some higher-level portion of the network that is more likely to be specific to the task at hand. Transfer learning or fine-tuning has been the key to the success of many ML models, especially in fields like CV and NLP, i.e. are even name-giving for such influential models such as GPT - generative *pre-trained* transformer. One of the major challenges in ML, and especially in RL and robotics [Zhu et al., 2023], but also for industrial use cases such as LCM for the manufacturing of CFRP [Stieber, 2018; Stieber et al., 2023b], is the so-called "simulation-to-reality" gap. This gap refers to the difficulty of transferring knowledge learned in a simulated environment to a real-world environment.

Simulated environments are often used for model training due to their scalability, safety, and cost-effectiveness. Simulations allow models to experience a wide range of situations and learn how to handle them without the risk of damaging real equipment or causing harm in the real world. However, simulations cannot perfectly replicate the complexity and unpredictability of the real world. Transfer learning from simulation to reality aims to bridge this gap. It leverages the model's ability to learn generalizable features in the simulated environment and then fine-tunes the model in the real-world environment. The goal is to leverage the efficiency and safety of simulations while still obtaining a model that performs well in the real world. Several strategies can be used to tackle the "simulation-to-reality" problem. One common approach is *domain randomization* [Tobin et al., 2017], where the simulation is run many times with different random variations each time, such as different lighting conditions, physical properties, or object positions. This helps the model learn to generalize its behavior across a wide range of conditions, increasing the likelihood that it will perform well in the real world. Another approach is aiming to minimize the discrepancy between the simulation and the real world. This can be achieved by refining the simulation based on real-world data, or by using techniques like adversarial training [Ganin et al., 2016], where the model is trained to be robust against the kind of changes it might experience when moving from the simulated to the real environment.

The applications, potentials and limitations of sim-to-real transfer learning for LCM are presented in the following chapters regarding the detection of dry spots (Section 5.1), process forecasting (Section 5.2) and the digital twin of product properties (Subsection 6.2.2).

3.1.3 Reinforcement Learning

Reinforcement Learning (RL) is a paradigm of ML that involves an agent that learns its behavior by interacting with an environment and receiving feedback in the form of rewards or penalties [Sutton and Barto, 1998]. The development of RL methods used today has been influenced by various theoretical backgrounds, with the theory of optimal control in stochastic processes being the most important [Sutton and Barto, 1998]. In the following section, the formal model within which RL algorithms operate and important concepts of these algorithms are explained. Two state of the art algorithms, A2C and PPO, are presented. Both use Deep Learning methods and will be used in the practical application of RL on the RTM process in Chapter 7.

3.1.3.1 Introduction to RL

The following section explains the basics that are important for understanding the concepts and algorithms of RL. A central feature of RL is the interaction of the agent with its environment,

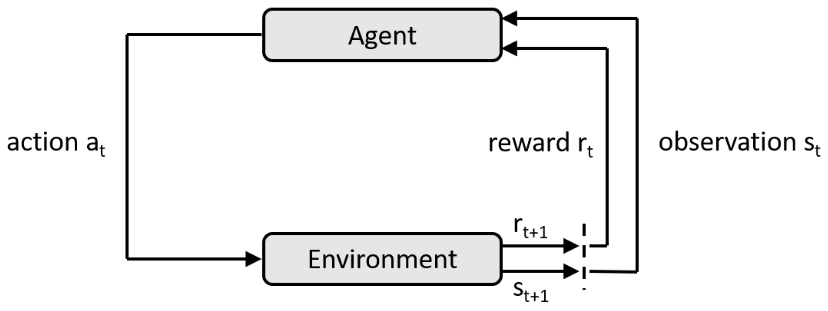


Figure 3.6: The RL agent's interaction with the environment, following Sutton and Barto [1998]. The dashed line indicates the next time step. Here, the agent puts the environment in state s_{t+1} and receives reward r_{t+1} after selecting action a_{t+1}.

which follows a defined pattern that is based on the formalism of the Markov Decision Process (MDP) [Bellman, 1957].

Agent and Environment In RL, a problem to be solved is divided into two systems. The *agent* makes decisions to maximize pre-defined objectives, and the *environment* responds to these decisions [Silver et al., 2021]. For instance, the environment may encompass the real world, a simulation, or a computer game. The agent interacts with the environment in discrete time steps. In each time step t, the agent receives a *state* s_t of the environment. Often it is also named *observation*, since only a part of the whole world state is passed to the agent. Typical types of states are images or vectors that represent a collection of scalar information.

Based on the state s_t, the agent selects an action a_t. As a result, the state of the environment changes probabilistically, conditioned on the current state and the chosen action. The agent receives the next state s_{t+1} and the *reward* r_{t+1}. Repeated interactions following this pattern, as shown in Figure 3.6, yield a *trajectory* of the form:

$$s_1, a_1, r_2, s_2, a_2, r_3, s_3, ...$$

Depending on the environment, trajectories can be finite or infinite. Finite trajectories are called *episodes* and end in a terminal state s_T:

$$s_1, a_1, r_2, s_2, a_2, r_3, s_3, ..., r_{T-1}, s_{T-1}, a_{T-1}, r_T, s_T$$

To simplify matters, we only consider episodic environments in the following, which assumes that every trajectory will ultimately lead to a terminal state.

The concept of RL operates on the premise that a scalar reward function can be used to define each objective that the agent aims to accomplish. The reward function assigns a reward r_t to every state s_t, and achieving the objective is equivalent to maximizing the total rewards obtained during an episode [Silver et al., 2021]. Accordingly, the agent's task is to select actions that lead to the best long-term outcome, even though they may yield suboptimal short-term results [Sutton and Barto, 1998].

Markov Decision Process and Goals of RL The RL algorithms were developed based on the formal environment model of the MDP. The Markov property [Mostowski, 1953] states that the probability of transitioning from the current state to a certain next one – as well as the probability distribution of attainable next rewards – only depends on the current state and the action taken, but not on any prior state.

An MDP consists of:

- a set of states $\mathcal{S}$
- a set of actions $\mathcal{A}$
- a set of rewards $\mathcal{R} \subset \mathbb{R}$
- a probability distribution $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$

The state set $\mathcal{S}$ can be finite or infinite. Early RL algorithms rely on explicitly listing all states [Sutton and Barto, 1998]. Therefore, they require finite state spaces; in particular, they must not be too large to solve the problem within realistic computational constraints. However, Deep Reinforcement Learning (DRL) algorithms usually use function approximators implemented by ANN, so this restriction does not apply. Also, the set of actions $\mathcal{A}$ may be finite or infinite. However, in many applications, including the work carried out in Chapter 7, the action space can be discretized to obtain a finite set of actions. Furthermore, the assumption is made that the same set of actions is available in each state. The probability distribution p represents the behavior of the environment. A fundamental assumption made to apply the MDP is that the dynamics of the environment can be modeled by a stochastic process. Thus, p maps four arguments, including the current state s_t, a reward r_{t+1}, a subsequent state s_{t+1}, and an action a_t, to the probability that by performing a_t in s_t, the next state is s_{t+1}, and this state transition yields r_{t+1}. Many environments in which RL will eventually be applied, do behave relatively deterministically, for example, when their behavior is determined by physical laws. Nevertheless, modeling by a stochastic process is justified because the dynamics of the environment may be either unknown, subject to minor fluctuations and disturbances, or too complex to build deterministic mathematical models [Brunton and Kutz, 2019].

A *policy* is defined as a function $\pi(s)$ that gives a probability distribution over the actions available in state s. In simple applications, these probabilities can be stored explicitly for all pairings of states and actions. This is referred to as the tabular method. However, for large state spaces, the storage requirements would be too large, so π is approximated by a parameterized function π_θ such as, e.g. a deep NN or a decision tree. The parameter vector θ may be of much smaller dimension than the state space [Brunton and Kutz, 2019].

The goal of accumulating as many rewards as possible over the course of an episode – called the *return* – can be formalized by a cumulative measure. Often the discounted sum is used with the discount factor $\gamma \in [0, 1)$ [Sutton and Barto, 1998]:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{3.10}$$

T describes the length of the episode. The discount factor weights short-term rewards higher than those that are far in the future. An important property of the return is that it can be formulated recursively:

$$G_t = r_{t+1} + \gamma G_{t+1} \tag{3.11}$$

This leads to a concept that is central to the theory of many RL algorithms. The expected value $\mathbb{E}_\pi[G_t \mid S_t = s]$ describes the cumulative rewards that can be expected when, starting at state s, future actions are selected by the policy π. Assuming that the environment can be described by an MDP and thus the state transitions and rewards are determined by the probability distribution p, this quantity can be recursively formulated like (3.11), using the law of iterated expectations:

$$\mathbb{E}_\pi[G_t] = \sum_{a \in \mathcal{A}} \pi(a \mid s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) \left[r + \gamma \mathbb{E}_\pi[G_t \mid S_t = s'] \right] \tag{3.12}$$

Here $\pi(a \mid s)$ denotes the probability of drawing a from $\pi(s)$. The probability of choosing action a from state s, thereby ending up in state s' and receiving reward r in the process, is multiplied by r and the discounted expected value starting from the new state. This is done for all combinations of actions, states, and rewards.

In practice, the calculation of this expected value is typically inexact, as the transition probabilities of the environment are generally unknown. Therefore, it is often approximated by recording episodes through interaction with the environment. Subsequently, by examining the observed state transitions and the acquired rewards, the anticipated reward from a previous state can be estimated [Schulman, 2016]. This estimate is denoted by $\hat{\mathbb{E}}_\pi[G_t \mid s]$. By G_t, a condition for an optimal policy can be formulated. A policy is optimal if it maximizes the expected value of cumulative rewards:

$$\pi^* = \arg\max_\pi \mathbb{E}_\pi[G_t] \tag{3.13}$$

This policy is denoted π^* and finding it is the goal of RL algorithms. In many real-world applications of RL, the environment cannot be shown to satisfy the pattern of the MDP, and in almost all meaningful applications, at least the transition probabilities between states are unknown [Brunton and Kutz, 2019]. Additionally, action delays degrade the performance of RL [Travnik et al., 2018]. Frequently used models of interaction between an agent and its environment, e.g. MDP, do not capture the fact that in an asynchronous environment, the state of the environment may change during computation. Nevertheless, the MDP is an important approach for motivating and deriving modern RL algorithms.

3.1.3.2 Approaches in RL

Solution approaches based on knowledge of the full MDP, especially including the transition probabilities p of the environment, are classified as *model-based RL*. Here, the environment model can be learned by ML methods or can be pre-implemented. It is characteristic of model-based methods that action selection occurs through planning in the environment model [Brunton and Kutz, 2019]. By contrast, methods of *model-free RL* directly learn functions that serve to select an action. This does not require a stochastic model of the environment, making the methods more general and easier to apply. This also implies that model-free methods can operate in environments that are unknown or partially known. In the next sections, various model-free approaches are explained. Asynchronous Advantage Actor-Critic (A2C) and Proximal Policy Optimization (PPO) combine a variety of concepts that have been developed over years. The following remarks cover the most important theoretical background and are extended at some points by outlooks on concrete algorithms that help the understanding.

3.1.3.3 Value-based Methods

The idea behind value-based methods is to approximate the long term value of states and actions for better decision making. The state-value function expresses the long-term value of being in state s, with actions chosen by a policy π. It is defined as the expected future rewards, see (3.12). This definition can also be expressed recursively:

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \sum_{a\in\mathcal{A}} \pi(a \mid s) \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s', r \mid s, a)\big[r + \gamma V_\pi(s')\big] \tag{3.14}$$

Similarly, the action-value function expresses the value of taking the action a in the state s:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \sum_{s'\in\mathcal{S}} \sum_{r\in\mathcal{R}} p(s', r \mid s, a)\big[r + \gamma V_\pi(s')\big] \tag{3.15}$$

When these functions are approximated, they are denoted by $\hat{V}(s)$ and $\hat{Q}(s, a)$ in the following, the policy π is omitted for brevity. By these formulations, finding an optimal policy is closely related to finding optimal state- and action-value functions. Thus, an optimal policy π^* maximizes value functions, and optimal value functions are those that follow an optimal policy. From this connection follows an important property on which value-based methods are built. When optimal value functions are known, the search for long-term optimal actions (i.e., by taking the expectation, hence, averaging over all possible future outcomes, given a certain policy) is reduced to the search for immediate optimal actions since all possible future consequences are already taken into account by the value functions [Sutton and Barto, 1998]. This property is called the *optimality principle* according to Richard Bellman [Bellman, 2003], who also developed an idea for solving such problems, namely, *Dynamic Programming* [Bellman, 2003]. If the transition probabilities of the environment were known, Dynamic Programming techniques such as value iteration or policy iteration could be applied to compute approximations $\hat{V}$ and $\hat{Q}$ [Brunton and Kutz, 2019].

To approximate the value functions despite an unknown environment (which is possible with model-free RL), *Temporal Difference (TD)* methods use actual experience. The iteration rule is

$$\hat{V}^{new}(s_t) = \hat{V}^{old}(s_t) + \alpha\big(r_{t+1} + \gamma\hat{V}^{old}(s_{t+1}) - \hat{V}^{old}(s_t)\big) \tag{3.16}$$

where $\hat{V}^{new}$ are the updated estimates and $\hat{V}^{old}$ are the old estimates, each stored in a table. The obtained reward r_{t+1} and the following state s_{t+1} are used in the recursive formulation of $\hat{V}^{old}(s_t)$. This still uses estimation to a degree, but by taking into account actual observed values, it is better than pure estimation. If $\hat{V}^{old}$ were already optimal, the estimate of the value of s_t would be equal to the recursive variant and the TD error $\big(r_{t+1} + \gamma\hat{V}^{old}(s_{t+1}) - \hat{V}^{old}(s_t)\big)$ would be zero. The new estimate for s_t, weighted by the learning rate α, is adjusted in the direction of the TD error. Through multiple iterations, an algorithm following this pattern converges to a correct solution [Brunton and Kutz, 2019]. Analogously, the action-value function could also be learned. Instead of just one step, multiple observed steps can be included in the update. This leads to the formulation of the *TD*(n) objective, where n steps are considered starting at state s_t:

$$J_{TD(n)} = \sum_{j=0}^{n} \gamma^j r_{t+j} + \gamma^{n+1}\hat{V}^{old}(s_{t+n+1}) \tag{3.17}$$

From this, the *TD*(n) error can now be formed by subtracting the original estimate $\hat{V}^{old}(s_t)$. The error can then be used to formulate an iteration rule as in (3.16). It should be noted here that value-based RL algorithms were originally designed primarily for environments with discrete and finite sets of actions and states. Tabular methods are often used, i.e., the values of all states, actions, or their combinations are explicitly stored and the individual values are updated by iteration rules, as in the $TD(0)$ algorithm. However, this is inefficient if not impossible in many use cases, since the sets are either infinite or too large to compute and store all values individually. In such cases, function approximators, i.e. a ANN from Subsection 3.1.1.1, can be used to realize $\hat{V}$ and $\hat{Q}$. For example, let $\hat{Q}_\omega(s,a)$ be a function parameterized by the parameter vector θ. The gradient method can be used to optimize the parameters θ, where the objective is formulated as minimizing the TD error. When the $TD(0)$ error is used, the parameters ω per training step $s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}$ are increased by

$$\delta\omega = \alpha(r_{t+1} + \gamma\hat{Q}_\omega(s_{t+1}, a_{t+1}) - \hat{Q}_\omega(s_t, a_t))\nabla_\omega\hat{Q}_\omega(s_t, a_t) \tag{3.18}$$

adjusted, where α denotes the learning rate [Sutton and Barto, 1998].

3.1.3.4 Policy-based Methods

The goal of policy-oriented methods is to learn a policy function directly. To do this, parameterized functions are used to approximate the functions being sought. The policy is denoted as π_θ, where θ represents the parameters. These are iteratively adjusted according to the gradient method pattern, where α represents the learning rate:

$$\theta_{k+1} = \theta_k + \alpha g \tag{3.19}$$

Here, g is the gradient of expected future rewards, namely $g := \nabla_\theta\mathbb{E}_{\pi_\theta}[G_t]$, and represents the goal of learning an optimal policy. Since the expected value of rewards depends on the policy according to which decisions are made, which in turn depends on its parameters θ, g is suitable for improving the policy π_θ according to the iteration rule (3.19). However, G_t contains the probability distribution of the underlying MDP (3.11), so this expression is inconvenient to differentiate. By applying the *policy gradient* theorem [Sutton and Barto, 1998], this gradient can also be expressed by:

$$g = \mathbb{E}_{\pi_\theta}\left[\sum_{t\in T}\psi_t\nabla_\theta log\pi_\theta(a_t \mid s_t)\right] \tag{3.20}$$

Various functions can be used for ψ that evaluate the choice of action a_t in state s_t. In the simplest case, expected future rewards are used [Schulman, 2016]. This expression has the advantage that only the parameterized policy itself needs to be differentiated. In training, the expected value is approximated by gathering experience, in the sense of recording episodes. The state transitions observed in this process, along with rewards, are used to compute the approximation $\hat{g} \approx \nabla_\theta\hat{\mathbb{E}}_{\pi_\theta}[G_t]$ and gradually improve π_θ.

Actor-Critic When a learned function, such as the approximation of the action-value function $\hat{Q}(s,a)$ from Subsection 3.1.3.3, is used instead of ψ, the algorithm is called an *actor-critic* method [Konda and Tsitsiklis, 1999]. Here, the policy function, the actor, and the evaluation function, the critic, are learned simultaneously. For example, the critic is trained by a temporal difference method, as in (3.18), and evaluates the decisions that the actor makes. The actor is trained according to (3.19), using the critic's evaluation to estimate the gradient [Grondman et al., 2012].

Actor-critic methods are one of the most important groups of RL algorithms today, as they combine the advantages of value-based and policy-based methods and can compensate for some disadvantages [Grondman et al., 2012]. While value-based methods often must resort to performing an elaborate optimization over the action space, a learned policy can be evaluated efficiently. However, pure policy-based methods often suffer from the fact that the cumulative rewards observed over the course of individual episodes are subject to high variance. This makes the estimation of the gradient inaccurate, leading to slow convergence. Here, value functions can be used to train the policy function. The variance of the value approximation decreases with ongoing training [Grondman et al., 2012].

3.1.3.5 RL Algorithms

In Chapter 7, RL algorithms are trained to control the resin injection process of the RTM process. The Asynchronous Advantage Actor-Critic algorithm [Mnih et al., 2016] and the Proximal Policy Optimization algorithm [Schulman et al., 2017] are used. Both are deep RL algorithms, so deep NN are used for function approximation. Since both algorithms follow the actor-critic pattern (cf. Subsection 3.1.3.4), the policy and an evaluation function must be approximated.

An important step in the development of deep RL algorithms was the extension of the RL paradigm to include parallelism [Nair et al., 2015]. This involves the use of multiple, parallel instances of the environment, which may be subject to different randomized initial conditions, depending on the nature of the environment. As a result, if appropriate computational resources are available, significantly more interactions can be performed in less time. Interaction with the environment occurs synchronously with all instances in A2C and PPO. Subsection 7.2.2 describes in more detail how parallelization was crucial for implementing DRL for the proposed RTM process.

An additional challenge to consider when designing algorithms is balancing *exploration* of possible strategies and *exploitation* of known strategies [Arulkumaran et al., 2017]. When an agent consistently selects actions deemed optimal based on its current evaluation, it is only capitalizing on its existing knowledge. As a result, the policy may reach a local optimum, but may not find better strategies even if they exist. Therefore, a mechanism must be introduced so that the agent also chooses suboptimal actions, especially in the early stages of training. Thus, the action and state space are explored to find the best possible solution. This can be achieved, i.e. by randomly varying the actions chosen by the agent, with the probability of doing so decreasing as time progresses [Arulkumaran et al., 2017]. Modern deep RL algorithms, on the other hand, often use stochastic strategies. Here, the outputs of the ANN representing the policy are interpreted a probability distribution. Thus, random variation of actions is automatically incorporated into the policy, resulting in an exploration of the action space [Schulman, 2016].

In the following sections, the two algorithms in use are presented. Since both algorithms conform to the actor-critic pattern, the idea behind the designed critics is presented. This defines the optimization objective according to which the policy is adapted, and is thus crucial to understand how they work. A detailed presentation of the algorithms as well as a discussion of all additional mechanisms used is omitted, since the focus of this work is on the application.

A2C The algorithm A2C is a synchronous modification of the *Asynchronous Advantage Actor-Critic* algorithm (A3C) [Mnih et al., 2016] introduced first. A3C uses multiple actors that interact independently and asynchronously with instances of the environment and are synchronized periodically. Subsequently, a synchronous variant was developed, which, while incurring higher idling times due to synchronization after each time step, in return simplified the implementation [Arulkumaran et al., 2017].

The use of the namesake advantage function as a critic defines the algorithm. It is defined as the difference between the action-value function (3.15), and the state-value function (3.14), and expresses whether an advantage or disadvantage was obtained by selecting an action. This information is utilized to modify the actor's policy, either increasing or decreasing the likelihood of selecting that action, depending on the result of the advantage function.

To avoid having to approximate both value functions, the advantage function can be replaced by the TD error, see Subsection 3.1.3.3. If the value functions were optimal, this reformulation would be exact; in Algorithm A2C, it can at least be used as an approximation [Mnih et al., 2016]. Otherwise, the algorithm follows the actor-critic pattern.

PPO PPO is an actor-critic algorithm and uses the same advantage function as A2C to evaluate actions and states, as described in Subsection 3.1.3.5. However, instead of forming the gradient according to the policy gradient theorem as in (3.20), a surrogate optimization objective is formulated. This incorporates and bounds the deviation of the new policy from the old one. The rationale behind this approach is that excessively large alterations in the policy may result in diminished performance during the training process. When the policy parameters fall within an unfavorable range, leading to poor performance, it may require a considerable amount of time or even fail to reach more optimal regions within the parameter space [Schulman et al., 2015].

The predecessor of PPO, the Trust Region Policy Optimization (TRPO) algorithm [Schulman et al., 2015] employed an elaborate second-order procedure to limit the step size during optimization. To make the algorithm more efficient, easier to implement, and more generally applicable, PPO sets lower and upper bounds to ensure continuous and stable improvement of the policy. Empirically, PPO has been shown to outperform even TRPO [Schulman et al., 2017].

Neural Network Architecture The NNs used to approximate the required functions, i.e., the policy and the evaluation function, can be designed independently of the choice of algorithm. Therefore, the same network architecture can be used for both A2C and PPO.

For actor-critic methods, the first layers of the two networks (processing the state observation as input features) can be shared between the actor and the critic, reducing the number of parameters [Mnih et al., 2016]. Moreover, the actor and the critic thus use the same abstracted and latent representation of the state of the environment to make their decisions. Details about the specific NN architecture in use for the RL models for the control of the RTM process are described in Subsection 7.2.4.

3.2 Simulation of LCM

Several steps of the LCM toolchain can be simulated separately, as shown in Figure 3.7:

1. Draping of the textile
2. Simulating the fluid flow
3. Curing of the matrix and computation of residual stresses and distortions

The focus of this thesis, lays on (2), the fluid flow, because it is one of the steps that can be influenced actively within the process itself by changing parameters. In general, the simulation

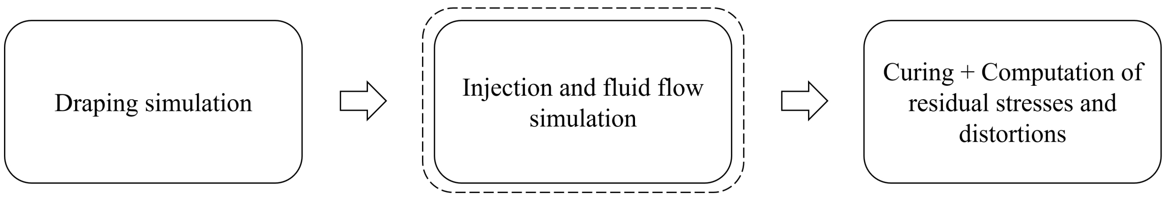

Draping simulation
Injection and fluid flow simulation
Curing + Computation of residual stresses and distortions

Figure 3.7: The LCM simulation toolchain [ESI Group, 2023]. Dashed box indicates main focus of this work: the injection and fluid flow process.

can be coupled and thus depend on the results of the previous step of the toolchain. As we focus solely on the fluid flow, permeability and porosity values are taken as is from material specifications or actual measurements and are not coming from a previous simulation step. Local changes in these properties often come from draping the textile in a complex mold. Within the presented case studies A and B, the geometries in use are flat (2D) and show no advanced curvatures, bends or edges etc., which justifies the simplification of using the property values as they are. The simulations are based on Darcy's Law, introduced in Subsection 2.2.5. The equation is reiterated here for clarity:

$$\mathbf{v} = -\frac{K}{\eta}\nabla P \tag{3.21}$$

, where ∇P denotes the pressure gradient, K describes the permeability of the textile and η is the dynamic viscosity of the resin, and the equation of continuity for incompressible flow:

$$\nabla \cdot \mathbf{v} = 0 \tag{3.22}$$

Here, $\nabla\cdot$ represents the divergence operator, $\mathbf{v}$ represents the velocity vector of the fluid. But according to Darcy's law, $\mathbf{v} = -\frac{K}{\eta}P$. Therefore, we can substitute for $\mathbf{v}$ in the continuity equation to obtain a form of the pressure equation:

$$\nabla \cdot (-\frac{K}{\eta} \cdot \nabla p) = 0 \tag{3.23}$$

This equation states that the divergence of the flux is zero, which is another way of stating the principle of conservation of mass for this particular system. In general, the mathematical problem of fluid flow can be solved numerically with different methods: Finite Difference Method (FDM), Finite Element Method (FEM), Finite Volume Method (FVM) in a time and space discrete manner. In software packages for general Computational Fluid Dynamics (CFD) mainly FVM are implemented.

For the simulation of the injection, the calculation takes place in the saturated part of the mold which is filled with the porous preform. Discrete and iterative propagation of the flow front is admissible for small time steps. This is an initial-boundary-value problem and initial and boundary conditions must be specified. For example, zero pressure outside the saturated

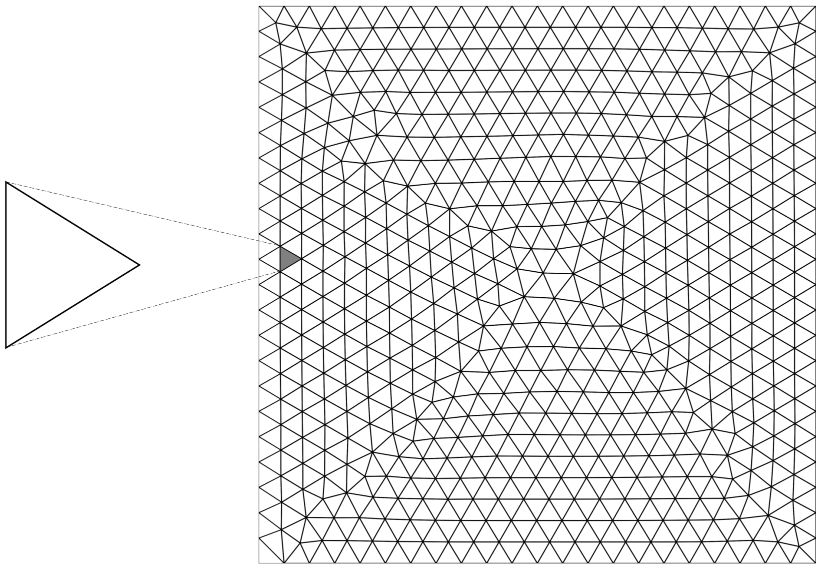


Figure 3.8: Exemplary mesh (from case study B), with single triangular mesh element.

part, slip wall boundaries, and prescribed pressure at the injection gates. At the beginning of the injection only the injection ports are saturated.

The pressure distribution within the saturated regions is described by (3.23). If the pressure distribution is known, Darcy's law (3.21), can be used to calculate the volume flow or the flow velocity can be calculated for each discrete element of the *mesh* (cf. Figure 3.8). Thus, in each time increment the new, larger saturated region is calculated. The new saturated region, serves in the next time increment as a starting point for the calculation of the pressure distribution. This procedure is repeated until a termination criterion is reached, which can be a certain time instance, step number, filling percentage or else. [Trochu et al., 1993]

There are several software packages available for simulating the specific LCM or RTM injection, including software tailored to the specific use case such as PAM-RTM [ESI Group, 2023], RTMSim [Obertscheider and Fauster, 2023], or LIMS [Simacek et al., 2004]. Other than that there are general purpose tools, such as OpenFoam [Weller et al., 1998] or Ansys [Ansys Inc., 2023]. The first two software packages are used to implement the experiments for this work: PAM-RTM, which uses the FEM and a variant of RTMSim, adapted for RL, that uses Finite Area Method (FAM), the 2D-variant of the FVM.

For fluid simulations in general, several specificities have to be considered, which are addressed in the following.

3.2.1 Trading of Accuracy and Simulation Speed with Different Mesh Resolutions

When performing a numerical simulation with the methods mentioned above, a key factor to consider is the resolution of the mesh (cf. Figure 3.8) that is used to discretize the domain, which can have a significant impact on both the accuracy of the simulation results and the computational resources required. A finer mesh can often provide more accurate results, as it can better represent the geometry of the domain and capture gradients in the solution. However, a finer mesh also requires more computational resources, as it increases the number of equations to be solved at each time step. Conversely, a coarser mesh requires less computation, but it may not capture the solution accurately, especially where the solution varies rapidly over small spatial scales. Therefore, there is often a trade-off between accuracy and simulation speed, and the choice of mesh resolution depends on the specific requirements of the problem, such as the acceptable error tolerance and the available computational resources. [Versteeg and Malalasekera, 2007; Ferziger et al., 2020]

3.2.2 Adaptive Step Size

Adaptive time stepping is a method used in numerical simulations, such as fluid dynamics, to enhance accuracy and computational efficiency. It dynamically adjusts the size of the time step (Δt) based on the system's changes. Rapid changes require smaller time steps for accuracy, while slower changes allow larger time steps for efficiency. This adjustment relies on local error estimation and local flow velocity: if the error is large, the time step is reduced, and if small, it's increased. Adaptive time stepping enhances a simulation's performance, but also introduces additional complexity [Hairer and Wanner, 2008]. The time step size can have a direct impact on the *convergence and stability* of the simulation. Smaller time steps may lead to improved convergence and stability, but may increase computation time and storage requirements, while larger time steps may result in faster simulations at the cost of accuracy [Belytschko et al., 2014]. The user can decide when to let the simulation save intermediate results, at certain simulation time steps, percentages of filling.

3.2.3 Simulation Software A: PAM-RTM

PAM-RTM [ESI Group, 2023] of ESI is a software for simulating RTM processes and planning and testing process layouts within the broader software suite of PAM-COMPOSITES. As a commercially available simulation program for the representation of the injection process it has proven itself in both industry [Baskaran et al., 2014; Verrey et al., 2006], and academia [Luz et al., 2012; Grössing et al., 2016; Schmidt et al., 2021] as a simulation tool for the specific task of resin injection. Within the CosiMo project (cf. Subsection 2.3.4), it was also agreed to use this tool. Therefore, a part of the experiments for this work was implemented with this software. The permeability distribution, as well as the FVC, can be specified for each mesh element within a `.csv`-style file, which makes it possible to introduce changes in the textile programmatically for many instances.

Data processing from simulation To be able to train models on data coming from PAM-RTM, it needs to be processed first. PAM-RTM writes all data from the simulation to `.erfh5` files, which are a type of `.hdf5` files. Within those files, the data is structured in a tree format, of which three of four branches are important for further processing: (1) constant, (2) multistate and (3) singlestate. *Constant* includes all constant values, such as mesh data including coordinates of nodes, connections between nodes etc., but also textile properties such as permeability and FVC per mesh element. *Multistate* involves all sensory data, including pressure sensors and sensors at inlets and outlets. *Singlestate* contains the data from the advancement of the simulation: information about filling, flow velocity etc. at each node at given points in time.

To use the flow front data from these simulations with CV models, the mesh-based filling factor information needs to transformed to image data first. We used two different approaches for this task, of one is more precise and works for all mesh resolutions on the cost of speed and the other is fast but only works for high-resolution meshes. For the precise method we employed a contourplot function on the given mesh nodes. For the other one, enough nodes very available to get simply pack nodes that matched coordinate-wise into a normal fixed image grid as pixels, which is only possible when the nodes are dense enough. The resulting images have varying dimensions, between 149×117 and 135×103, depending on the use case. The images are grayscale and show the *filling level* or *filling factor* at each time step, which can be between 0 and 1. When the images are based on exact node information, within case study A (central injection) they show a gap in the middle, where the sprue is located. For the simulation, these nodes in the middle are used for injection and are not meshed.

3.2.4 Simulation Software B for Control of Process: Variant of RTMSim

Both proprietary and open-source fluid simulation programs exist that can model the RTM process and could be used as an RL training environment. However, the automation and parallelization capabilities of these programs are usually limited, as they tend to be designed to solve singular, complex problems. We tested the aforementioned PAM-RTM for the RL for RTM control approach [Wilfert, 2022] and identified two problems:

First, in PAM-RTM it is only possible to change the injection pressure during a running simulation to a limited extent so that the simulation often has to be restarted with adjusted initial conditions, which leads to a large loss of time and regular crashes. The ability to change parameters during the process is crucial for online control and needs to be possible without terminating the executing process to achieve high efficiency and stability. Second, PAM-RTM uses a thread-wise license model, which severely limits the number of parallel instances. Thus, it is impossible to train RL models in a parallelized and thus time-efficient manner with this software solution, as later shown in Chapter 7. So the simulation for the control of the process has several special requirements that differ from what off-the shelf software has to offer. Since a relatively simple case of the RTM process is considered in Chapter 7, which, however, has to be simulated very frequently and as parallel as possible, a finite volume simulation adapted and reduced to it is implemented following the example of the work of Obertscheider and Fauster [2023], which was also tested and approved experimentally [Obertscheider et al., 2023]. By a separate implementation, requirements to enable parallelization and compatibility to interfaces of existing libraries providing current algorithms of RL can be observed. The simulation is implemented in the language *Julia* [Bezanson et al., 2012], which was developed for scientific work, where, in contrast to languages commonly used in this area such as *Python* or *Matlab*, special focus was placed on performance.

Data from the simulation is serialized in binary form in Julia and can be imported directly in Python for RL training. All necessary or wanted aspects of the simulation can be stored that way: constant information about e.g. geometries, permeabilities or FVC but also dynamic or time-variant values such as pressure or flow front data. The information can also be exported as images.

3.3 Digital Twin

This section presents a preliminary discussion on the concept of a *digital twin*, touching upon its relationship with simulation and ML. Further, the application of this technology in the thesis is described: to implement quality control during the manufacturing of FRP components. At a fundamental level, a digital twin is a virtual reflection of a physical object or system, developed with real-time data and advanced simulation methodologies to optimize performance, resolve potential issues, and evaluate different scenarios [Singh et al., 2021]. These digital twins serve as mirrors of real-world assets, detailing both tangible objects and intangible elements information and services. The concept has gained momentum in the era of Industry 4.0, encouraging predictive maintenance, efficient resource allocation, and planning [Tao et al., 2018; Kuhn, 2017].

A digital twin can range widely in complexity: it can vary from enabling the replay of past production cycles (which involves elevating manufacturing data assessment to a level that allows detailed replay) to incorporating simulations [Li et al., 2022]. These simulations can also co-simulate the process based on current parameters, extending up to ML-based models that can co-simulate [Haghighat et al., 2021; Fitzgerald et al., 2019] or predict the future outcomes of the process based on inputs, much faster than FEM models. This capability allows for adaptive control and optimization of manufacturing processes [Tao et al., 2018]. Applications of digital twins with ML in manufacturing include predictive maintenance, where ML algorithms can identify patterns in sensor data to predict equipment failures, allowing timely maintenance, and reducing unplanned downtime [Lee et al., 2015]. An advanced digital twin can analyze product quality data to spot deviations, enabling immediate corrective actions and ensuring consistent product quality [Mourtzis et al., 2016; Stieber et al., 2021b].

If a manufacturing plant is still in the conceptual stage, a digital twin encapsulating its key features can already be established. Similarly, a workpiece in early production can have a digital twin that not only records its present properties but also encompasses the characteristics it will adopt upon completion. Digital twin technology can profoundly augment manufacturing processes by elevating efficiency, minimizing downtime, and enhancing product quality. Another primary incentive for digital twins is to facilitate a continuous exchange of data through standardized interfaces [Kuhn, 2017]. This data can later be employed to train ML models.

3.3.1 The Significance of Simulations in the Context of the Digital Twin

There is no universally agreed definition of a digital twin. But, contrary to some assumptions, digital twins need not merely be data repositories that store process data of real-world assets: in manufacturing it can be of different aspects, e.g. the process and its corresponding machines and tools, or the product itself. Instead, some digital twins also include algorithms that precisely depict their real-world analogues. These are often simulation models that might simulate the functional or physical attributes of the digital twin. When these simulation models use real data as a starting point, the digital twin ideally replicates the behavior of its real counterpart. However, the simulation-reality gap inevitably exists, a phenomenon that will be explored for this thesis' application in Section 4.5. Nonetheless, sophisticated high fidelity simulations can potentially replace actual tests with virtual equivalents. Complex systems can be virtually commissioned before completion, allowing for early programming optimization and saving time. In addition, costly tests, such as those for aircraft or automobiles, can be partially replaced by simulations. The data offered by digital twins can also be used in failure analysis to determine the cause of a potentially disastrous error that destroyed the real-world counterpart. If the digital twin allows access to not just current but also historical data, the working part's history and the defective part's history can be compared. This can help identify significant events during production or usage that might have led to the failure. Low-quality parts could therefore be detected early with the help of their digital twin and removed from the system. The most notable domain for digital twin applications continues to be production engineering [Li et al., 2022], offering several use cases. These range from specific processes such as friction stir welding [Sigl et al., 2020] and robotics [Oyekan et al., 2020], to applications in CFRP [Zambal et al., 2018; Stieber et al., 2023b], and general assembly [Zhuang et al., 2021; Demartini et al., 2019].

However, the application of digital twins extends beyond production. As far back as 2012, NASA researchers proposed the digital twin as a solution to soaring certification and testing expenses [Glaessgen and Stargel, 2012]. As certification processes required more complex testing, some actual testing was to be replaced by digital twin testing, which is achieved by merging real data and simulation models. The authors concentrated on the physical attributes of novel materials to demonstrate their robustness at critical points of, for example, a probe or an aircraft. They also emphasized that digital twins should have the capability to integrate different specialized simulation models to achieve the simulation accuracy required for such investigations. These differences or inaccuracies between simulation and real world, i.e. the simulation-reality gap can potentially be bridged using sim-to-real transfer learning, as previously outlined in Subsection 3.1.2.

Simulation is a crucial aspect of the digital twin, but the boundaries are not sharply defined. Typically, digital twins themselves can be perceived as a simulation technique, though not all simulations are digital twins. For instance, in aircraft manufacturing, digital twins must supply sufficiently precise simulation models from all suppliers to facilitate integrated simulation of an aircraft's load-bearing capacity. These would need to interact with each other to execute an integrated simulation. Additionally, it is important to note that different manufacturers utilize different tools to create simulation models. Thus, beyond data exchange, the algorithms used for simulation must be integrated with each other. Additionally, the production process must be reflected in the digital twins. If a part is not sufficiently resilient due to a production error, this should be recognizable with the help of the data and simulation models contained in the digital twin.

3.3.2 Integrating ML into the Digital Twin

Several trends have contributed to the surge in popularity of data-driven models in general:

- Significant advancements in sensor technology, data transmission, and data storage.
- A dramatic increase in computing capacity (e.g. Central Processing Units (CPUs), GPUs, and Tensor Processing Units (TPUs)).
- The application of ML.

The availability of open-source ML tools and frameworks reduce the cost and barriers to establishing such a model. The integration of ML allows these digital counterparts to learn from historical and real-time data, generating accurate predictions and identifying trends. ML models give the manufacturing industry the ability to address high-dimensional, multi-objective problems, such as composite manufacturing. [Brunton et al., 2021]. This is because data-driven models can extract potential correlations between a large number of variables from data without any prior physics knowledge, opposed to regular controllers, that are based on mathematical models, which need further feature extraction techniques to handle higher-dimensional input [Brunton and Kutz, 2019]. The basis of data-driven models is essentially a hypothesis that presumes the variability of data can potentially describe system behavior. The perspectives of physics-based and data-driven models are fundamentally different. The foundation of physics-based model are the fundamental principles of physics, such as equations of motion, thermodynamics, and fluid dynamics. Hence, they are usually accurate as long as the assumptions and boundary conditions hold true. They might not capture all complexities of real-world systems, especially when there are unknown factors or nonlinearities. For instance, in aerospace manufacturing, ML models are used to support production line robotics and automation [Al Khawli et al., 2018]; non-destructive inspection [Gardner et al., 2020] and aerospace system health status [Trusova, 2020]. They are also used to build surrogate models to replace parts for which it is challenging to create physics-based models, such as aero-engines [Wei et al., 2020] and aerofoil [Ocampo et al., 2017]. While discussing digital twins, various definitions are often presented, generally focusing on the machinery or the production system of a manufacturing process.

While this is a fascinating perspective, this thesis is concerned with two aspects of a digital twin for LCM processes. A duplication of the materials or workpiece itself: we are interested in the properties of the textile that contribute to an assessment of the quality of the current process, which is fully described in Chapter 6. This ML-based approach gives experts the possibility to inspect properties that indicate the quality of the future component from sensor data only. The second application of a digital twin in this thesis is the visualization of the current state of the process from sensor data (cf. Section 5.3): analyzing the current progression of the flow front and giving insights on potential problems, i.e. dry spots. We focused on the feasibility of the approach, but the resulting data can be stored in a uniform manner, if that is necessary. In summary, a digital twin can range widely in complexity and has numerous applications not only in Industry 4.0.

Chapter Summary and Outlook
In this chapter, the fundamentals and methodologies that will be further explored and utilized throughout this thesis were introduced. The starting point was an overview of ML, with an emphasis on the integral concept of supervised learning. This served as a stepping stone into the detailed exploration of several pivotal neural network architectures. We then turned our attention to transfer learning, with a particular focus on the crucial transition from simulation to reality. Additionally, an introduction to the realm of Reinforcement Learning was provided, setting the stage for the process control later in this thesis. Following that, we ventured into the principles of LCM simulation. We acquainted ourselves with the basic aspects of fluid simulation, reviewed a range of simulation tools and two types of simulation environments employed in our ensuing work were introduced. Finally, we unraveled the concept of the digital twin. We introduced this notion, offered our interpretation of the digital twin, and established its vital connection between ML and simulations. This vital comprehension paves the way for the forthcoming chapters, wherein the concept of the digital twin will assume a crucial part in the research and applications that we are set to unfold.

Chapter **4**

Process and Data Foundations

Summary. In this chapter, the process and data foundations for the study of resin flow in composite materials are described. We will explore two case studies (A and B) that focus on the FVC and in-plane permeability. To obtain real-world data for case study A, we utilized an optical permeameter employing the radial flow technique in conjunction with optical flow front tracking. By incorporating patches with altered textile density in the preform, we induced significant changes in fluid flow patterns. This approach allowed us to track the flow front using images that closely resemble those from FEM-based process simulations. We conducted two series of flow experiments, RealPermSmall and RealPermLarge, to generate meaningful data for training neural networks. Data augmentation techniques were employed to make the most of the limited real-world samples. To supplement the scarce experimental data, we also generated simulated data in several evolutionary stages. These datasets were crucial for enabling transfer learning from simulation to reality, with each stage aiming to provide more accurate and realistic flow simulations. Furthermore, the simulation-reality gap for LCM processes and our efforts to bridge it are discussed in detail. For case study B, we relied solely on simulated data in a process setup involving linear injection, allowing control over the flow front. This setup is explored further in Chapter 7. In summary, this chapter highlights the importance of incorporating realistic material properties, flow characteristics, and data processing techniques in both real and simulated datasets for training neural networks capable of analyzing fluid flow in composite materials.

Publication. The concepts from this chapter are published in [Stieber et al., 2021a,b, 2023a,b]

4.1 Case Studies

For this work, two different case studies on how to impregnate the preform were selected, enabling the applications that are presented in the following chapters. The primary distinction between *case study A* and *case study B* lies in the location of the resin injection in the mold: case study A features a central inlet port, while case study B employs a linear inlet port on one side. Another significant difference between the case studies concerns the sources of data.

For case study A, both simulation and real data are at hand, while only simulation data can be utilized for case study B. Both simulations use similar materials and both have inserts of different textile density to provoke alterations in the flow front, that can ultimately lead to dry spots (cf. Subsection 4.2.3).

4.1.1 Case Study A: Central Inlet Port

As mentioned in Subsection 2.1.1, there are various methods for injecting resin into textiles, and the injection port positions need to be adjusted for different target geometries [Arnold, 2017; Neitzel et al., 2014]. A *central inlet port* is widely used, partly due to its shorter cycle times resulting from the reduced paths the resin has to cover from the center compared to injections from the side. The 2D-optical permeameter (cf. Subsection 4.3.1), which is the primary source of real-world data for this thesis, also features a central injection. This setup was also employed in the CosiMo research project (cf. Subsection 2.3.4), which was another (, the initial) reason use this type of mold and injection configuration.

Case study A is utilized for both in-process analysis concerning future fluid flow and fault detection related to dry spots (see Chapter 5), as well as for analyzing product properties as part of the digital twin (see Chapter 6).

Overview and Characteristics We designed and implemented several versions of case study A (cf. Figure 4.1), with different LCM machines as examples which results inter alia in different shapes of the mold and different placements and numbers of sensors. For this work we only use one configuration family of case study A, which is closest to the permeameter of which we also have real world data, as described in Subsection 4.3.1. The dimensions of this setup are 375 mm in width and 300 mm in height in simulation which is comparable to the size of the textiles in the real-world permeameter experiments (390×290 mm).

One other configuration that is worth mentioning is described by Sobotta [2022] in a master's thesis. Here, he tried to apply the approach for dry spot detection on the data from the CosiMo project (cf. Subsection 2.3.4). Due to the lack of usable data and from this approach, we discuss the setup that is closer to the permeameter and give only a short description of how we analyzed the data from that project in Subsection 5.1.1.

For the simulation, we use a mesh that is quite fine-grained with $56,232$ mesh elements and $28,464$ nodes. This resolution is too high to show the mesh in detail here, that is why a simplified version of the simulation is shown in Figure 4.1.

Sensors and Actuators In case study A, it is essential to distinguish between data from simulation and data from reality, i.e., the permeameter. In simulation, we could place as many pressure sensors as desired, which was not done in the real world. We placed $1,140$ sensors in an equidistant grid (38×30) on the upper mold surface, which we then utilized for process analysis, as seen in Chapter 5.

This grid size and spacing resulted from placing sensors at a 1 cm distance from each other, in both the x and y directions. While this number of sensors is not entirely realistic due to costs and hardware or wiring constraints, we considered it a suitable starting point from which sensor omission would be possible and an easy way to test various setups. The different levels of quality in predicting with different numbers of sensors are described in Section 5.1. Additionally, the image of the flow front progression is readily available in simulation. From the permeameter we mainly used images of the flow front with the corresponding time stamp.

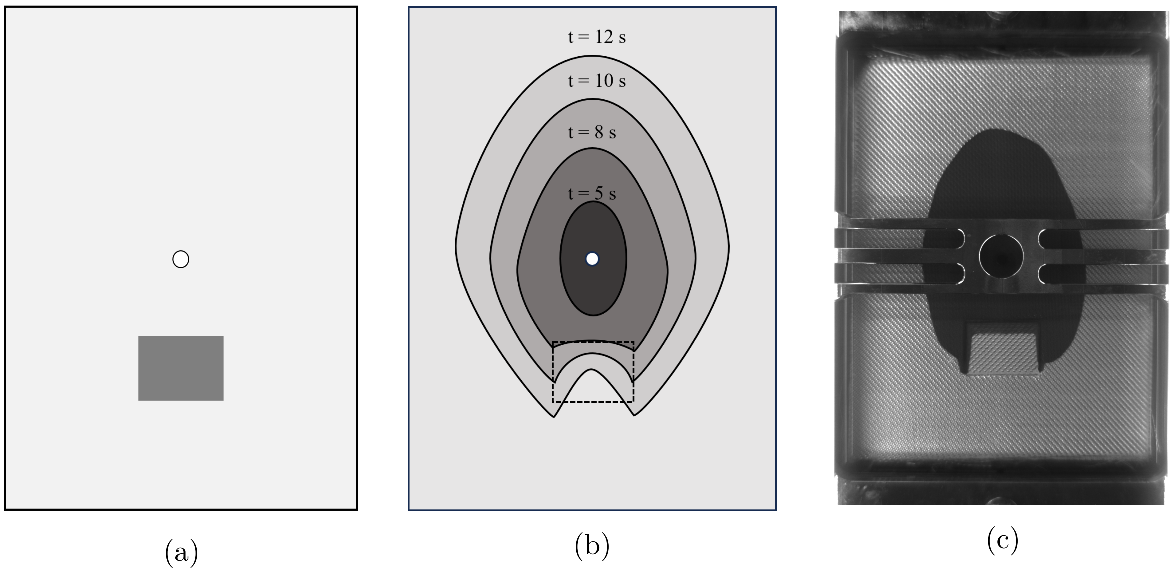


Figure 4.1: Case study A: LCM process similar to permeameter: (a) Map of change in textile FVC (b) (Idealized) simulation at different time steps (c) Real data point from permeameter

Since it is not possible to change the form of the flow front when only one inlet is available, case study A does not have an actuator that can control the flow front.

4.1.2 Case Study B: Linear Injection

Case study B was designed to show the possibilities of steering the flow front in an LCM process, described in detail in Chapter 7. It has a linear injection on the left side and an outlet on the right. Case study B was specifically designed for Stieber et al. [2023a] and was the central case study for Heber's bachelor thesis [Heber, 2022].

Overview and Characteristics In the spatial domain, we use a mesh that consists of $1,878$ triangle elements and models a planar quadratic part, as depicted in Figure 4.2. The part has a side length of 500 mm and a thickness of 5 mm. We chose to employ this comparably smaller mesh size to minimize simulation times (cf. Subsection 3.2.1), which was crucial for the control task described in Chapter 7. A finer mesh, resulting in lower simulation speed would have made efficient in-the-loop RL training impossible.

Sensors and Actuators As sensory information, we utilized the progress of the flow front and the pressure field, which is described in detail in Chapter 7. Both are readily available as 2D image data in the simulation.

Three almost equally wide resin inlets (not completely equal due to discretization with a coarser mesh) of 158 mm (Inlet 1), 184 mm (Inlet 2) and 158 mm (Inlet 3) width, that are independently controllable, are placed on the left side that enable the steering of the flow front (cf. Figure 4.2: areas with differently colored mesh triangles).

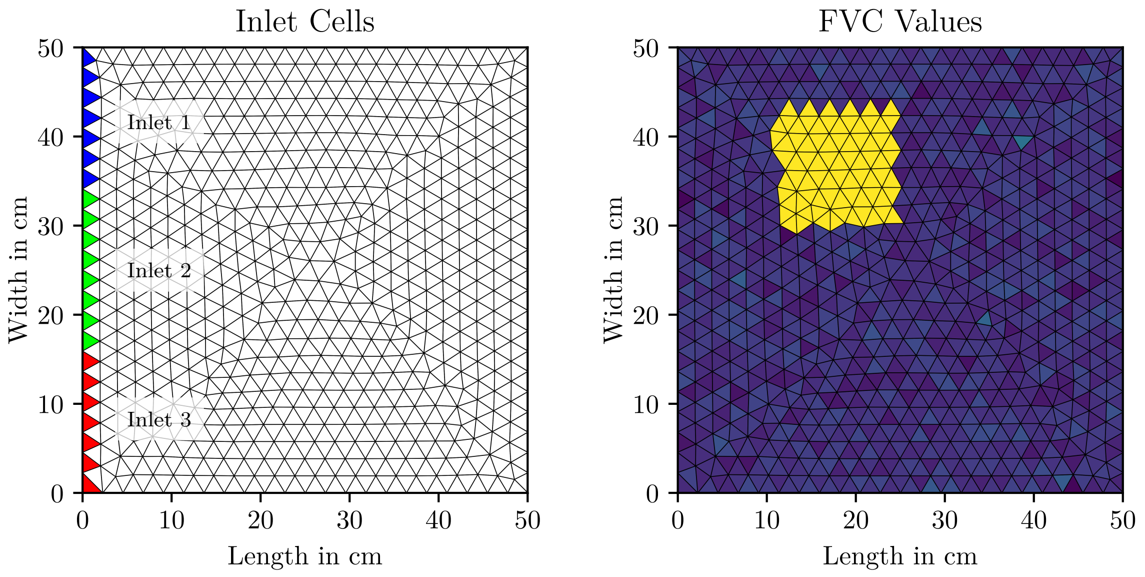


Figure 4.2: Case study B - Linear injection: Left: Three different inlet cell groups that can be actuated independently are depicted in different colors. Right: FVC contents in different areas of the preform. Patch with higher FVC is shown in bright yellow.

4.2 Materials

In the following section the materials in use for both case studies are described: fluids, textiles and whole preforms.

4.2.1 Test Fluid

In the flow experiments for case study A, conducted on the optical permeameter, standard plant oil was employed as a test fluid, which is a common practice in experimental permeability characterization, such as the IBE [May et al., 2019] to avoid handling chemically reactive thermosets. The viscosity of the test fluid was experimentally characterized in a temperature range between 15 and 30 °C, covering the typical range seen in non-conditioned research labs. In this range, a nearly linear decreasing trend was observed, ranging from about 90 to 48 mPas. This viscosity is comparable to that of uncured epoxy resin, which is commonly used in RTM and is relevant during the impregnation of fibrous preforms. In the flow experiments, the fluid temperature was measured in the feeding line and interpolated into the aforementioned characteristics to obtain the fluid viscosity relevant for each experiment. We utilized a value of 65 mPas (referring to room temperature), for the simulation runs. The test fluid was colored with red pigment (Sudan Red IV, Sigma Aldrich) to improve the contrast between saturated and unsaturated preform regions in the digital images. Comparative rheometric tests confirmed that the color pigment does not affect the test fluid's viscosity.

For case study B, a viscosity of 100 mPas was used, which is also in the same range as the viscosity for case study A.

4.2.2 Fabric

In case study A, the permeameter experiments employed a glass fiber, 2/2 twill woven fabric, specifically Hexcel's Hexion 1202, featuring a nominal areal weight of 290 g/m^2 and in-plane permeability characteristics as outlined in Table 4.1. This commercially available material has been well-documented in recent IBE studies on in-plane permeability characterization by May et al. [2019]. The in-plane permeability data used for the simulation runs were obtained by interpolating the corresponding results of principal in-plane permeability k_x and k_y (cf. Subsection 2.2.5) versus φ_f (fiber volume content, also abbreviated as FVC). The material exhibits principal flow directions closely aligned with the orientation of the woven fiber bundles, resulting in $k_{xy} \approx 0$, as seen in (2.6). The simulation parameters were tuned to match this setup.

For case study B, a natural fiber was used as a model for the configuration. This textile promised to offer quasi-isotropic permeability behavior and was thus suitable to design an isotropic and thus manageable case (more on isotropy and anisotropy in Subsection 2.2.6). The permeabilities for that textile vary between 1.464×10^{-9} m^2 and 2.268×10^{-10} m^2 [Berg et al., 2015].

4.2.3 Preform and Patches

One central motivating idea of this thesis is that the quality of FRP parts made with LCM processes suffers from strong variability in the source material, the textile. This variability stems from changes in produced batches of the textile - for one type of textile, the permeability can vary up to 20 % [Tifkitsis and Skordos, 2020; May et al., 2019], and handling errors (that lead to e.g., fringing of fiber bundles, breakage of filaments) can also lead to higher or lower FVC in certain areas of the preform. These variations of the permeability of the preform are not necessarily stemming from unwanted behavior of the process but can also be a result of the architecture of the preform itself. In Subsection 2.2.4, an overview on ply stacking sequences or "ply books" is given, that can include different layer architectures or inserts for sandwich constructions, that can drastically interfere with the flow front. To summarize, it is possible to have both, unwanted and wanted changes in the permeability or FVC of the preform and we will use both assumptions, having knowledge about the changes or not, in the following chapters. To create a dataset that certainly has these strong variations, we added inserts of changed FVC in the textile.

In case study A, the real world flow experiments and resulting data detailed in Section 4.3 were conducted using specially manipulated preforms of the woven fabric. A rectangular patch of the fabric was introduced at a specific location within the preform, with variations in: (i) the number of layers, (ii) the orientation of the fabric, (iii) the FVC, and consequently, (iv) the in-plane permeability. The experimental test plan for the permeameter data is described in Subsection 4.3.3. Table 4.1 outlines the most relevant properties of the preform and patch region. Multiple patch locations were selected for the flow experiments to generate a meaningful dataset for subsequent neural network training (cf. Subsection 4.3.3). Figure 4.3 illustrates the geometric specifics of such a configuration. For the simulated data for case study A, we tried to produce data as close as possible to reality to narrow the simulation-reality gap. The process to acquire the best dataset possible for this use case is described in Section 4.4.

In case study B, we also incorporated modifications into the textile in the form of patches. The base FVC value was fixed at 35 %, resulting in a preform permeability K of 1.464×10^{-3} mm^2.

Table 4.1: Preform setup for real-world trials

Region	# of layers	φ_f [%]	k_x [mm^2]	k_y [mm^2]
Preform mat.	11	41.7	72.47×10^{-6}	17.21×10^{-6}
Patch region	16	60.7	1.19×10^{-6}	15.32×10^{-6}

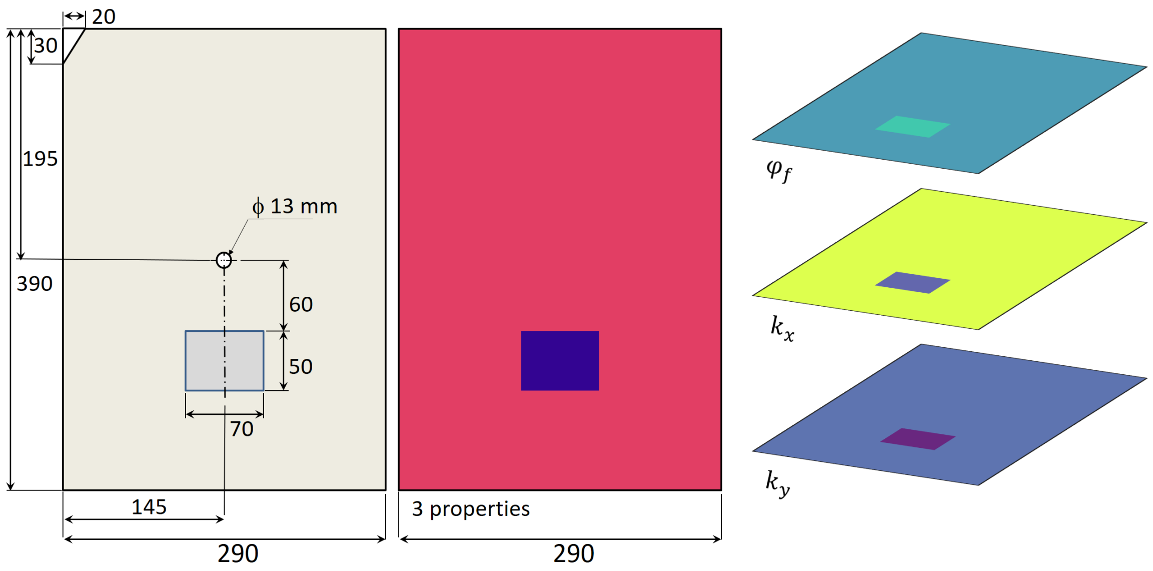


Figure 4.3: Geometry of experimental material, Left: Geometric measures of an exemplary chosen test configuration; upper left corner is a cutout to track position and rotation/flips of resulting images. Middle: Three properties combined as channels of the output image. Right: φ_f, k_x and k_y.

In related studies, it has become common to use rectangular inserts or patches of altered textiles to induce and analyze flow front perturbations [Wang et al., 2018; Stieber et al., 2021a]. The position, dimensions, and extent of FVC deviation determine the difficulty of the control task. During training, the placement is derived from a random distribution, subject to specific constraints in each experiment, which will be explained in Subsection 7.2.5. Details regarding the FVC level within the patches are also described in that section.

4.3 Data from Flow Experiments

In the following the real data generation process for case study A is described. Alas, we were unable to produce data from real world experiments for case study B.

4.3.1 Optical Permeameter

For case study A, real world data was generated using an optical 2D permeability characterization cell (cf. Subsection 2.2.5), or permeameter, which employs the radial flow technique combined with optical flow front tracking. Utilizing an optical permeameter provides the benefit of tracking nearly the entire flow front in the form of planar *images* rather than sensory time series

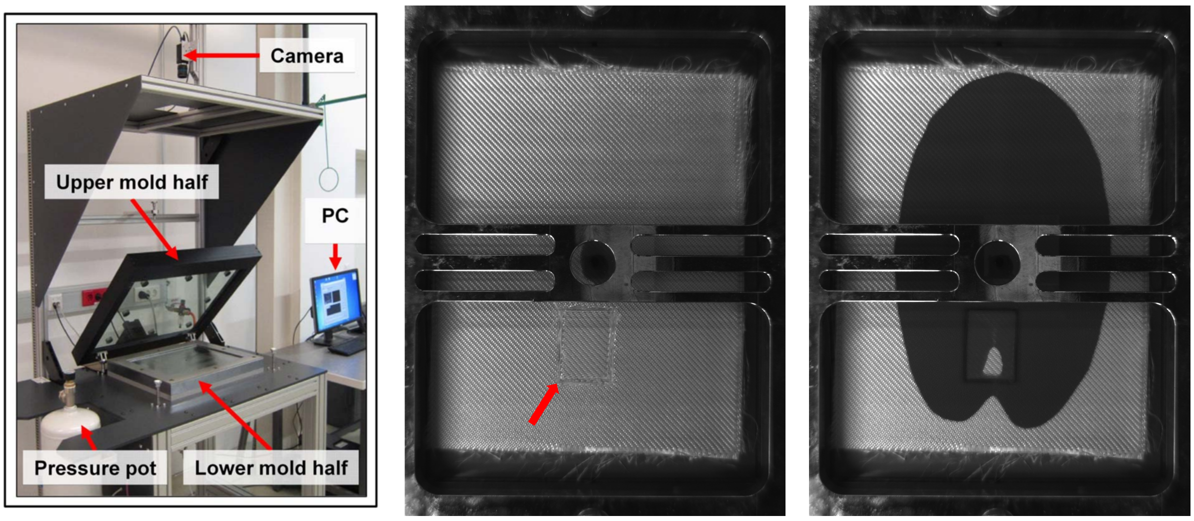


Figure 4.4: Permeameter experiments: Left: In-plane permeameter test rig featuring an optically transparent upper mold half. Middle: The onset of the injection trial in the permeameter, with a dark spot visible between the metal frame. The patch is highlighted with a red arrow. Right: The injection is ongoing, and a noticeable deviation of the flow front is caused by the patch with altered permeability.

at specific locations only. In current industrial practices, it is common to even have no in-mold sensors at all. Furthermore, the acquired flow front images (see Figure 4.4) closely resemble those from an FEM-based process simulation, enabling a sim-to-real transfer approach.

The experimental data employed in this work was gathered using the permeameter depicted in Figure 4.4. The mold of this cell can be viewed as a flat plate RTM mold with a central injection gate (Case study A) and an optically transparent top mold half. This setup allows for flow front tracking using an industrial camera system as shown by Fauster et al. [2019].

4.3.2 Image and Data Processing

For sim-to-real transfer learning (cf. Subsection 3.1.2), which is a goal of many applications of case study A, we needed to utilize the experimental data in a model trained solely on simulation data, thus the acquired image sequences needed to undergo fixed pre-processing (see Figure 4.5): (a) The images were rotated to align them with those generated by the simulation, resulting in a image resolution of 1392×1040, b) empty zones outside the fabric were removed, image areas displaying the stiffening frame of the mechanical setup were eliminated, and c) parts of the flow front, obstructed by the stiffening frame, were supplemented using a specially developed automated mechanism. This latter step involves fitting an elliptical geometry model to selected data points along the fluid flow front [Fauster et al., 2019] and extrapolating parabolic models for the major and minor ellipse axis lengths [Fauster et al., 2018], respectively. A description of how pseudo-sensory data was extracted from permeameter images, that are necessary for sim-to-real transfer learning experiments on fault detection in the form of dry spots (cf. Section 5.1), can be found in Section 4.5.

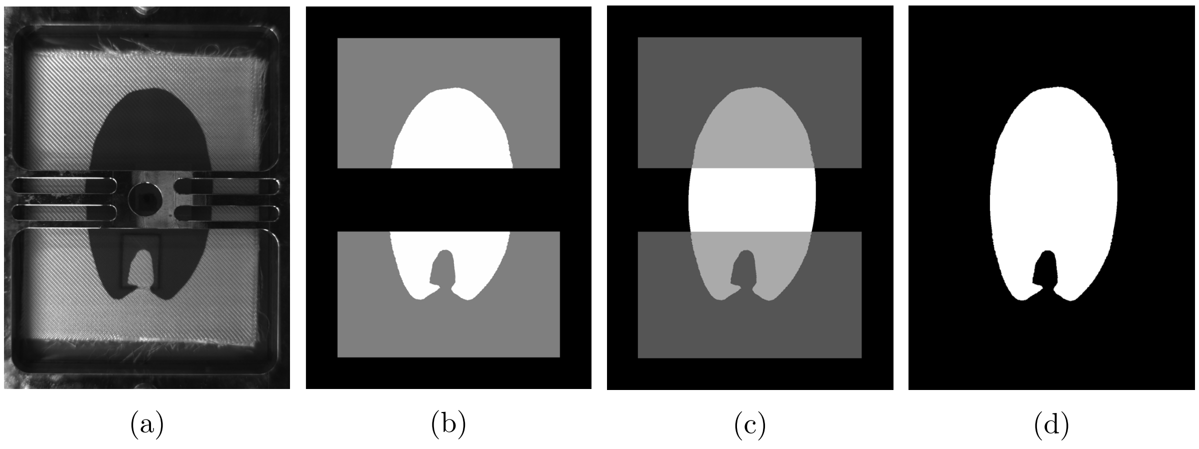


Figure 4.5: Evaluation/Preprocessing procedure on an example image: raw camera 1040 × 1392 image (a), b/w image with cut out center frame (b) and locally completed image of flow front (c) and actual label image, which is used for training, dimension: 111 × 143, shown in (d).

4.3.3 Experimental Test Plan and Data Augmentation

For case study A, two sets of flow experiments were conducted specifically, which are referred to as RealPermSmall (from 2020) [Stieber et al., 2021b] and RealPermLarge (from 2021) [Stieber et al., 2023b], with the resulting datasets having the same names. For simplicity, both resulting datasets combined are called RealPerm. Table 4.1 gives an overview of the textile used for the permeameter experiments.

During the development process, we underwent several iterations of simulation and patch rearrangement to identify the optimal placement for the patches. It was crucial to ensure that the patches were not positioned too far out, as the injection process stops when one border of the textile is reached, and also not too close to the metal frame in the middle to minimize obstructions for further imaging procedures down the preprocessing pipeline. The final positions can be found in Figure 4.7.

Additionally, the number of layers in the patch and preform was initially tested in simulation, followed by mathematical analysis to determine if there could be potential issues due to excessive pressure on the upper mold half at the patch area, which could potentially cause damage to the glass mold. Eventually, we settled for 16 layers of fabric in the patch area and 11 layers in the remaining preform.

Figure 4.6 shows the resulting distributions of φ_f from the experiments. Figure 4.7 shows in short form of the full test plan for the permeameter tests for RealPermLarge. Initially, we planned to carry out experiments with only 20 configurations, but the outcomes were not as expected: During the RealPermSmall experiment campaign in 2020, all six runs resulted in the formation of a dry spot, the patch was located as shown in Figure 4.7, configuration 1/13/17. In 2021, for RealPermLarge, however, the same theoretical configuration did not lead to dry spots, but only a leading flow front or racetracking along the edges of the patch. That is why another nine configurations were added (see also Figure 4.7), resulting in overall 29 experiment configurations. The nine additional configurations have the same placement as the configurations before, but an additional gap between patch and preform was added.

To maximize the value of this limited number of real-world samples, the injection experiments were designed so that the patches were situated in only one quadrant of the preform. Data

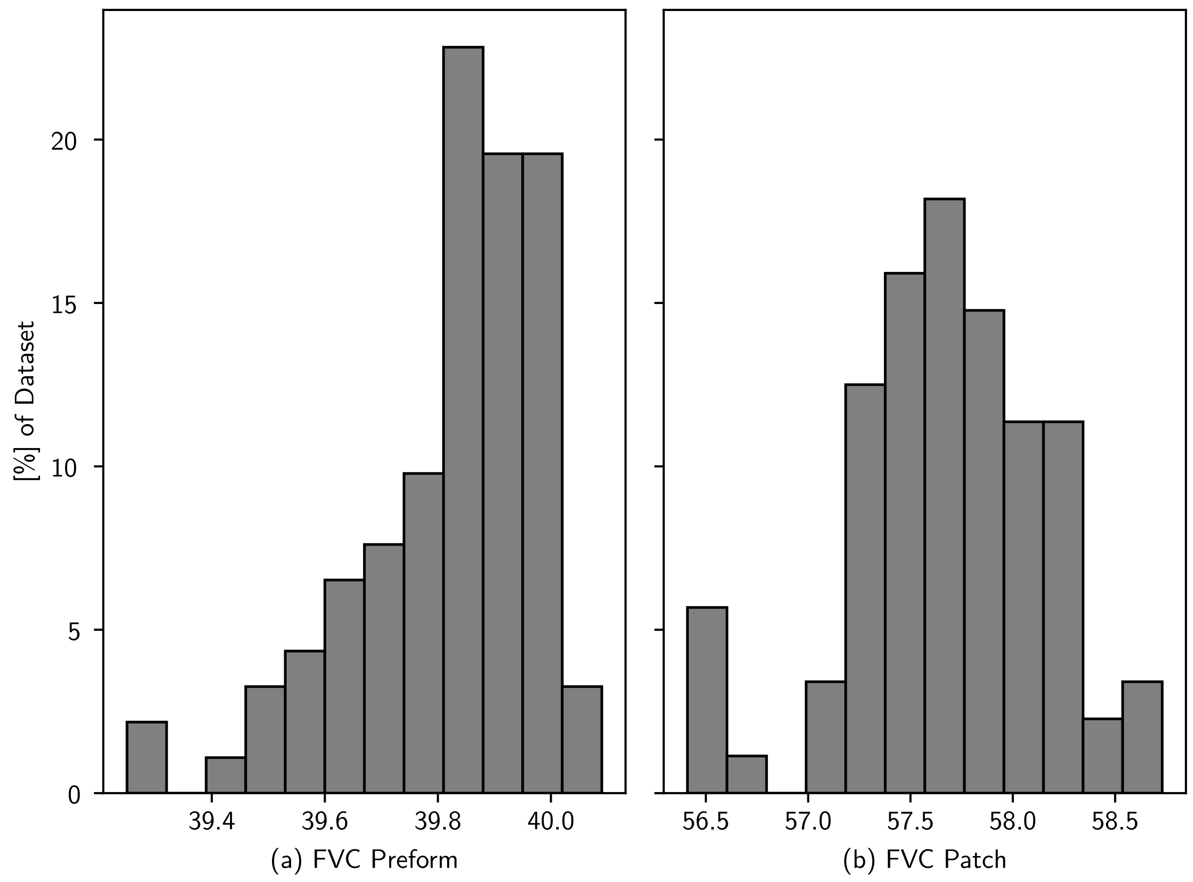


Figure 4.6: Measured distributions of FVC in (a) preform and (b) patch in RealPermLarge

augmentation was performed by flipping the resulting image sequences along the $x-$ and $y-$directions, respectively, which generated additional image sequences.

Data Split for Data Set from Reality In this study, the training, test, and validation sets were fixed across all experiments for dataset RealPerm. The test set comprised all real data samples flipped both in the x and y directions. The remaining three flip configurations (not flipped, flipped in x, and flipped in y direction) formed the training and validation sets.

This data split is reasonable considering that the models employed in this study utilize convolutions for feature extraction. Convolutions exhibit invariance to translation, but not to rotation, as discussed by Weiler and Cesa [2019]. Consequently, the models must be able to generalize effectively in order to accurately predict dry spots or the future flow in locations where the flow front direction differs from the training data. This approach ensures that the models are robust and can provide reliable predictions in various scenarios.

4.4 Data from Flow Simulation Runs

Experimental Test Plan for Case Study A For case study A, we employed a strategy to obtain data from simulation runs, also referred to as the *simulated* data, which we have described

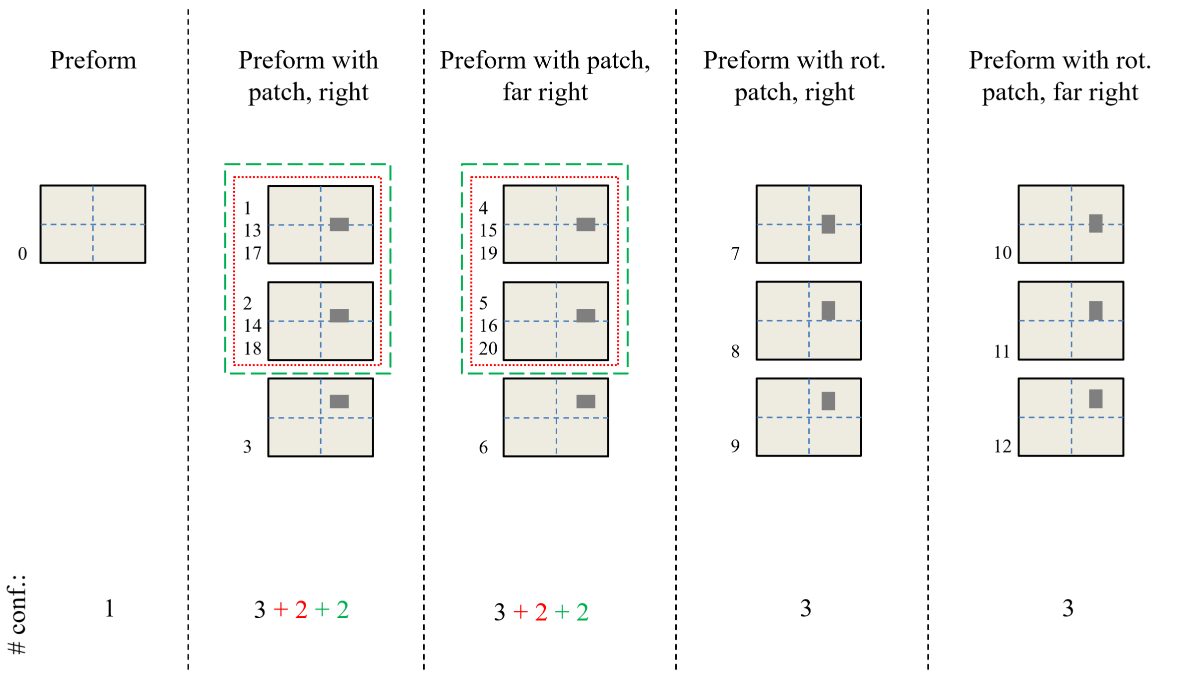

Preform
Preform with patch, right
Preform with patch, far right
Preform with rot. patch, right
Preform with rot. patch, far right
0
1 13 17
2 14 18
3
4 15 19
5 16 20
6
7
8
9
10
11
12
# conf.:
1
3 + 2 + 2
3 + 2 + 2
3
3

Figure 4.7: Overview of test configurations. Description on top, followed by a row of visual representations of the positioning of the patch within the preform. Next to the small tiles the IDs of the corresponding configurations are shown. On the bottom the numbers of configurations is shown. Configurations 1, 2, 4 and 5 are surrounded with red and dotted or green and dashed boxes, because they were used with different channel widths, i.e. different patch sizes.

in other works, particularly Stieber et al. [2021a,b]. This strategy involves an automated pipeline: Starting with a two-dimensional representation of a preform with anisotropic material properties, or simply termed *preform*, small patches (either rectangular or circular) with varying in-plane permeability and φ_f were randomly inserted to change the material setup in different runs and thus perform a type of domain randomization (cf. Subsection 3.1.2) from the beginning. Subsequently, fluid flow through the preform was numerically predicted using PAM-RTM (cf. Subsection 3.2.3), resulting in a sequence of "label images", or briefly called *labels*. They were sampled over the filling percentage of the preform.

Simulation data were generated in several stages, and Table 4.2 provides an overview of the evolution of datasets created for this work. The difference in φ_f between SimAniso and SimAnisoChannel, respectively, and the predefined setup specified in Table 4.1 stems from slight deviations between the nominal and real values of the fabric's areal weight. The latter was measured directly before initiating the flow experiments and was found within the tolerances specified in the material's technical data sheet.

The most crucial aspects of the datasets are detailed in the *description* column of Table 4.2, showcasing an evolution towards more realistic flow simulations. The first campaign, SimIso in 2019, represented an initial attempt targeting a single image classification task, where sampling over time was not considered. The SimIsoCustom dataset from 2020 introduced regular sampling but still relied on isotropic material with (unrealistic) high variability in φ_f. With the creation of the SimAniso dataset in 2021, the simulation moved closer to reality: preform properties were chosen according to real flow experiments, and anisotropic in-plane

Table 4.2: Datasets - Simulation

Dataset	# of experiments (ca.)	φ_f Preform [%]	φ_f Patches [%]	Textile	Description
SimIso	40,000	26.8	$20 - 80$	Natural Fiber (iso.)	First dataset, for [Stieber et al., 2021b]
SimIsoCustom	10,000	26.8	$20 - 80$	Natural Fiber (iso.)	Custom uniform subsampling, for [Stieber et al., 2021a]
SimAniso	10,000	40	$56 - 61$	HexForce1202 (aniso.)	Closer to reality through use of data from flow experiments
SimAnisoChannel	10,000	40	$56 - 61$	HexForce1202 (aniso.)	Introduction of flow channels to represent “race-tracking” effects

permeability (cf. Subsection 2.2.6) was introduced as an additional layer of complexity. Finally, SimAnisoChannel represents a dataset covering geometrically small (i.e., 1 - 2 mm) flow channels along the patch edges to reflect race-tracking, (cf. Figure 4.6). This latest dataset is not only more realistic due to the added specificities but also because of the distribution of failed vs. successful runs. This increased realism arises from modeling the dataset after the real measured property distribution of the textile, which results in fewer failed runs. Consequently, this dataset provides a more accurate representation of real-world scenarios and better prepares the ML model for practical applications.

Experimental Test Plan for Case Study B Regarding case study B, we did not need to produce bigger datasets since we trained the model in an RL way. We conducted two series of experiments: During training, the insert parameters are drawn from experiment-specific random distributions, which will be explained in this section. To be able to compare agents within one series of experiments, we evaluate them on fixed test sets of 100 parts, that were created according to the same distribution as used in training.

In the first set of experiments inserts are placed and varied in such a way that *slight perturbations* of the flow front occur when injecting the fluid with steady pressure into an isotropic natural fiber textile. Thus, the flow front lags significantly when it passes an insert, but no dry spots are formed. Rectangular inserts of height 21 ± 1 cm and width 16 ± 1 cm are used for this purpose. These are placed randomly on the component with the restriction that a strip of 5 cm width is excluded from each of the left and right edges. The setup of both experimental campaigns, especially the differences are presented in Table 4.3. The FVC of the insert is 42 %, which equals a permeability of $3,969 \times 10^{-4}$ mm^2. The results are presented in Subsection 7.3.1.

In the second series of experiments, *strong perturbations*, were provoked that led up to the formation of dry spots. In preliminary tests, agents tended to have greater control of the flow front in the first third of the part than in areas far from the resin inlets. This can be explained by Darcy's law (cf. Equation (3.21)), according to which the flow velocity is proportional to

Table 4.3: Characteristics of slight perturbations and strong perturbations experiments.

	Slight Pert.	Strong Pert.
# and shape of insert	1 Rect.	1 Rect.
Height of insert in cm	21 ± 1	15 ± 1
Width of insert in cm	16 ± 1	15 ± 1
φ_f preform in %	35	35
Perm. preform in mm^2	$1,464 \times 10^{-3}$	$1,464 \times 10^{-3}$
φ_f insert in %	42	45
Perm. insert in mm^2	$3,969 \times 10^{-4}$	$2,268 \times 10^{-4}$
% dry spots	0	54

the pressure gradient. Close to the inlets, this can be strong and immediately changed by applying different pressure values, while the influence decreases when the flow front is farther away. Therefore the inserts are placed only in the left third of the part to test whether an agent of RL is in principle able to prevent the formation of dry spots. The inserts are of square shape and have a side length of 15 ± 1 cm. The inserts have a FVC of 45 % and a permeability of $2,268 \times 10^{-4}$ mm^2. They were placed 5 cm from the inlets and 15 cm from the outlets, since perturbations closer to the inlets made it possible to still change the flow front sufficiently. Additionally, a 5 cm margin is applied to the upper and lower edge to avoid cases where a dry spot touches a border.

An uncontrolled injection process leads to strong irregularities of the flow front as soon as it reaches an insert. In 54 % of the cases within the test set of strong perturbations, this ultimately leads to the formation of a dry spot, causing the filling cycle to be prematurely terminated. It is noteworthy that the difference between slight and strong perturbations is only 3 % in FVC. The amount of change in FVC necessary to provoke enclosures of dry textile, i.e. dry spots was determined experimentally. The results of these experiments are described in Subsection 7.3.2.

In the following subsections, specific characteristics of flow simulations in preforms with (i) anisotropic in-plane permeability and (ii) inhomogeneities introduced by patches with properties deviating from the preform are described. These aspects are particularly significant for the application of ML techniques in the FRP domain, to bridge the simulation-reality gap.

4.4.1 Isotropic and Anisotropic Textiles

General isotropy and anisotropy is described in Subsection 2.2.6. Here, additionally, the orientation of the patch is irrelevant, as it does not impact directional permeability values.

Regarding the simulation here, in Figure 4.8 (left), the impact of anisotropic in-plane permeability in a preform with a rectangular patch is highlighted with two different patch locations. The patch exhibits a higher number of fabric layers, resulting in a higher level of φ_f. Moreover, the orientation of the fabric layers in the patch is rotated by 90° concerning the remaining preform, indicated by arrows pointing in the direction of the major in-plane permeability. The top left image shows the expected effect: the flow front trails inside the patch region due to reduced permeability in the vertical flow direction. However, the bottom left image shows the opposite situation: the flow front inside the patch region leads compared to the surrounding preform. This effect is caused by (i) the specific relation of in-plane permeability

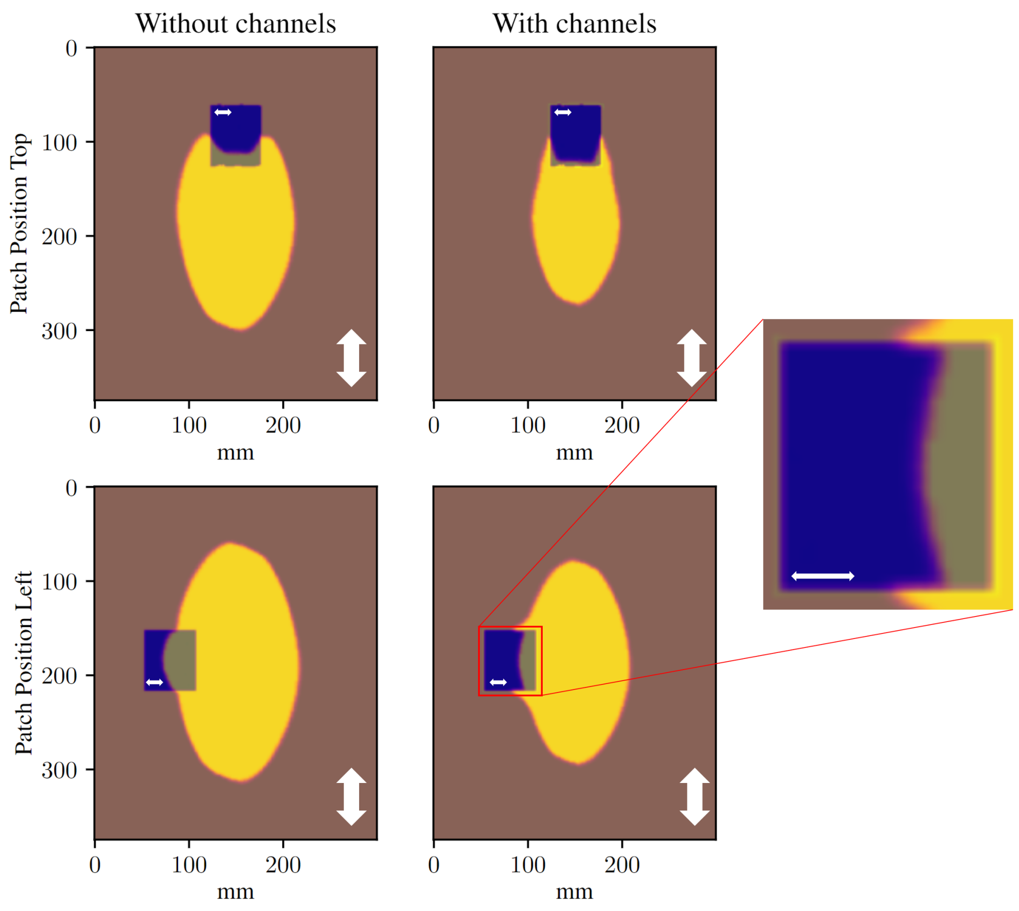


Figure 4.8: Race-tracking explanation. The brown part is the unsaturated preform and the yellow part is the saturated preform. The blue rectangles show patches with changed FVC and permeability. The orientation of the major in-plane permeability in the preform and patch regions is indicated by arrows. The left column shows the filling factors from fluid simulation for two different patch locations are shown, revealing the effect of anisotropic in-plane permeability. The right column shows the impact of considering race-tracking channels on the flow pattern in the fluid simulation for two different patch locations. The patch location directly impacts the flow front behavior. The race-tracking effect causes the fluid to preferentially flow along the edges of the patch, leading to a different flow front behavior than what would be expected solely based on the anisotropic in-plane permeability. By taking into account race-tracking channels, the simulation can better capture the complex flow patterns observed in real experiments.

values (see Table 4.1) and (ii) the level of pressure gradient driving the fluid flow in this particular region of the preform and patch, respectively. Although numerically correct, this flow pattern is not observed in the flow experiments. Instead, the flow front trails inside the patch region, which can be explained with race-tracking.

4.5 Bridging the Simulation-Reality Gap in LCM

After presenting both data sources, real data from a permeameter and simulation, the differences between both are described and the steps we took to narrow the gap between simulation and reality.

4.5.1 Race-Tracking

The simulation environment typically implements the edges along the patch region as "ideal", meaning that there are no gaps between the preform and patch, and the mesh properties change instantaneously from a cell of the preform to a cell in the patch region. However, this does not accurately reflect reality, or can be described as a typical *simulation-reality gap* within this domain. In Subsection 2.2.5, the effect is mentioned the first time, but it is described here in more detail, referring to the challenges it caused for the applications in this thesis. In real-world scenarios, despite the best efforts in manual handling, a gap remains between the preform and patch, forming a flow channel around the patch. This channel, typically in the size of 1 to 2 mm, shows a level of axial permeability that is usually one to two orders of magnitude higher than the surrounding fibrous material, causing the fluid flow to advance along the edges of the patch. This effect is well-known for LCM and referred to as *race-tracking* [Bickerton et al., 1998].

To better represent this effect in the flow simulation, 2 mm wide flow channels were added around the rectangular patches in the dataset SimAnisoChannel. The permeability of the corresponding mesh cells was specified as isotropic and one order of magnitude higher than the major in-plane permeability of the preform. The results of these changes, as shown in Figure 4.8 (right), align well with the observations from the flow experiments. By incorporating the race-tracking effect into the simulation, it provides a more accurate representation of the complex flow patterns observed in real-life situations, narrowing the simulation-reality gap.

4.5.2 Sensors in Simulation and Reality

In simulation, pressure sensors are the only readily-available sensor type to detect the filling factor inside the mold. In the permeameter used in the RealPermSmall and RealPermLarge dataset, there are no pressure sensors available, but the flow of the fluid can be observed with a camera. The filling factor can be derived from the images by applying image processing techniques described in Subsection 4.3.2. To use the data from the permeameter experiments at hand, available after preprocessing as black and white images, some further transformations are necessary. For this we map a grid of 30×38 points onto every image, representing the sensors placement from the simulation datasets. Then the information whether the flow front has reached the sensor is extracted, derived from the brightness of the corresponding pixel(s). Furthermore, all 6032 frames from 96 runs were labeled regarding dry spots: 704 samples contain a dry spot - approximately 11.67 % of 6032 samples in total as shown in Table 4.4.

4.5.3 Timing Differences

Temporal resolution While we initially trained the model solely on simulated training data and only used single frames for prediction (Stieber et al. [2021b]), the temporal resolution became more important when using sequential data (Stieber et al. [2021a, 2023b]). As described in Subsection 3.2.2, the timing in FEM is adaptive. Within the simulation solution PAM

Table 4.4: Dataset sizes

Dataset	# Runs	# Samples	# Samples w/ dry spot	% (Samples w/ dry spot)
RealPermSmall	6	315	94	29.84
RealPermLarge	90	5717	610	10.67
RealPerm	96	6032	704	11.67

RTM, the rate of when the intermediate results are saved can be adjusted by the user. For the first dataset SimIso, the timing was not fixed per a certain time interval but per filling percentage. That means that initial steps covered much longer time intervals than later ones in the simulation, resulting in strong variations of the time interval per step. Starting with dataset SimIsoCustom, the timing was fixed to one second per frame. Additionally, the time information stemming from the simulation software does not show precise outcomes [Grössing et al., 2016] and the comparability to reality is not always given [Magagnato et al., 2013; Rodrigues de Oliveira et al., 2013], which is another point to keep in mind when using time information from FEM simulations. The real dataset contains more than one image per second but to align the data from reality and from simulation, we only used a subsampled set of 100 frames, as described in Subsection 4.5.4.

Length of injection runs In reality, the runs are shorter than in the simulation. Whenever the flow front hits a border of the textile in reality, the flow front does not advance any further since it can flow through that connection to the outlet, which completely surrounds the mold. Thus permeameter experiments are stopped when the border of the textile is met. To overcome this difference, simulation runs were randomly shortened in a certain value range that reflected the length of the real data runs.

4.5.4 Additional Differences between Simulation and Real-World Data

The images from simulation and reality had to be aligned regarding size and rotation to enable training models from both sources. Aside from the fact that the simulation data has pressure sensors and the real-world data only contains images, there are additional differences between the two datasets concerning the size and number of alterations in the textile. As described in Section 4.4, the simulation data is automatically generated with parameters and boundary conditions that match the observations from real-world data. This led to the dataset SimAnisoChannel being similar to the real-world data, with anisotropy and race-tracking channels involved. Nonetheless, there are some differences that will be discussed in the following.

1. Number of data points Regarding the amount and variety of data, the simulation data is much more abundant than the real-world data, which is one of the main reason to employ sim-to-real learning, to make use of deeper models for low-data tasks (cf. Subsection 3.1.2). There are $11,000$ runs in the SimAnisoChannel dataset, whereas there are only 96 runs in the RealPerm dataset. Each run has several hundred time steps or frames, and for some tasks, these steps are sampled down to 100 frames per run. Each frame represents one percentage of the whole preform filled, so the number of frames is downsampled by filling percentage.

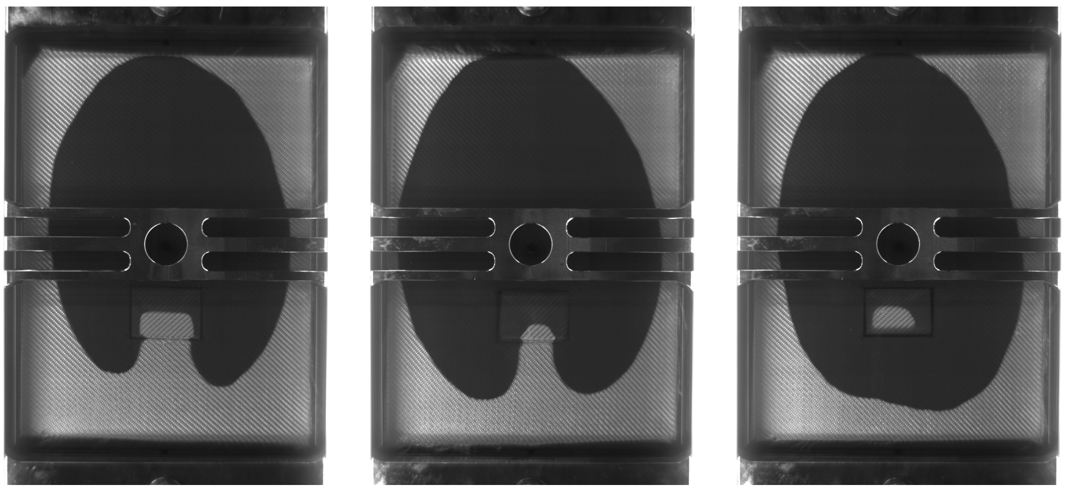
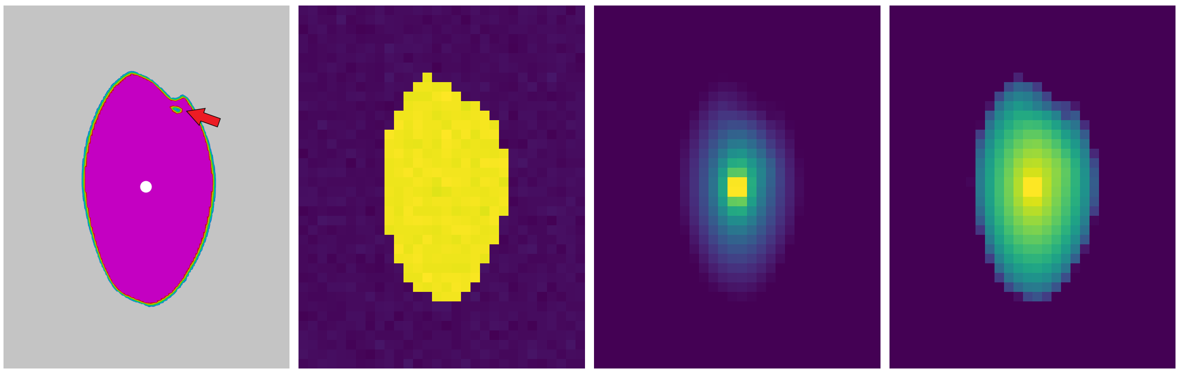

(a) Channel width 0 (b) Channel width 1 (c) Channel width 2

Figure 4.9: Samples from the RealPerm dataset taken at the same point in time. The patches are in the same location, but the channel width is different: 0, 1 and 2 mm, resulting in different flow front behavior.

2. Size, form, and position of textile changes The simulation data has patches of varying FVC in the form of one rectangle of side length ranging between 4 and 8 cm and an optional circle of radius between 2 and 4 cm. The dimensions of the patches in the real data experiments are in the same region, with side lengths between approximately 5 and 7 cm. For more details, see Section 4.3. Additionally, the patches in the real-world data are all in the same area of the preform. That was planned this way (a) to enable data augmentation and (b) because the space of the preform is limited because of the steel beam in the middle of the permeameter. For the simulation, the patches are randomly placed in the preform, which is a more realistic scenario.

3. Anomalies in real world data There are also other anomalies regarding race-tracking and length of runs within real data only, that are described in detail in Subsection 5.2.3. Overall the simulation data has more variants and is more complex than the real-world data. To illustrate the differences some samples from the datasets are shown. First we inspect a sample from the RealPerm dataset, which is a single frame from a run. All 20 patch location configurations from the dataset have similar sizes of the patch: it is shown in Figure 4.3. For a channel width of 0 mm, size of the patch is 50 × 70 mm, for channel sizes 1 and 2 mm it is 48 × 68 mm and 46 × 66 mm, respectively (cf. Figure 4.7). These patches result in larger disturbances in the flow front. Figure 4.9 shows three samples from the RealPerm dataset, with the same placement of cutouts from the preform, but different sizes of the surrounding channel and thus slightly different sizes of the patches. Here, always the last frame of each run is shown, when the flow front hits one of the borders, in all three cases the upper border. In Figure 4.9c we can see that the flow front forms a dry spot when the channel size is large enough. Next, a single image sample from the SimAnisoChannel dataset in Figure 4.10 is shown.

We can see that there is a dry spot visible in the flow front image in the top right. In the image of the binary sensor output (Figure 4.10b), the dry spot is not big enough or at the



(a) Flow front (b) Binary (c) Pressure (d) Log pressure

Figure 4.10: Sample from the SimAnisoChannel dataset: (a) shows the image of the flow front with small but visible dry spot in the top right, indicated by the red arrow. (b) is the binary image of the sensor grid, from the filling factor, all values less than 1 are set to 0. (c) shows corresponding pressure sensor output. For (d), 0.1 was added to all output values and the logarithm was applied to increase the visibility of the pressure data, which can be very small next to the flow front.

wrong position to show up. From the output of the pressure sensors (Figure 4.10c), there is more information visible, also for the human observer, especially when the logarithm is applied (Figure 4.10d). Whenever a CV task seems feasible for a human, it often can be automated. But if it is not, it is really challenging for an algorithm to accomplish it. [Janiesch et al., 2021] The latter is the case for the dry spot classification within the binary sensor output (cf. Section 5.1).

To conclude, the binary sensor output is not always sufficient to identify a dry spot. When pressure sensor information is available, distinguishing between acceptable and unacceptable frames becomes more feasible, which is interesting for the task described in Section 5.1.

Chapter Summary and Outlook

In this chapter, the data science and process foundations were presented. We investigated two case studies (A and B) involving the flow of resin in composite materials, specifically focusing on the FVC and in-plane permeability. To obtain real data for case study A, an optical permeability characterization cell (permeameter) was used, following the radial flow technique combined with optical flow front tracking. Patches of changed textile density were placed in the preform to provoke strong alterations of the fluid flow. This enabled the tracking of the flow front in the form of planar images, which closely align with those from an FEM-based process simulation. Two series of flow experiments, RealPermSmall and RealPermLarge, were conducted to provide meaningful data for neural network training. Data augmentation was employed to maximize the benefit from the limited number of real-world samples. Due to limited real data from experiments, we also produced simulated data in several evolutionary stages, as shown in Table 4.2.

Although a fairly “basic” situation with strictly two-dimensional flow in a plate-shaped mold and homogeneous fabric material was considered for case study A, flow simulations meeting the real-world situation require a decent amount of expert knowledge. However, such flow simulations scale very well. Flow experiments, by contrast, are very costly in a laboratory or industrial environment, and they scale only linearly. As a result, limited data is available.

These simulated datasets were necessary to enable transfer learning from simulation to reality. The evolution of these datasets aimed to provide increasingly realistic flow simulations, ultimately resulting in the SimAnisoChannel dataset. This dataset considered anisotropic material properties, race-tracking channels, and property distributions based on real textile measurements. Incorporating anisotropic in-plane permeability and race-tracking channels in the simulations allowed for more accurate predictions of fluid flow (cf. Chapter 5). To combine data from flow experiments and simulations, image and data pre-processing were performed, which included alignment, cropping, and padding to create sequences of the same length.

For case study B we used only simulated data within a process setup with a linear injection that allows to control the process in terms of the flow front, which is further discussed in Chapter 7.

Chapter **5**

In-Process Analysis and Fault Detection

Summary. In the following chapter, various forms of in-process analysis for LCM processes, which have been experimentally conducted and evaluated, are presented. In addition to detecting issues during injection, a method for reconstructing the flow front from sensory input is introduced. Moreover, potential approaches for predicting the future flow front are discussed. For both of these tasks, data from simulation and reality is utilized and the possibilities to apply sim-to-real transfer learning are tested. The first part discusses the importance of detecting and preventing dry spots during the RTM process (part of the LCM process family) which can result in poor quality, ultimately increasing production costs. This section explains how dry spots occur when certain areas of the preform are not wetted by the liquid, leading to incomplete mold filling, air entrapment, and weak or incomplete parts. In-situ monitoring using sensors can help improve quality assurance by tracking the flow front of the fluid and predicting spatial deviations from proper RTM cycles, including the presence of dry spots. The chapter also presents different forms of in-process analysis of the RTM process and provides results of validation on a real-world dataset from a permeameter. The second part discusses the purpose of a process predictor for LCM, which is to predict the behavior of an LCM process in simulation and reality, and to test the ability to improve forecasts on real data with sim-to-real transfer learning. The section explains why a predictor of the future flow front is a useful supervisory part of the digital twin, to be able to predict future faults or other problems. The approach is an image-based prediction, a deep CNN that predicts the next future flow front state based on the current flow front state and information about the textile's properties.

Publication. The concepts and results presented in this chapter are published partly in [Stieber et al., 2021a].

In this chapter, two methods for analyzing LCM processes (including RTM) are presented: (1) dry spot detection, which serves as a form of fault identification, and (2) process forecasting, which aids in anticipating future faults. This information enhances process understanding and offers additional perspectives, enriching the digital twin of the process. To achieve this, a technique for reconstructing the flow front using sensor data is introduced. For both methods, data from simulations and real-world scenarios are utilized, and attempts are made to employ sim-to-real transfer learning in each case.

5.1 Detection of Dry Spots

During the injection phase of the RTM process, numerous factors can significantly compromise the quality of the resulting product, or even render it unusable, thereby increasing the overall production costs [Sorg, 2014; Heuer et al., 2015; Liu et al., 1996]. One important disruptive factor can be high input variances of the fiber contents in the textile or preform (cf. Subsection 4.2.3). *Dry spots* refer to areas of the preform that are not wetted by the liquid, as shown in Figure 4.9c. This can lead to incomplete mold filling, air entrapment, resulting in weak or incomplete parts. Sometimes these imperfect components can be repaired manually but most often they end up as scrap. Either way, automated process monitoring based on sensors applied to the mold that do not interfere with the process itself, can improve the quality assurance. By tracking the flow front of the fluid and predicting *spatial* deviations from proper RTM cycles, it is possible to indicate problems such as dry spots resulting in air traps, providing diagnostic insight, or even control actions to avoid rejects (which is discussed in Chapter 7). In this chapter, different forms of in-process analysis of the RTM process are presented. Aside from identifying problems during injection, a way to reconstruct the flow front from sensory input is presented. The results are validated on a real-world dataset from a permeameter. The discrepancies between simulation and reality that need to be addressed have been discussed in Section 4.5.

5.1.1 Binary Classification of Injections, Description and Evaluation

As described in Chapter 4, we used several different types of datasets to test the performance of different ML algorithms. Initially, we obtained data from simulation, which shows some differences to real-world data, described in Section 4.5. Most importantly for the following use case, the simulation data includes pressure that can be extracted at certain sensor positions while the real-world data has flow information that can be discretized into single "flow front sensors". Also, the simulation data is not noisy, which is a common characteristic of real-world data in general. Having a different type of sensory output makes it necessary to adapt or even retrain the ML algorithms to the new data. Flow front sensors, derived from images, only give binary output (wet or not wet) while pressure sensors give a continuous output, that is richer (cf. Figure 4.10) and thus can be handled much better by NNs. In [Stieber et al., 2021a], we trained deconvolutional and convolutional NNs to produce an image of the flow front from the pressure sensor input in supervised fashion (see the left part of the network highlighted with a green box in Figure 5.1). After reaching sufficient flow front image quality, froze the parameters of this first part, appended and trained the binary dry spot classification appendix, highlighted with the orange box in Figure 5.1, so this approach is not trained end-to-end, but in two steps. This network classifies wether the image of the flow front contains a dry spot or not.

Excursus: Generative Adversarial Networks (GANs) for flow front reconstruction For the generative part of FlowFrontNet, we tested another approach. Nik Julin Nowoczyn investigated in his bachelor thesis [Nowoczyn, 2021] the suitability of a GAN for generating flow front images from pressure sensor data. He trained and evaluated a conditional GAN based on Pix2Pix [Isola et al., 2017] and tested it on different sensor resolutions. He found that the two flow front image generating methods are relatively similar in terms of performance. Therefore, and also because of the complicated training of GANs compared to the supervised approach of the deconvolutional network, this approach was not pursued further.

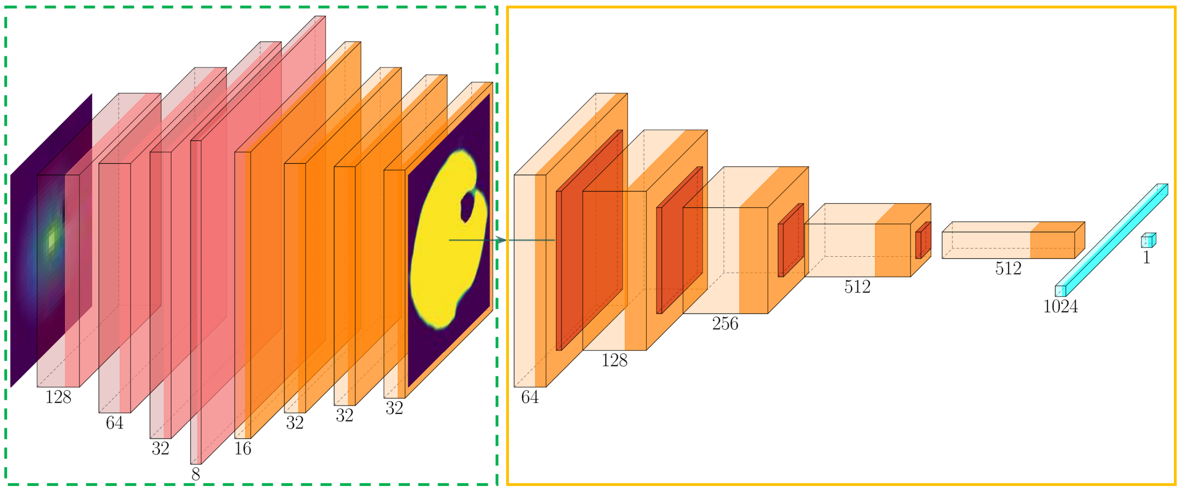


Figure 5.1: FlowFrontNet: The initial part (dashed, green) is a deconvolutional and subsequently convolutional network that maps sensory data to images of the flow front. The second part in solid/orange is the binary classifier network consisting of five convolutional/max-pooling layers and two fully-connected layers. Numbers below the boxes indicate the depth of the resulting feature maps. Figure from Stieber et al. [2021a].

Table 5.1: Results for *single-frame* Binary Classification of Injections. Metrics are shown in %

Model	Dataset	Accuracy	Bal. Accuracy
Feed Forward Baseline	SimIso	82.74	-
FlowFrontNet	SimIso	91.68	-
CNN	SimAnisoChannel	98.00	89.40
FlowFrontNet	SimAnisoChannel	98.11	88.75

Table 5.2: Accuracies of three different sensor resolutions for FlowFrontNet, dataset: SimIso

# Sensors	1140	285	80	20
Distance [mm]	10	20	30	40
Accuracy [%]	91.68	89.40	83.69	75.22

The results of this two-step approach were promising, but the network was only trained on simulation data (dataset SimIso, cf. Table 4.2) due to the absence of real-world data. We managed to achieve an overall accuracy of 91.68 % when classifying single images. The results are also shown in Table 5.1. For comparison, we trained a baseline model which consisted of a flattening layer first and two subsequent fully-connected layers, which achieved an accuracy of 82.74 %. The dataset that we used, SimIso, was balanced because it had almost equal numbers of wet and dry images, so the metric accuracy is a good indicator of the network's performance.

The input consisted of 1140 sensor readings. However, it's essential to note that in real-world situations, using fewer sensors is often more practical, as evidenced by the decrease in performance demonstrated in Table 5.2. The term *distance* refers to the gap between the sensors in the mold along both principal axes. We persisted in employing the maximum sensor count in subsequent investigations, operating under the assumption that this would yield the greatest precision. Nevertheless, such a high sensor count is not feasible in industrial environments due to both economic and spatial constraints. The requirement for each sensor to be wired necessitates integration into the mold, which only has limited available space.

There exist alternative flow front reconstruction methods [Di Fratta et al., 2013; Konstantopoulos et al., 2014; Achzet et al., 2022], that claim to use less sensors. These approaches would need to be tested for the necessary accuracy, but the methodologies presented later in this thesis would require minor modifications to enable their usage, given their reliance on the image representations of the flow front or the pressure field. Further, this specific form of flow front reconstruction is limited to closed mold processes, such as RTM. For other process groups within LCM, like VARI, a transparent upper mold half allows the direct visual access to the flow front, making it available for additional processing [Lekanidis and Vosniakos, 2020; Matveev et al., 2021].

Upon retraining the network with the latest SimAnisoChannel dataset, which was designed to better resemble real-world data, it became necessary to reconsider the evaluation metric due to a heavy imbalance favoring good runs without dry spots. Consequently, balanced accuracy was utilized, which represents the average of the recall achieved for each class, i.e., how many of the actual positive instances were recognized as positive and how many of the actually negative instances were recognized as negative. The following acronyms are used, that are common in binary classification: true positive (TP), true negative (TN), false negative (FN) and false positive (FP).

$$\text{Balanced Accuracy} = \frac{1}{2} \cdot \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right) \tag{5.1}$$

When using the two-step *FlowFrontNet* approach, the balanced accuracy was 88.75 % (accuracy 98.11 %) and when using a single-step approach, containing a straightforward CNN, the balanced accuracy was 89.40 % (accuracy 98.00 %). As a result, this metric is only slightly different from the one used on the SimIso dataset. Alas, when using the flow front sensors instead of the pressure sensors, with the same number and positioning, the balanced accuracy was only 70.41 % (accuracy 97.30 %) with the FlowFrontNet approach and 72.14 % (accuracy 97.29 %) with the single-step CNN approach on the SimAnisoChannel dataset. This is a clear indication that the binary data does not yield enough information to classify the injections, which will be discussed further in the next section. Beforehand, the performance of the single-step CNN approach on real-world data, without any transfer learning, is shown. On RealPerm the balanced accuracy was 99.18 % (accuracy 99.20 %), which is an extraordinarily good result. However, this is not surprising, since the dataset is very small and the data is easier to classify than the simulation data. The RealPerm dataset features only one patch in rectangular form, whereas the simulation dataset often involve two patches in different forms (circle and rectangle). Other than that, the patches are located in a relatively small area of the preform due to the construction and the constraint that the filling procedure finishes earlier than in simulation.

Excursus: Sensor value forecasting in CosiMo
Sobotta [2022] tried to detect dry spots in the real data set coming from project CosiMo (cf. Subsection 2.3.4) in his master thesis. To do so, a sim-to-real approach was planned, where the simulation came from project partners. Due to several issues with both the dataset from simulation and reality, we refrained from detecting dry spots within that context but focused

on the prediction of sensors data from other sensors. If one sensor could be predicted from others, this sensor could be saved. That was a use case where sim-to-real transfer learning was applicable and for which the dataset was usable for. To summarize the results, CNNs with only one convolutional layer showed the best performance for the prediction of the sensor values. The temperature sensors can be successfully predicted by others with a concordance correlation coefficient (CCC) value between 0.95 and 0.99. The metrics measure significant agreement of the prediction with the test set, but very similar results for the models with and without transfer learning.

5.1.2 Sim-to-Real Experiments with Binary Sensor Data, Description and Evaluation

Aside from single frame classification, the analysis and classification of full cycles (*sequences*) are presented. This classification approach focuses on the entire injection process, rather than just a single frame as in Subsection 5.1.1, based on the data from binary flow front sensors. It proposes a method to retrospectively determine whether the injection process was successful or not. This approach can be more robust, since a threshold ε of consecutive single frames, above which a run is classified as containing a dry spot or not ok is introduced. With a lower ε, the classifier is more eager to return the dry spot class. After empirical experimentation, 5 emerged as the best ε for this task, as a trade off between a too eager and a lagging too far behind. For this task, each run is sub-sampled to 100 frames (as described in Subsection 4.5.4), each frame indicating one per cent in flow front progress, which means 5 consecutive frames indicate 5 % of the process. That is why we resorted to subsampling and using 100 frames per injection for several tasks, not just the classification of one single process. This results in a significant reduction of data, with only 11,000 runs of 100 frames in SimAnisoChannel and 96 runs of 100 frames in RealPerm.

We use a simple transformer network *SeqClassTrans* to classify these runs, which is shown in Figure 5.2. It is based on the ViT architecture [Dosovitskiy et al., 2020], that uses positional encoding. The input is a sequence of 1140 flow front sensor values (which are reshaped to 2D, 38×30), and always has the same length of 100 frames, that are stacked and then fed into the positional encoding. In the transformer block, the empty frames at the end of the runs, when they are prematurely ended, are masked, so they do not have an influence in the self-attention (cf. Subsection 3.1.1.5) of the transformer. Afterwards, it goes into the classification block, which is a Multi-layer Perceptron (MLP) or linear layer from the embedding size to one class.

With this approach, fine-tuning shows much better results than training from scratch, which can also be seen in Table 5.3. The main metric, balanced accuracy, increases by 11.96 % from

Table 5.3: Results for Binary *Sequence* Classification of Injections.

Model	Dataset	Accuracy [%]	Bal. Accuracy [%]
SeqClassTrans	SimAnisoChannel	87.00	85.30
SeqClassTrans	RealPerm	82.29	86.97
SeqClassTrans w/ fine-tuning	RealPerm	**95.83**	**97.37**

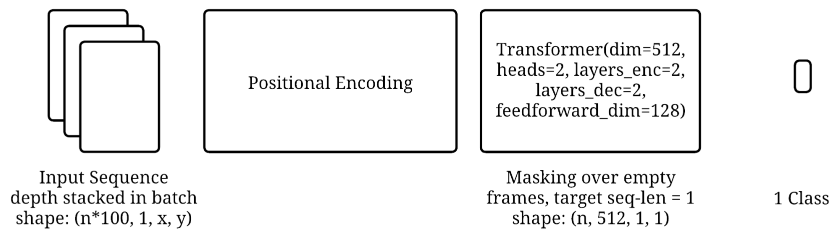

Positional Encoding
Transformer(dim=512, heads=2, layers_enc=2, layers_dec=2, feedforward_dim=128)
0
Input Sequence depth stacked in batch shape: (n*100, 1, x, y)
Masking over empty frames, target seq-len = 1 shape: (n, 512, 1, 1)
1 Class

Figure 5.2: Transformer architecture used for binary sequence classification: SeqClassTrans

86.97 % to 97.37 %, which is a considerable leap. To elaborate here, we can discuss absolute numbers: the TP went up from 19 to 20, TN 60 to 72, FP down from 16 to 4, and FN down from 1 to 0. To summarize, the model improved from 17 misclassifications down to 4 out of 96 samples. When comparing between different models and datasets, the metrics can show little fluctuations when compared directly. The accuracy is increased by 16.45 %, from 82.29 % to 95.83 %. This shows that transfer learning gives decent results for this binary sequence classification. Other use cases for this approach are explored in the following chapters.

5.2 Process Forecasting

The goal of this approach is to predict the behavior of an LCM process both in simulation and in reality and to test the ability to improve forecasts on real data with sim-to-real transfer learning. This would also open the possibility for model based control approaches that interfere with the injection online, which is valid for both Model Predictive Control (MPC) and model-based RL. For the prediction, that includes the textile information, we assume that the ply stacking sequence is available (cf. Subsection 2.2.4) for the current process.

5.2.1 Purpose of a Process Predictor for LCM

A predictor of the future flow front is useful as (1) a supervisory part of the digital twin, and (2) to be able to predict future failures to enable to steering against them. We implement an image-based prediction model, similar to the one implemented by Thuerey et al. [2020]. We train the model to learn the subsequent image of the flow front in a run, and based on that, we predict subsequent images. We train a deep CNN that predicts the next future flow front state, based on the current flow front state and property maps: permeability in x and y direction and FVC.

5.2.2 Data Set Description, Model Architecture and Training

For the prediction of future flow fronts we used both real data (cf. Section 4.3) and data from simulation (cf. Section 4.4) from *case study A* (cf. Subsection 4.1.1), referencing the central inlet port. The dataset from real flow experiments was RealPerm and the simulation dataset is SimAnisoChannel, the most sophisticated and close to reality dataset that we developed over time, as described in Section 4.4. Predicting the future flow front behavior was an ongoing

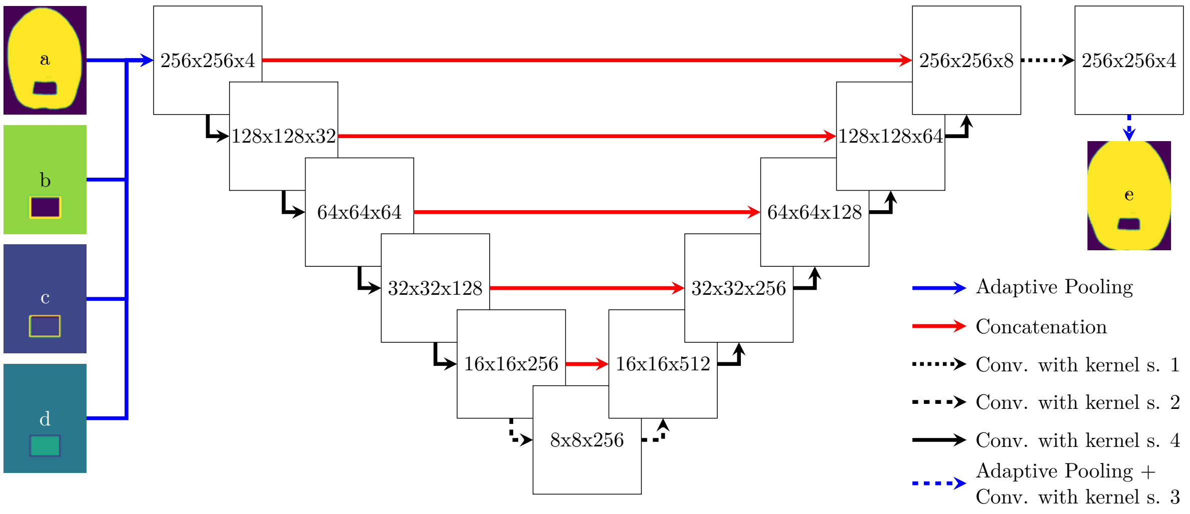


Figure 5.3: U-Net based model structure including parameters of convolution layers. From four input images of (a) flow front, (b) FVC, (c) permeability in x and (d) y to next (e) flow front.

task, that offered several challenges. The differences and the steps towards alining both data types are described in Section 4.5.

The model of choice is a heavily modified version of a U-Net model [Ronneberger et al., 2015], that was adapted by Thuerey et al. [2020] to work in predicting flows in a Navier-Stokes environment. The architecture of the model is displayed in detail in Figure 5.3. It consists of two parts, a downsampling part and an upsampling part. To use the model with different sized input samples, from simulation (different sizes of images from simulation are described in Subsection 3.2.3) and reality (cf. Subsection 4.3.2), the images are first adaptively pooled into images of dimensions 256×256. For downsampling, we use six convolutional layers, each followed by a leaky Rectified Linear Unit (ReLU) activation. All six layers in the upsampling block are transposed convolutions. Skip connections are added between corresponding upsampling and downsampling layers by concatenating the output of the downsampling to the input of the upsampling layers. Batch normalization is added between every layer within the encoder and decoder blocks. A simple head is added to the upsampling layers to allow for a last skip connection from the original input image to the output of the network and smoothing of the output images. This was done under the assumption that the reconstruction of future flow fronts is an additive process. Already filled parts of the mold are never evacuated again, such that the model only needs to learn adding upon the edge of the flow front.

After an initial experimental setup based on Thuerey et al. [2020], tweaked towards RTM with the datasets SimIso and SimIsoCustom, several steps were taken to improve the model and to adapt it to the LCM use case featuring real data in RealPerm that are described in the following.

Prematurely ending injections The simulation data sometimes behaves unrealistically when the form is already filled, so we cut off simulation runs, when the average filling state of the simulation reaches 99 % for the first time. The transformation from mesh-based simulation data to image data, that is used here is described Subsection 3.2.3. Only states before are used to train and evaluate the models. This does not affect the real data, since the runs already end at max. 75 % filling (cf. Subsection 4.5.3).

Shape of dry spot Initially, we had a reoccurring problem discovering dry spots with checkered filling formed. We assume that it happened because the sprue was not filled (was a black spot in the middle of a large filled area) and had a permeability of 0 in the input and label images. So the network had to keep the this false "dry spot" open all the time, which influenced the task to learn. The model learned to keep the form of the sprue unfilled and bigger dry spots were built of multiple such forms, the little borders between made it look like a checkered flag sometimes. This spot existed from the simulation data, the mesh was not filled there, it was used as injection point (cf. Subsection 3.2.3).

The model is trained three different times. (1) On real data only. (2) On simulation data only. (3) On simulation data first and then fine-tuned on real data. All these models are trained for 100 epochs with the AdamW optimizer. During fine-tuning, the learning rate was reduced by two orders of magnitude, from 1×10^{-4} to 1×10^{-6}.

The models were evaluated on a set of real world samples with the same split as introduces in Subsection 4.3.3. To be able to achieve comparable results between different model types and test the models ability to predict further into the future, the models outputs were used as model input up to 10 times and the MSE between ground truth from within the run and the corresponding predicted image was measured. This was done for every possible subset of images within a cycle resulting in 40 to 60 images being usable for evaluation in each run.

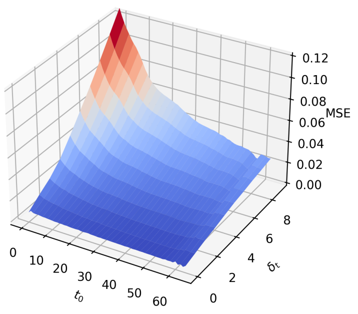
5.2.3 Evaluation

The approach is tested on real data samples: we show that transfer learning from simulation to reality helps with the problem of flow front forecasting.

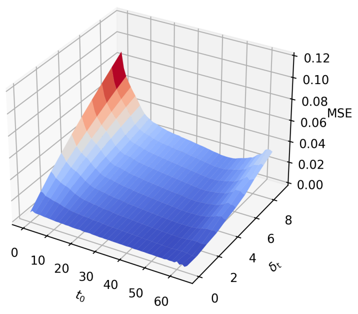
Fine-tuning the model shows a clear increase in model performance from a mean MSE of 1.559×10^{-2} to 1.421×10^{-2}, a decrease of 9.65 %. A model trained only on simulation data also achieves a significantly better result than a model trained on real world data samples from 2.407×10^{-2} to 1.559×10^{-2}, despite never having seen real data examples (an decrease of 54.41 %). This can be attributed to the fact, that the amount of available training data is two orders of magnitude larger for the simulated examples.

For comparison, the error when using the model that was trained on simulation data only, on a sim-only test set is only 3.678×10^{-3}. For completeness we also evaluated the other models as well on the sim-only test set: the model that was trained on real data only achieved an error of 1.788×10^{-2} and the fine-tuned model achieved an error of 1.107×10^{-2}. That means, the sim-only trained model was the best on the sim-only test set, which is not surprising. Other than that, all models showed a lower mean error on the sim test set compared to the real test set, which indicates that the real data is more challenging to forecast. Additionally, the sim-to-real model works better than the real data trained model, which also does not come as a surprise. This shows the potential of models trained exclusively on simulation data for transfer learning. Despite being trained solely on simulation data, these models demonstrated superior performance compared to models trained only on the task of predicting real-world data. This superiority indicates better generalization capabilities in these models, which makes them excellent candidates for fine-tuning via sim-to-real transfer learning.

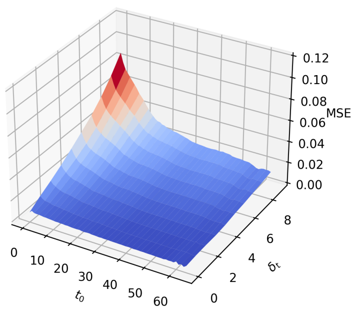
To interpret the performance of the different models in more detail, another representation of the results is shown in Figure 5.4. The model trained only on simulation data shows worsening model performance on starting images taken later from a run as can be seen from the dent pointing upward or *lip* on the right side (regarding t_0) of the plot. All models' performances get better, the later the prediction starts within a run. This points toward the task of predicting



(a) Real



(b) Sim



(c) Transfer

Figure 5.4: MSE depending on start frame of the prediction t_0 and how far the prediction reaches into the future, the frame within the prediction δ_t. The model trained on simulated training examples and the fine-tuned model show a significantly lower maximum MSE (≈ 0.08) than the model only trained on real world examples (≈ 0.12). All models perform worse when beginning to predict closer to the start of the run. All model predictions deteriorate for predictions farther into the future (with higher δ_t).

flow front images getting easier the later a prediction starts within a run. This is reasonable, since the flow front seems to behave more chaotic toward the beginning of a run and later the next step is often just a small, quasi uniform enlargement of the filled area. The fine-tuned model achieves a much smoother MSE landscape overall.

When directly inspecting the outputs of the models, we can see that the output produced by a model trained on real data not only looks inaccurate but also unrealistic (cf. Figures 5.5 to 5.7). These models produce image artifacts that cannot be smoothed out by further predictions, leading to an unrealistic flow front where resin appears in random locations and then expands. This effect rarely occurs in the outputs of other models. When it does occur, it is usually towards the end of a prediction using the real-data only model and sometimes even the fine-tuned model. So, fine-tuning can sometimes worsen model performance, as seen in Figure 5.7, leaving the non-fine-tuned model as the best for that specific run.

Data anomalies During evaluation it was discovered that some experiments were easier to predict than others. This was hinted at by the worsening performance visible as a *lip* on the right side of the t_0 axis Figure 5.4. In Figure 5.8, the mean MSE shows that for all three models experiments 62 to 70 were more challenging to predict, i.e. the error was higher. These experiments also have a longer duration. By manually inspecting the test data it was discovered that there were two different types of data anomalies.

Strong race-tracking As can be seen in Figure 5.8 there were specific runs, that caused a massive decrease in model performance. These were runs that showed strong race-tracking, which does not occur to the same extend in the simulation data. One such case of strong race-tracking can be seen in Figure 5.9.

No perturbations Figure 5.8 also shows that there were specific experiments with a much lower error: 6 to 9. These were runs that had no patches (artificial perturbations) added to them, thus were much easier to predict.

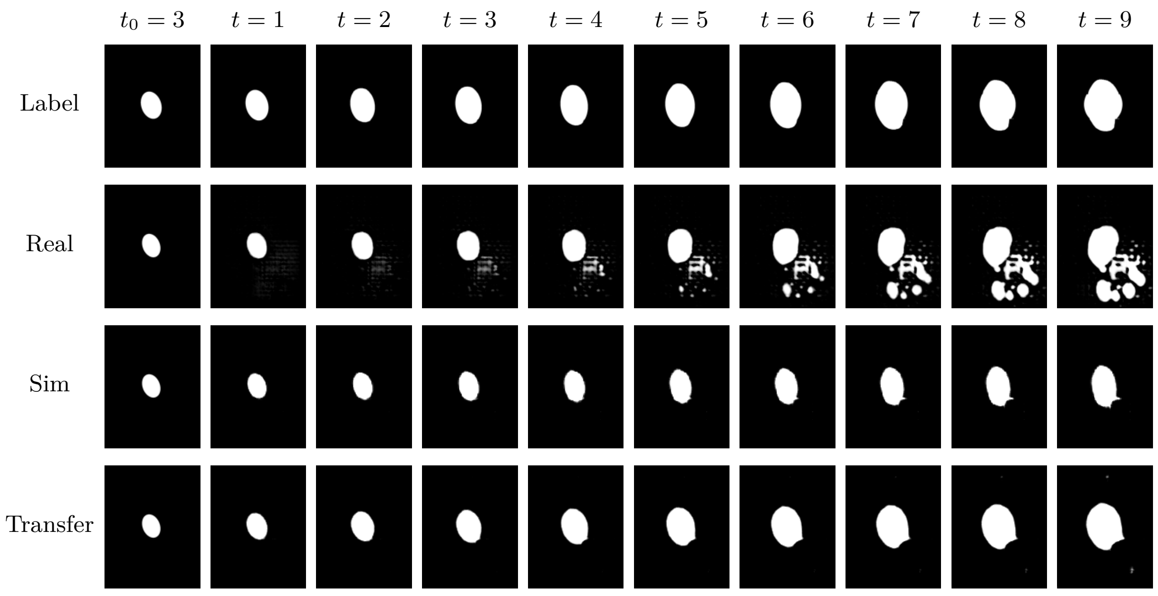


Figure 5.5: Example predictions starting early in a run and their label.

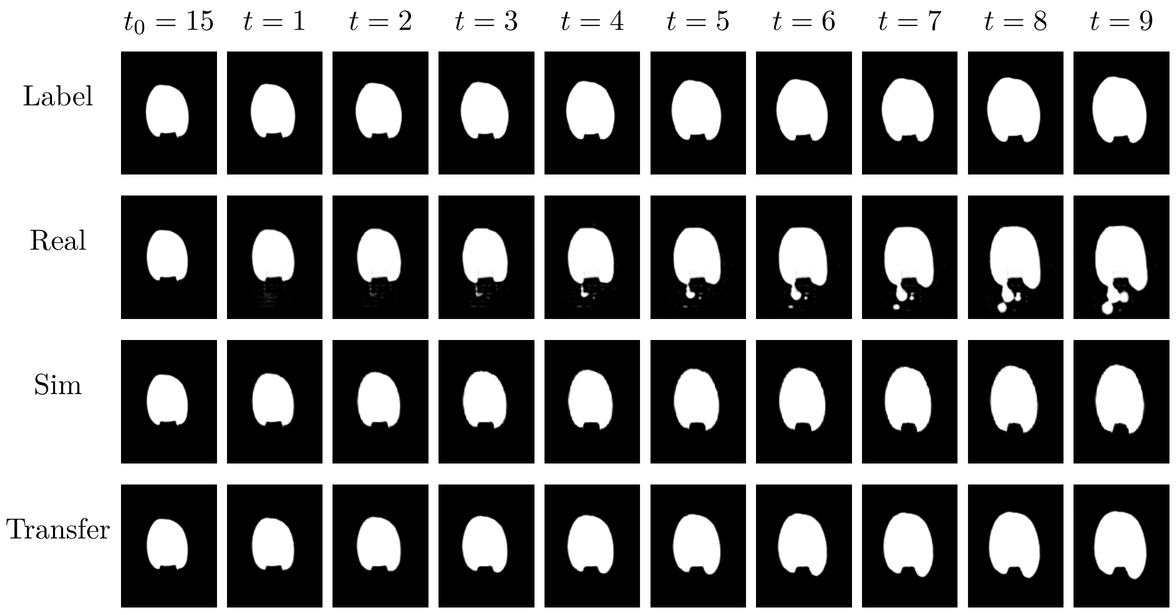


Figure 5.6: Example predictions starting later in a run and their label.

When combining the cases where strong race-tracking occurred with their sequence length, the bumps in the model evaluation can be explained. Those runs happen to also be the only runs that are long enough to have starting frames after about 60 steps, but are much harder to predict than all the other runs, thus causing irregularities.

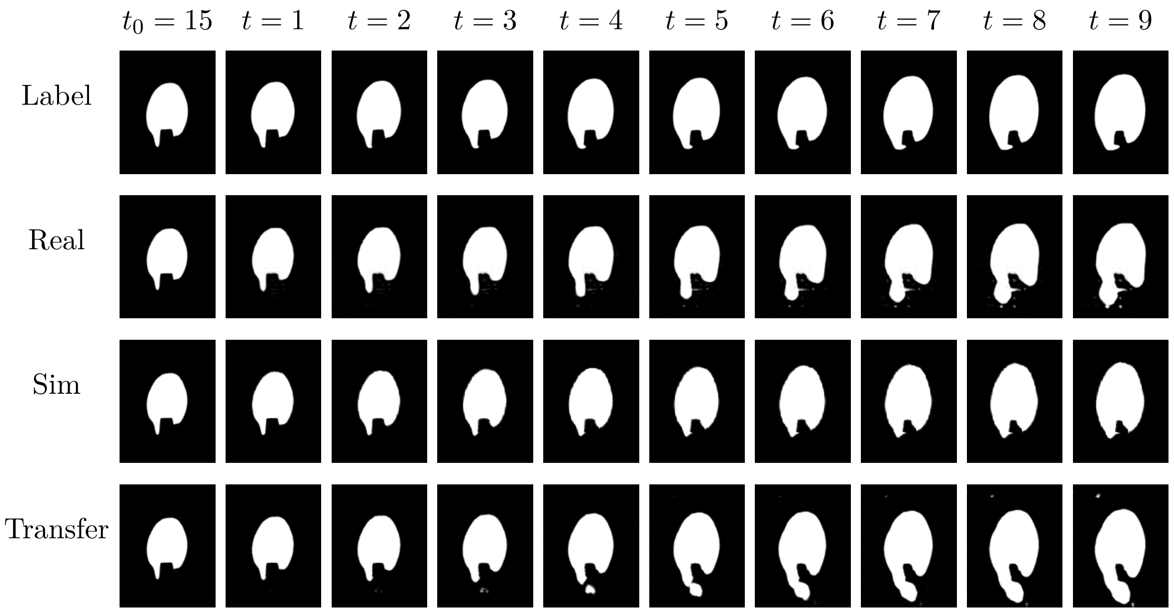


Figure 5.7: Example predictions with worsening transfer learning performance.

5.3 Conclusion

Anomaly Detection in LCM We discussed the detection of dry spots during the RTM process. Dry spots refer to areas of the preform that are not wetted by the liquid, which can lead to weak or incomplete parts. Automated process monitoring based on sensors applied to the mold that do not interfere with the process itself can improve quality assurance. The section also presents different forms of in-process analysis of the RTM process.

We discussed the binary classification of injections during the RTM process. We used several different types of dataset setups to test the performance of different ML algorithms. We used a deconvolutional and convolutional neural network to produce an image of the flow front and then classify the image as wet or dry with a convolutional neural network leading to a binary classification of the injection. The results of this approach are promising, but the network was only trained on simulation data due to the absence of real-world data. We managed to achieve an overall accuracy of 91.68 % when classifying single images of the injection. We also discuss the use of flow front sensors, which only give binary output, and how this does not yield enough information to classify the injections. When using these binary sensors within a sequence prediction task, classifying full runs within the process as of good or bad quality, sim-to-real transfer learning yields better performance, increasing up to 97.37 %.

Process Forecasting The goal of this approach was to predict the behavior of an LCM process both in simulation and reality and to test the ability to improve forecasts on real data with sim-to-real transfer learning. The approach implemented an image-based prediction using a deep CNN that predicts the next flow front state based on the current flow front state and property maps. Real and simulation data were used, and a modified version of a U-Net model was used to predict the flow front behavior. The model was trained to learn the subsequent image of the flow front in a run, and based on that, subsequent images were predicted. The

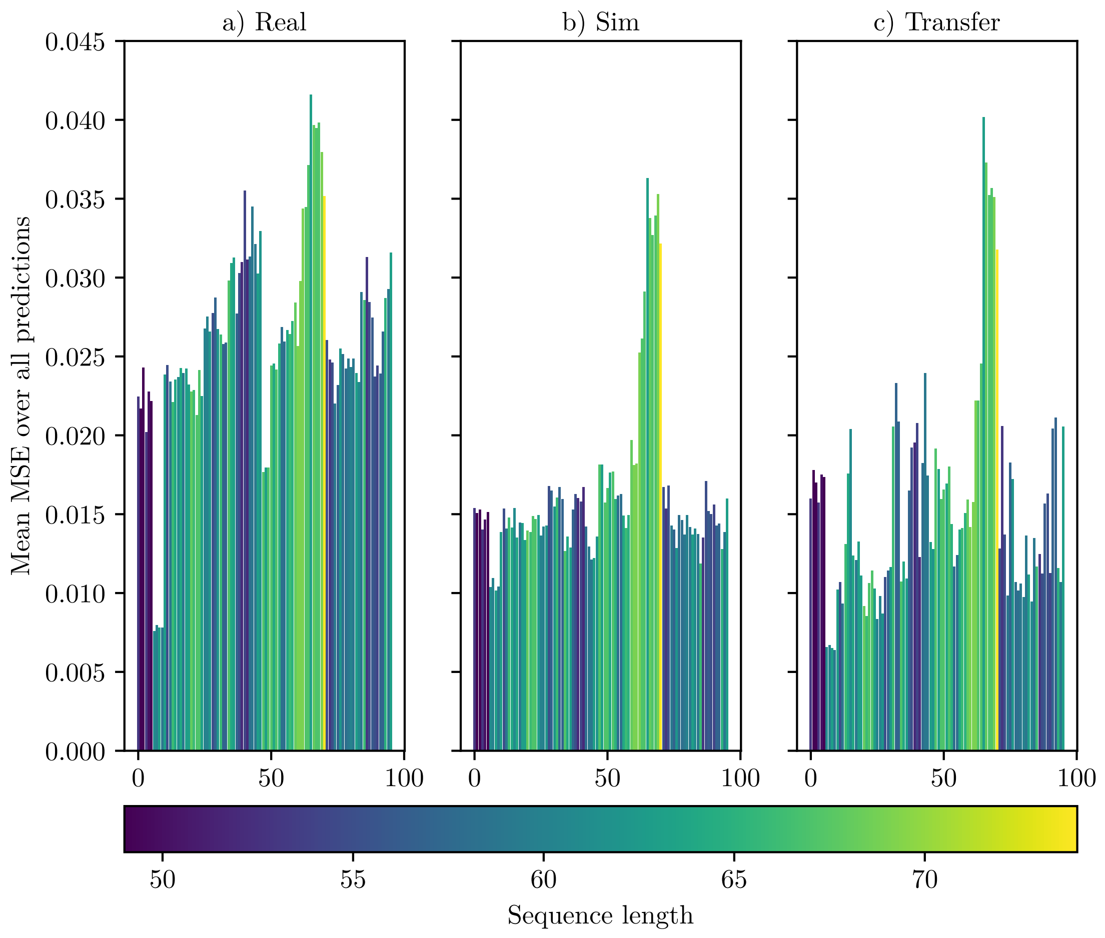


Figure 5.8: Mean MSE per sample and sequence length. The x-axis corresponds to different test data samples, the y-axis to their mean MSE. Color corresponds to the length of that experiment, i.e. the percentage filled of the preform, where a darker color corresponds to a shorter run.

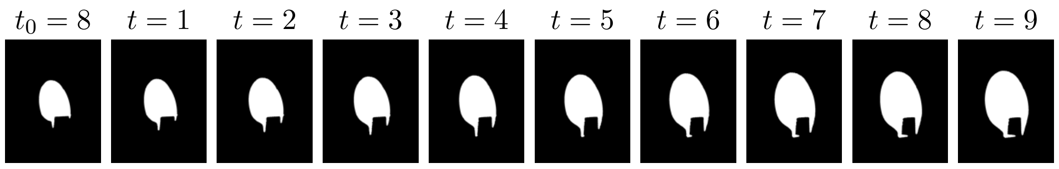


Figure 5.9: A case of strong race-tracking.

model architecture consists of two parts, a downsampling part and an upsampling part. Several steps were taken to improve the model and adapt it to the LCM use case. Overall, this approach offers a supervisory part of the digital twin and allows predicting future failures to steer against them.

In summary, both anomaly detection and process forecasting approaches were evaluated using simulation data and real data, with transfer learning applied successfully to varying degrees. The binary classification of dry spots based solely on flow front data was less effective than using pressure data, which was available in simulations but not in reality. However, the classification of sequences did benefit from sim-to-real transfer learning. Similarly, the process forecasting model showed some improvement when retrained on real data. In conclusion, while sim-to-real learning proved beneficial for certain aspects of these applications, it was not universally effective. Further exploration of this approach will be carried out in the following Chapter 6, which focuses on the digital twin of product properties.

5.4 Related Work

For both major aspects of this chapter, anomaly detection and process forecasting, to cover the relevant related work, two aspects have to be addressed: The technical process and accompanying simulations and the ML models.

Anomaly detection for LCM There have been several publications on in-situ monitoring the RTM process. Pantelelis and Bistekos [2010] present a system on how to detect the curing state but only mention ML-based analyses as future work. Zhang and Pantelelis [2011] use simulated and actual sensor data to detect the curing rate of the process. They use aggregated shallow NNs to achieve this based on two sensors. They further use the model to adapt the heating of the process. Our work, by contrast, focuses on the resin injection and detect dry spots in the flow front from multiple sensors. Leveraging the spatial capabilities inherent to CNNs has never been applied to dry spot detection in the RTM process before.

Unrelated to composite production, deconvolutional layers that increase the spatial dimensions have been used for image enhancement tasks. Xu et al. [2014] used them to reduce blur in images. Shi et al. [2016] approach *superresolution*, the task of scaling up pictures from low to high resolution, with deconvolutional networks. For semantic segmentation, Noh et al. [2015] used a combination of convolutional and deconvolutional networks. They compress the input image with a VGG-16 to afterward decompress the encoding with deconvolutional layers.

Out of these approaches, Shi et al. [2016] comes closest to our task of extracting information from a grid of sensors since the compressed data is known before and not a learned encoding as in Noh et al. [2015]. However, our approach works with more heterogeneous data in that it generates images from sensors rather than improving the resolution of conventional photos. Furthermore, the dimensions of the input are enlarged and not kept the same as in Xu et al. [2014].

Process Forecasting Thuerey et al. [2020] evaluate the accuracy of Deep Learning models, specifically a modernized U-net architecture, for the inference of Reynolds-averaged Navier-Stokes (RANS) solutions. The researchers assess a large number of trained NNs to determine the accuracy of pressure and velocity distributions within a simulation for airfoil flow, which can be seen as a similar simulation to LCM simulations. While this study focuses on RANS solutions, the NN architecture and learning setup are applicable to a wide range of partial differential equation boundary value problems on cartesian grids, which we face with the LCM problem. They use simulation data only, while we also apply our approach on real data.

Ronneberger et al. [2015] present a network and training strategy that utilizes data augmentation to efficiently use the available annotated training samples for successful deep network training. The network is also fast and was originally used for image segmentation but the architecture has proven to be performing on many use cases. We adopted it for process forecasting, as proposed by Thuerey et al. [2020].

Excursus: Graph Neural Networks for Process Forecasting
To address upcoming NN architectures for simulation surrogates, we tested Graph Neural Networks. These networks can handle data correlated in graphs, and our FEM simulations are based on meshes, which are a type of graph. That led to a longer experimentation time with applying them to the mesh-based simulations for the LCM process, to forecast the process or to even create a surrogate for the numerical simulation that is faster than the original. Lodes [2021] directly applied them to the LCM use case in his master's thesis, but it was not a resounding success, since the amount of mesh data, the number of nodes was to large to be efficiently processed by the network, plus there were to many connections of which not all were relevant for message passing. That led to the problem that information was not passing far enough. Later, Pfaff et al. [2021] described how to apply Graph Neural Networks indirectly on mesh-based simulations, by creating a *latent graph*, that must not be directly correlated to the original mesh structure, and how to predict future behavior in these types of simulations. That solved the problem of diminishing message passing though the graph. In the following, we continued applying the introduces principles on our case, but found two other major limitations that eventually led us to stop continuing that path of research. First, we found that these graph networks could only be applied to smaller simulations, so we created a smaller mesh with around 1000 nodes instead of $\approx 20,000$ nodes, that we used before. This downsizing leads to a noticeable loss of accuracy in the simulation, but we decided to test it. Dimension-wise, it is comparable to case study B, which is also a considerable downsizing, see Subsection 4.1.2. And second, the applicability to the dataset from reality was not clear, since there is no mesh in the real data, consisting of images and some meta data only. To summarize, we tried to apply graph neural networks on our case, but the lack of accuracy and the ability to apply it to real data let us stop this endeavor.

Demirci et al. [1997] utilized a numerical simulation to train a NN to predict the flow front position in the next time step based on the current flow front position and injection pressure. They conducted experiments using a non-rectangular mold with two inlet gates. They defined an optimal flow front as the control target, which not only predicts future flow but also includes the control problem. This approach is further discussed in Chapter 7 where their methodology is revisited. However, due to technical limitations at the time, their study had a comparatively coarse measurement of the flow front and used a shallow network architecture. In contrast, in this study, we were able to significantly increase the measurement resolution, network depth, and number of training samples.

Chapter Summary and Outlook
To summarize, two major topics were presented in this chapter, Anomaly Detection and Process Forecasting in LCM. Anomaly detection involves the detection of dry spots during this process and the binary classification of injections during the same process. To be able to detect anomalies, a method to reproduce the form of the flow front from sensory data was presented. This study demonstrates the importance of incorporating realistic material properties, flow

characteristics, and data processing techniques in both real and simulated datasets for training neural networks to predict fluid flow in composite materials. Process forecasting involves predicting the behavior of an LCM process both in simulation and reality and testing the ability to improve forecasts on real data with sim-to-real transfer learning. The chapter discusses the success of his transfer learning on both simulation and real data, highlighting that it was useful in certain aspects, but not for all.

Chapter 6

Digital Twin of Product Properties

Summary. The quality of impregnation and the degree of FVC play vital roles in determining the final part quality in LCM processes. In this study, we demonstrate a method for concurrently learning three primary textile properties (FVC and permeability in both x and y directions), which are represented as a three-dimensional map and serve as a *digital twin of the product properties*. This approach is based on both data from simulation and image data collected during permeameter flow experiments, and we compare the performance of CNNs, ConvLSTM models, and Transformer models. With those two different data sources available, we illustrate how simulation-to-real transfer learning can enhance a digital twin in FRP manufacturing, as opposed to simulation-only models and models relying on sparse real data. The best overall metrics on real data achieved are: IoU 0.5031 and pixelwise accuracy 95.929 %, which are obtained by Transformer models pre-trained on simulation data.

Publication. The concepts and results presented in this chapter are published in [Stieber et al., 2021a, 2023b].

6.1 Property Prediction: FVC and Permeability in Principal Directions

To recap from Subsection 2.1.1, LCM techniques are among the most renowned and cost-effective for manufacturing FRP components and RTM is suitable for medium production volumes. As discussed in Subsection 4.2.3, the *permeability* properties of the preform, which describe how well a porous medium transmits fluids, greatly influence the flow front dynamics. In FRP, preform permeability is primarily determined by (i) FVC and (ii) preform layout. Localized variations in preform permeability can occur at locations with varying wall thickness or curved sections of the component (compression or even folding of the preform on the inner surface of the curvature). They can also result from manual handling of the fibrous structure or material defects such as missing or misplaced fiber bundles (e.g., fringing of fiber bundles, breakage of filaments). For a single type of textile, permeability can vary by up to 20 %, according to Tifkitsis and Skordos [2020] and May et al. [2019]. Local permeability differences may lead to suboptimal impregnation quality or even dry areas, negatively impacting the mechanical performance of the

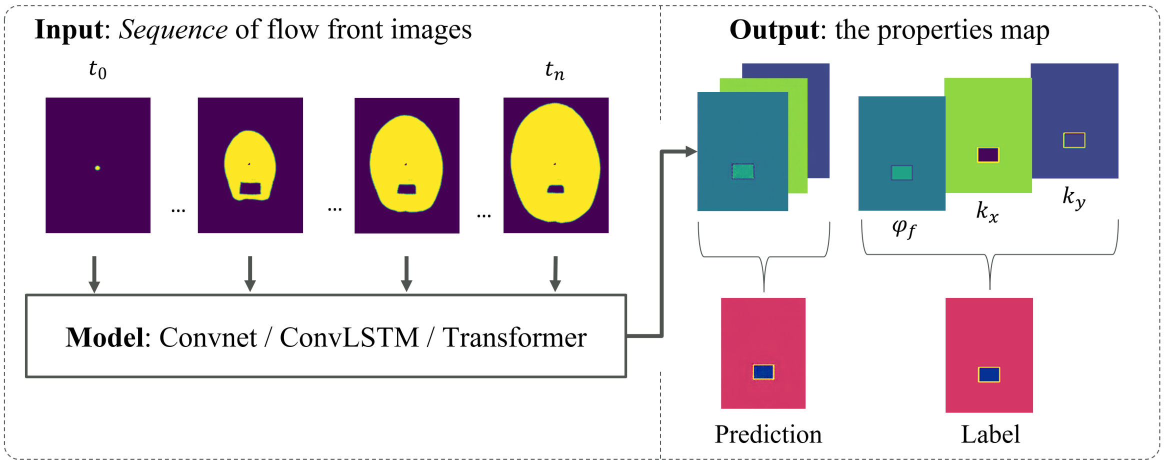

Input: *Sequence* of flow front images
t_0
t_n
...
Model: Convnet / ConvLSTM / Transformer
Output: the properties map
φ_f
k_x
k_y
Prediction
Label

Figure 6.1: Overview of the sequence-to-instance learning task: a sequence of flow front images is mapped to an image that contains the permeability values of the material, i.e. the permeability map (from [Stieber et al., 2023b]).

final composite part. These variations also influence the temporal progression of the flow front. As the changes in a specific preform are often unknown before or during an industrial RTM process, having a *map of the permeability and φ_f values would be invaluable for identifying potential issues as early as possible.*

We propose using ML models that, given an observed *sequence of flow front images* (cf. Subsection 5.1.2) during an injection procedure in a permeameter, predict *changes* in the permeability and φ_f, resulting in a *sequence-to-instance task* (cf. Figure 6.1). The training data is collected from two sources, both from Case study A (cf. Subsection 4.1.1): (1) flow experiment data from a permeameter, and (2) flow simulation data from PAM-RTM. This task is part of a broader pipeline that ultimately aims to predict a textile property map from in-mold sensory data. For this, an intermediate step that converts *sensors* to *flow front images* is required, employing techniques such as transposed convolutions [Stieber et al., 2021b], generative models (cf. excursus in Subsection 5.1.1), or analytical methods [Achzet et al., 2022].

In summary, the *contributions* are as follows:

- For this sequence-to-instance task in an engineering context, we compare several ML models (see Table 6.1 and Table 6.2).
- We demonstrate the effectiveness of sim-to-real transfer learning (cf. Subsection 5.1.2) in low data regimes by using transfer learning to adapt a model trained in the simulation domain to real data (see Table 6.3).

This method can be considered a component of the process' digital twin [Stieber et al., 2020] (cf. Section 3.3), as it provides valuable additional information in digital form about the future product from sensory data without any post-process testing. It is meant for process workers or engineers to determine the quality of the product. A component can be deemed acceptable if it does not exhibit excessive fluctuations in permeability and φ_f. The modified textile property maps result in altered flow fronts that need to be rediscovered by the ML models. These models take a sequence of flow front images as input, as shown in Figure 6.1.

For isotropic material, an ML model predicting φ_f would enable an analytical computation of in-plane permeability, assuming that a corresponding equation is known for a specific material.

In an anisotropic material, restricting an ML model to predict only φ_f is not sufficient to deduce information about in-plane permeability and their principal directions. However, knowing the principal flow directions of the fabric is crucial for reasoning about fluid flow, particularly when introducing patches with unknown fabric orientation. Therefore, considering in-plane anisotropy was the primary motivation for developing a model that predicts all three properties - φ_f, k_x, and k_y - from sequences of flow front images.

In a first work [Stieber et al., 2021a], we predicted the k_x permeability as a feasibility check of this approach for isotropic textiles. In another subsequent work [Stieber et al., 2023b], we predict all three major properties of the fluid flow through the textile:

- φ_f - volume-averaged fiber volume content
- k_x - permeability in x direction
- k_y - permeability in y direction

For this work, the training, test, and validation sets were fixed across all experiments, as described in Subsection 4.3.3.

6.1.1 Approach: Sim-to-Real Transfer Learning and Models

Using sim-to-real transfer learning (cf. Subsection 3.1.2) for our models, we first train them on a larger dataset from simulation only (SimAniso and SimAnisoChannel). Next, the trained networks are retrained on a smaller dataset containing data from real experiments, but with a reduced learning rate. Decreasing the learning rate in the second training step helps prevent the network weights from being altered too drastically, which could lead to a drop in performance.

In general, this approach allows the use of larger and more capable models for datasets that would typically be too small, as described in Section 6.5. In this transfer learning approach, the model trained on simulated data serves as an initialization point or warm start for the model trained on real-world data. This assists the model trained on real-world data in learning more quickly and achieving better performance.

We train all models end-to-end, from the input sequence of flow front images to the property map (see Figure 6.2). We use several NNs that can handle this 3-dimensional sequence-to-instance task. Our baseline model is a CNN with four 2D convolutional layers. We subsample the input sequence to get 100 single-channel images of the injection process time steps, as described in Subsection 4.5.4. We treat these time steps as separate channels for the first convolutional layer, since 2D convolutions can operate on any number of channels.

The second model is based on the Transformer mechanism (cf. Subsection 3.1.1.5), which works on one-dimensional embeddings. Therefore, we need to convert the input image sequence into an embedding sequence. Opposed to ViT (cf. Subsection 3.1.1.6), that cuts images in little patches to then use the transformer mechanism, we use a fully convolutional encoder to create feature vectors for each image and train it end-to-end in the sequence-to-instance pipeline. The encoder part merges the sequence length and the batch dimension, so that the convolutions work on each image individually. After the encoder, we restore the dimensions to use the sequence capabilities of a Transformer (see Figure 6.2). We use the Transformer output vector to create the property map with 2D transposed convolutions that are followed by some convolutional layers. This helps gaining sharpness in the images, an approach we also employed in FlowFrontNet (cf. Subsection 5.1.1). In summary, the model has an encoder — Transformer — decoder structure.

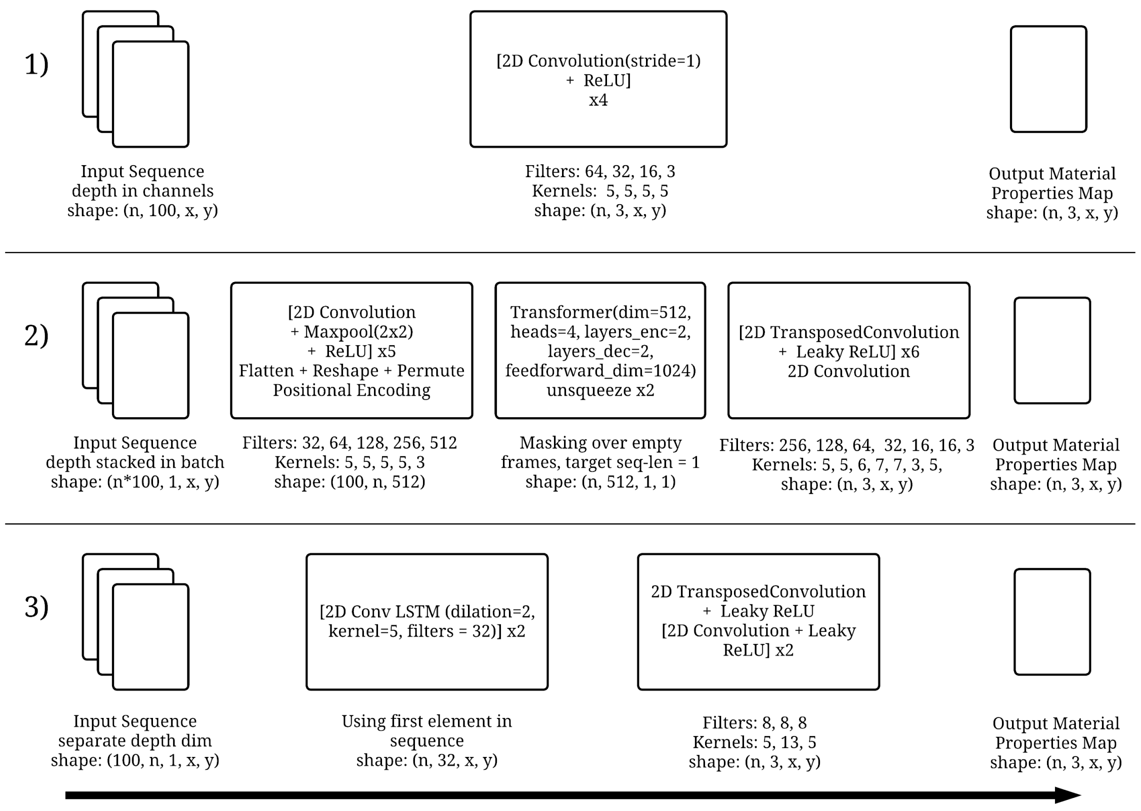


Figure 6.2: Overview of the three proposed models: 1) A simple fully 2D convolutional network, 2) a Transformer-based approach with a 2D convolutional encoder and decoder and 3) a 2D convolutional LSTM network.

The third approach focuses on the temporal aspect by using a ConvLSTM architecture [Shi et al., 2015]. To recall Subsection 3.1.1.4, ConvLSTMs work on sequences of two-dimensional matrices, instead of one-dimensional vectors as regular LSTMs [Hochreiter and Schmidhuber, 1997], which makes them suitable for our task.

We improved all models compared to Stieber et al. [2021a] by using leaky ReLU instead of regular ReLU activation functions in the image reconstruction part and by adding batch normalization to the network. These changes led to faster and more stable training.

6.2 Evaluation

Motivated by Stieber et al. [2021a], which used networks trained on simulated data for inference on real data, we explored the following research questions:

1. How do the network architectures that we proposed in Subsection 6.1.1 perform relative to each other?
2. Can the task be learned by real-world samples only and what is the benefit of using simulation data?

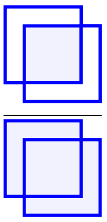
3. How much improvement can we achieve by retraining NNs with real data that were pre-trained on simulated data?
4. How does the simulation fidelity affect the output of the trained models?

6.2.1 Metrics

To assess the performance of the output images beyond an "expert eye", we first had to identify a suitable set of metrics. Identifying a statistic that accurately represents observed performance was challenging because mean *pixelwise accuracy* often favored blurry predictions. Given that a considerable portion of the property maps had nearly identical base properties, accuracy alone was not sufficient. As a result, we allowed for an ε-tolerance to still classify certain pixels as correct. We also considered measuring the deviations only in a selected area where the patch is. When examining actual predictions of the models that occasionally were far off not just in the region of the patch but in a far greater area as shown in Figure 6.5, we decided to stay with the more broader metric of pixelwise accuracy in the complete image.

Additionally, we used the IoU metric, also known as the Jaccard index, which focuses on the size and position of the introduced patch, representing the observed variation in properties. The IoU metric is largely used in other CV-related domains, such as semantic segmentation [Yu et al., 2018] or object detection [Zhang et al., 2021] where it is necessary to measure the correct size and positioning of certain shapes (mostly boxes) within an image. We established thresholds for each property based on the label. For example, in the RealPermLarge dataset, the lower and upper bounds for φ_f would be 0.38 and 0.41. As described earlier, φ_f was intended to be 0.4, but manual handling of the textile resulted in the measured range, shown in Figure 4.6 (a). These values were derived by examining the entire dataset for one property, binning it into 100 bins between 0 and 1, and selecting the bins with the most entries to represent the majority of the textile. Consequently, for φ_f in this case, the border values are 0.38 and 0.41, (cf. Figure 4.6). Next, we created a binary image with pixels either falling within or outside the desired window of values. Finally, we calculated the IoU according to Equation (6.1) using this binary image and a binary version of the label.

$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \qquad (6.1)$$



The IoU metric is calculated independently for all three different properties, and we also compute the mean of the three values. For accuracy, we calculate the three independent values for each property, as well as a combined value that takes all channels into account.

To summarize, the pixelwise accuracy metric measures if the correct value was predicted by the model for each pixel and the IoU focuses on the positioning and size of the introduced patch(es) as an additional metric.

6.2.2 Results - Value of Sim-to-Real Transfer Learning

In the following the proposed research questions regarding the online prediction of material properties are answered.

Table 6.1: Results: Model comparison for the simulation data test set, focusing on three properties: k_x, k_y, and φ_f. Including training time for one epoch of 10,000 samples on 4 GPUs.

Model	Acc. Tot.	IoU (Mean)	Training Time (One Epoch)	Parameters
CNN	85.802	0.1406	**0:24 m**	$225,315$
Transformer	92.191	0.4068	1:45 m	$17,293,691$
ConvLSTM	**96.893**	**0.6943**	27:30 m	$328,491$

Table 6.2: Results: Networks pre-trained on simulation networks with real data as test set and pre-trained networks with retraining with 10 real samples - Model comparison for 3 properties: k_x, k_y, φ_f

Model	Acc. Tot.	IoU (Mean)
CNN (Real Data Test Set)	64.901	0.1985
CNN (Transfer)	81.964	0.2262
Transformer (Real Data Test Set)	93.404	0.3321
Transformer (Transfer)	**95.331**	**0.3826**
ConvLSTM (Real Data Test Set)	91.783	0.3375
ConvLSTM (Transfer)	94.148	0.3333

How do the network architectures that we proposed in Subsection 6.1.1 perform relative to each other? Addressing the first research question, we present the metrics from all three proposed architectures on the SimAnisoChannel dataset in Table 6.1. The results indicate that the ConvLSTM achieves the best performance on the simulation test set in terms of both accuracy and IoU, although it has the longest training duration.

Can the task be learned by real-world samples only and what is the benefit of using simulation data? Table 6.2 demonstrates that the Transformer surpasses the ConvLSTM in performance when using real data as a test set and when re-training with only 10 real samples, addressing research question 2. The samples were taken from the RealPermSmall and RealPermLarge datasets (cf. Section 4.3) to showcase the potential of using a limited number of samples. During the retraining step, no weights were frozen and the learning rate remained constant at 1×10^{-4}. It's also worth noting that the Transformer trains much faster than the ConvLSTM, enabling quicker results during development and evaluation (see Table 6.1).

How much improvement can we achieve by retraining NNs with real data that were pre-trained on simulated data? We further examined the extent to which the best model, the fine-tuned Transformer, benefits from additional real data for training. This question is of particular interest since a common concern before implementing data-driven models for specialized use cases is: "How much real data will be needed?" To address this, we used our augmented training dataset of 288 injections and trained the Transformer model from scratch

Table 6.3: Overview: Effects of sim-to-real pre-training for the Transformer

Model	Acc. Tot. [%]	IoU (Mean)
Real data only: 240 samples	80.323	0.2143
Sim data only + No re-training	93.404	0.3321
Transfer: 10 real Samples	95.331	0.3826
Transfer: 240 real samples	**95.929**	**0.5031**

and from a checkpoint based on training with simulated data. An overview of the benefits from pre-training can be found in Table 6.3, with results also shown in Figure 6.6.

A possible explanation for these results is that when the Transformer is trained from scratch using only real data, it can only predict the presence of dry spots, disregarding the offset and orientation of the patch. However, it can differentiate between dry spot locations in the y direction. The sample size does not seem to significantly impact overall performance, as the metrics are erratic and do not surpass a certain level. In contrast, starting the training with a sim-data-only checkpoint results in much better performance, with more accurate predictions of orientation, location, and extent, which is reflected in the metrics. Additionally, performance increases almost linearly with the number of real data samples used.

Considering that the Transformer has many parameters and may not perform well on smaller datasets like our real-world dataset, we also use the CNN with fewer parameters as a second baseline for real-data-only training. In Figure 6.6, we compare the CNN and the Transformer in terms of real-data-only learning and transfer learning capabilities. The figure indicates that while the smaller CNN outperforms the larger Transformer in terms of accuracy, they are equal in IoU. However, when leveraging simulation data through pre-training, the Transformer greatly outperforms the CNN.

The best overall metrics are achieved with a dataset size of 240, which already includes data augmentation (cf. Subsection 4.3.3), boiling down to just 80 non-augmented samples, with IoU reaching a good value of 0.5031 (an increase of 0.12 from 10 samples) and accuracy at 95.929 % (an increase of 0.6 % from 10 samples). This answers research question 3, and examples of well-defined property maps can be seen in Figure 6.3 and Figure 6.4, the latter showing less perfect outcomes. Although the latter occurs less frequently than the former, we wanted to include imperfect outcomes in the evaluation as well.

How does the simulation fidelity affect the output of the trained models (SA vs SAC)? To address research question 4 on the impact of simulation quality on the overall performance of the sim-to-real Model, we tested the best-performing model, the Transformer, on a lower-quality simulation dataset. As a reminder, Table 4.2 shows all simulation-based datasets, with SimAniso and SimAnisoChannel being the latest and closer to reality. In the anisotropy preliminary section (4.4.1), we explain the importance of anisotropic textile features and the introduction of channels to create a more realistic simulation. We tested the Transformer on the SimAniso dataset without channels for race-tracking to assess its performance on a less realistic dataset. The comparison in Table 6.4 shows that the models pre-trained on the more elaborate SimAnisoChannel (SAC) data perform better than their SimAniso (SA) counterparts in terms of accuracy and IoU. This difference is also noticeable in Figure 6.5.

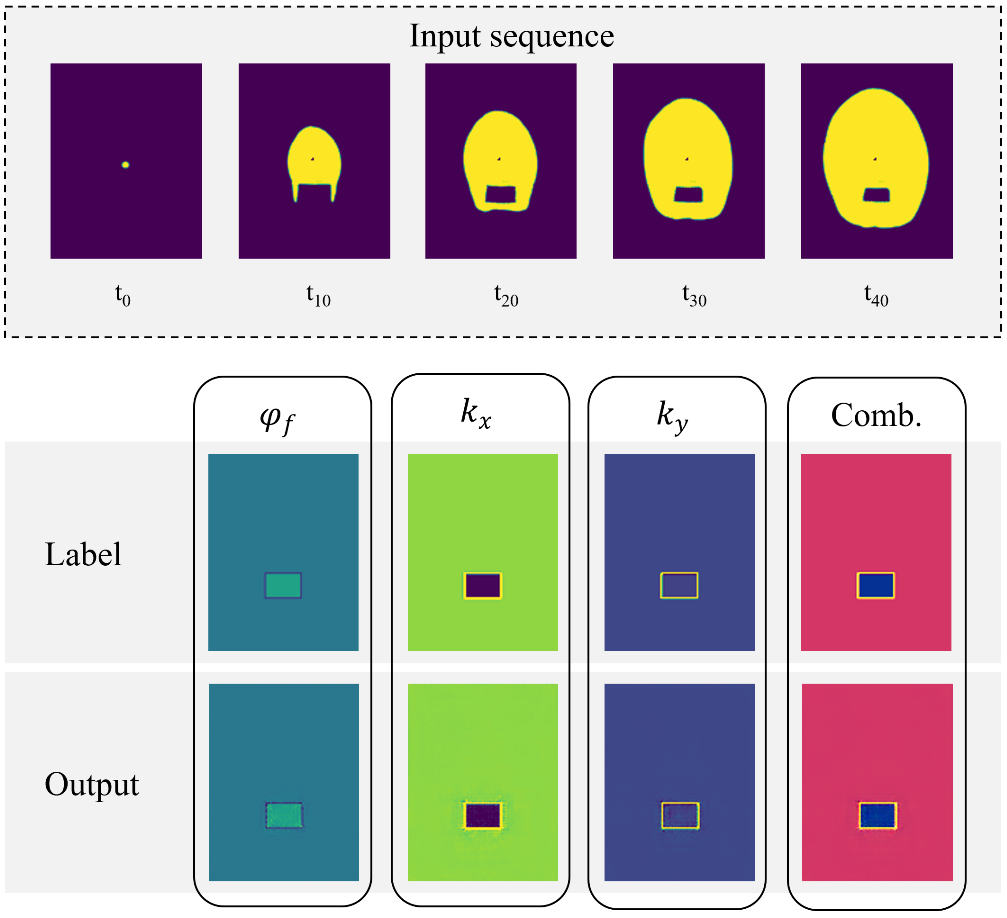


Figure 6.3: Great example: Input sequence (shortened), Labels of all three properties and combined, output of the transformer for each property and combined

Both transfer learning models (T/SA and T/SAC) show better results than models trained solely on simulation data (NT/SA and NT/SAC). However, there are notable differences in performance when examining run “R1”: The Transfer SA (T/SA) model generates a misaligned patch in comparison to the Transfer SAC (T/SAC) model. When looking at models trained exclusively on simulation data without exposure to real data during training, the differences become even more pronounced. In run “R2”, the No Transfer SA (NT/SA) model creates strong artifacts, while the No T/SAC model displays lighter artifacts and nearly accurately predicts the patch size and position. These dataset excerpts support the claim made by the metrics tha t improved simulation datasets facilitate the transfer learning process from simulation to reality.

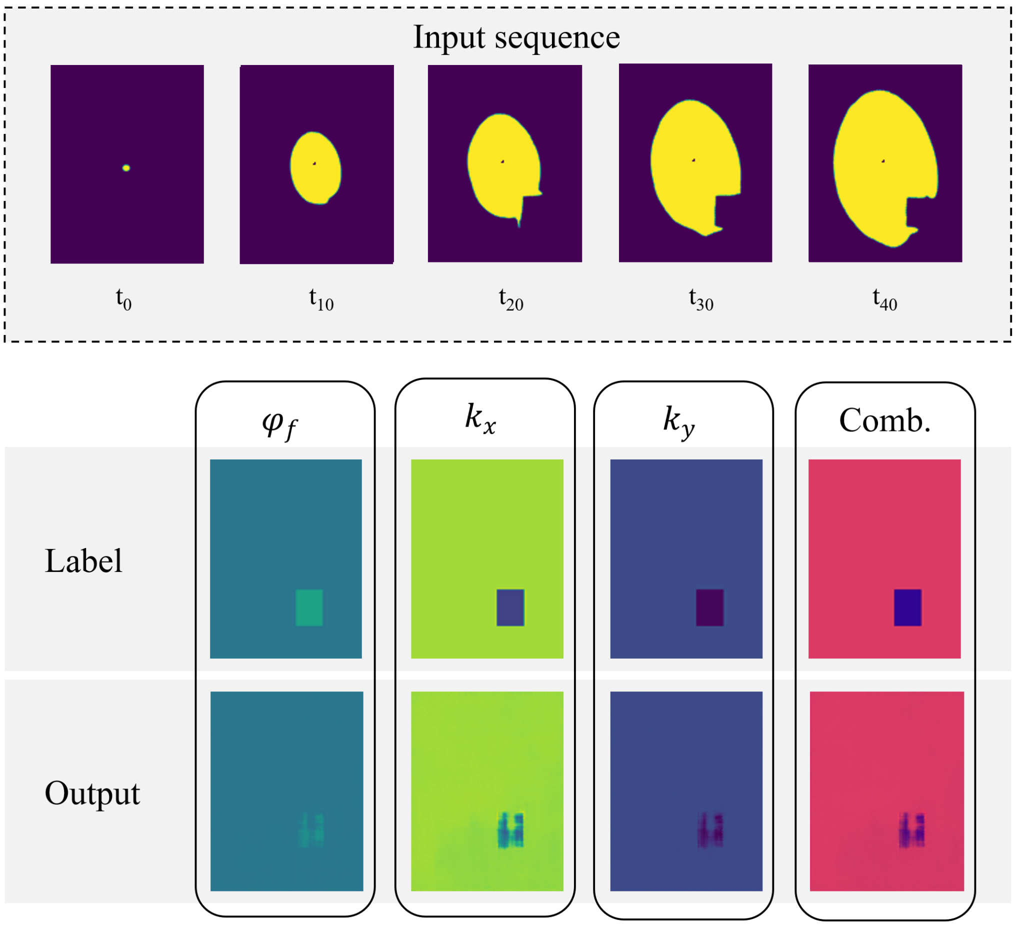


Figure 6.4: Underwhelming example in comparison to Figure 6.3, here the borders of the patches are not as well defined.

Table 6.4: Transformer model pre-trained on two different datasets of different levels of simulation fidelity: SA and SAC

Model	Dataset	Acc. Tot.	IoU (Mean)
Transformer	SA	92.079	0.1648
Transformer (Transfer on 10 samples)	SA	93.974	0.3036
Transformer	SAC	93.404	0.3321
Transformer (Transfer on 10 samples)	SAC	**95.331**	**0.3826**

6.3 Discussion and Future Work

In-mold sensor data would be necessary to apply the results of this paper to an industrial RTM setting [Stieber et al., 2020]. A modular approach could first take sensor data and map them to flow front images and then feed those outputs directly into the models presented here. Alternatively, the mapping from sensor data to permeability deviation maps could be learned

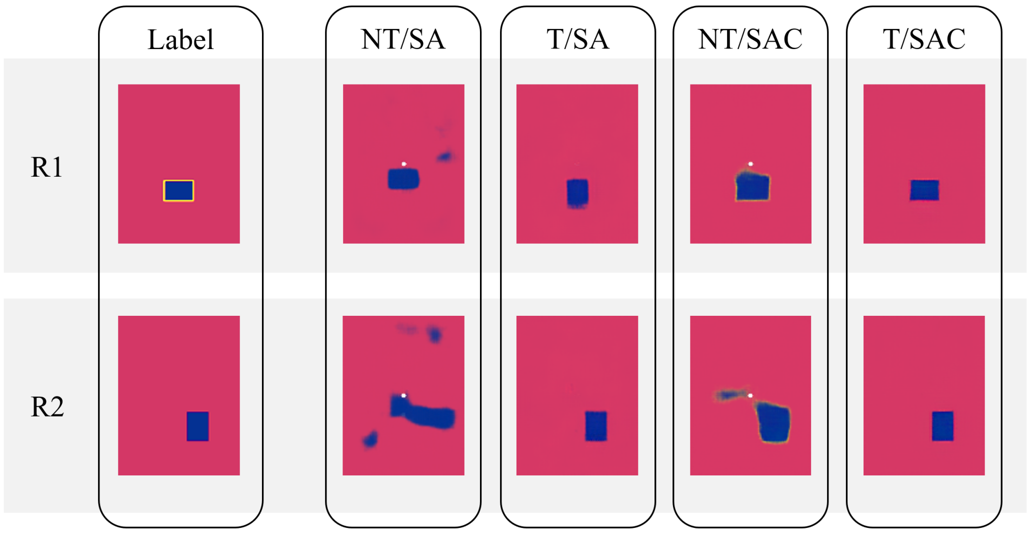


Figure 6.5: Label and output from different datasets from simulation: SA has no race-tracking channels, SAC has channels. Matches Table 6.4

end-to-end, given a small dataset of respective real data. For Vacuum Infusion, the images of the flow front coming from a camera on top of the transparent upper mold would suffice with this approach.

6.4 Conclusion

We set out to investigate how ML can be used to infer material properties during LCM processes. Generating huge amounts of real data for LCM processes is time-consuming, expensive, and therefore undesirable. Instead, we chose to leverage sim-to-real learning, by generating large amounts of simulated RTM data, and only a small amount of real data. We trained three different models, a simple CNN, a ConvLSTM, and a novel Encoder-Transformer-Decoder and evaluated their performance. Our research shows that leveraging simulated data for pre-training can greatly improve performance compared to a model trained solely on a small amount of real data. Further, we investigated how little data is truly needed to have a real-world improvement. We found that even as little as 10 samples can be enough if the data is generated carefully with respect to expert knowledge. To add a little more detail, using 10 real samples with a network that was pre-trained with a well-crafted simulation, we gain 25 percent points in accuracy on a non-pre-trained network. In terms of IoU, we gain 0.20 points. The difference between two datasets based on better or worse simulations is 1.4 percent points in accuracy, from 93.9 to 95.3 and 0.08 in IoU, from 0.30 to 0.38.

The IoU is a delicate metric because the numerator can decrease rapidly due to a shift in the prediction relative to the label in both the x- and y- directions, and the denominator grows with the size of the predicted patch. Therefore, even seemingly small improvements are a significant or “noticeable” improvement in performance and visualization

That shows that only small amounts of data can be used when a simulation for pre-training is available and better simulations leverage the results even further. Additionally, we showed

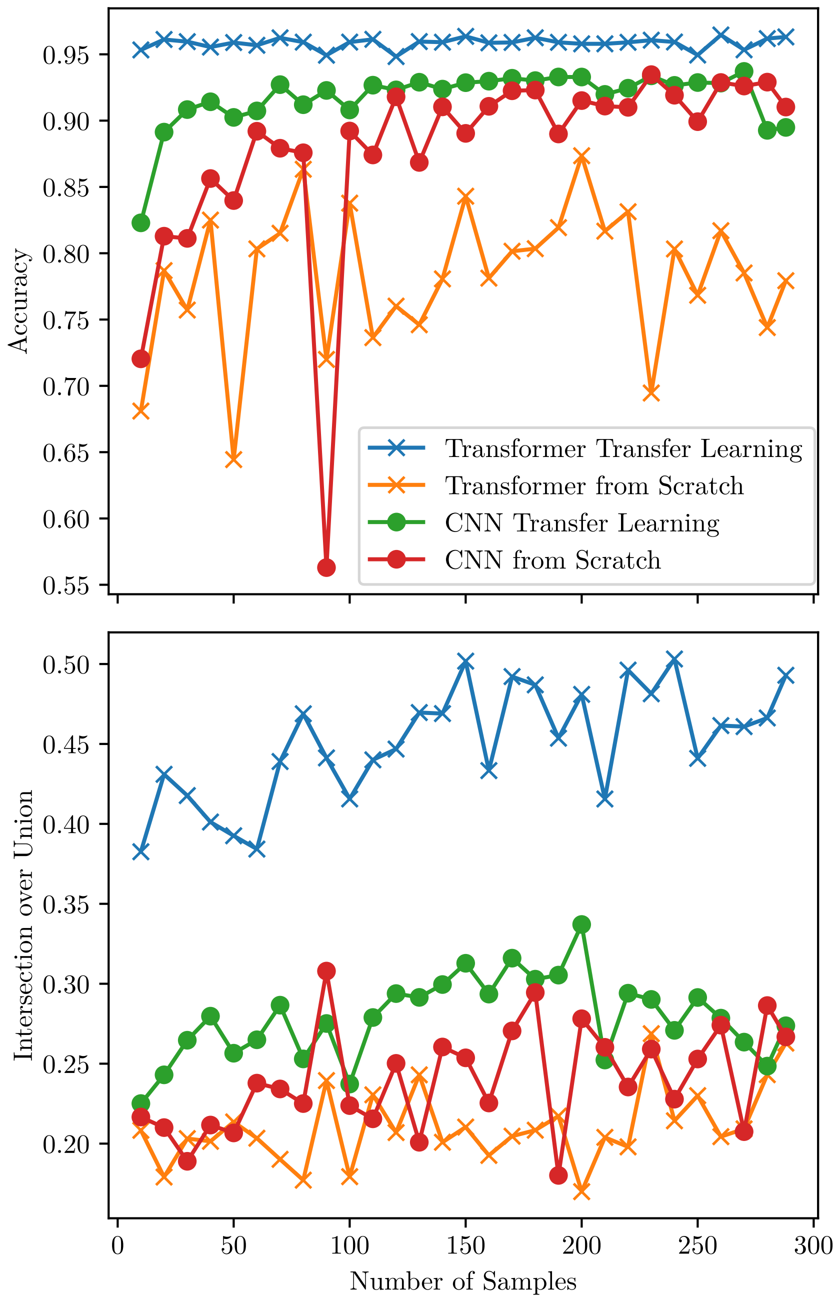


Figure 6.6: Accuracy and IoU over different dataset sizes (from RealPermLarge and RealPermLarge) on with CNN and Transformer (Transfer from models pretrained on SimAnisoChannel)

how the quality of the simulated data can impact model performance. Our results suggest that keeping the domain shift between real and simulated as small as possible can help to improve the quality of the predictions.

6.5 Related Work

The composites processing literature has taken an interest in online measurements of permeability or its variation. The issue of in-situ monitoring the filling status of RTM processes using lineal sensors is taken into consideration by Tifkitsis and Skordos [2020]. They use Kriging models to forecast the arrival times of the flow front at specific sensors. PAM-RTM is utilized to generate the RTM simulations that are needed to fit these substitute models. However, we take into account the entire spatial expansion of the flow front (i.e., a high-dimensional prediction) while they just take into account two measurement lines across a plate. By using Darcy's law (cf. Subsection 2.2.5) to calculate it, Wei et al. [2016] estimate global and local permeability during the injection with pressure and flow front sensors. A Regularising Ensemble Kalman filter Algorithm (RENKA) was demonstrated by Matveev et al. [2021] as a means of detecting permeability changes throughout an RTM process. To reduce the computational complexity of pure Bayesian inversion, they introduce this filtering technique. However, the algorithm is less usable than a fully data-driven ML approach because it requires nontrivial mathematical steps and must be modified for new component designs. CNNs were utilized by González and Fernández-León [2020] to monitor changes in the flow front caused by pressure sensors to detect permeability changes. However, they are only able to identify discrete, rectangular changes in permeability by using the data of all recorded runs. Our method is more flexible since it creates the full permeability map of the preform rather than just identifying specific elements of the patch, like its length, width, or center point. Moreover, the flow fronts in each of the aforementioned works differ because they all employ a lineal injection gate rather than a central one. CNNs are used by Caglar et al. [2022] to measure the permeability of textiles at the microscopic level.

All the above works employ rectilinear flow, whereas our work employs radial flow. From the manufacturing perspective, this a more realistic case since it is two dimensional instead of one dimensional and thus is applicable to a broader range of processes.

Chapter Summary and Outlook
In this chapter, we explored the application of Deep Learning models to predict the textile properties within LCM processes using a sim-to-real transfer learning approach. We investigated three neural network architectures: a CNN, a Transformer, and a ConvLSTM. The models were first trained on large simulated datasets, followed by fine-tuning on smaller real-world datasets. The ConvLSTM outperformed the other models on simulation data only, while the Transformer excelled when tested on real data and for transfer learning and had a faster training time. We examined the impact of the amount of real data on model performance, and the results indicated that leveraging simulation data through pre-training significantly improved the model's ability to predict textile properties. The best overall metrics were achieved with a dataset size of 240 samples. Furthermore, we compared the CNN and the Transformer regarding real-data-only learning and transfer learning capabilities. Although the CNN achieved higher accuracy due to its smaller parameter count, both models performed comparably in terms of IoU. However,

the Transformer clearly outperformed the CNN when using simulation data for pre-training. Finally, we addressed the influence of simulation quality on model performance by testing the Transformer model on a lower-quality simulation dataset. The results showed that models pre-trained on the more realistic dataset (SimAnisoChannel) performed better in terms of accuracy and IoU than their counterparts trained on the less realistic dataset (SimAniso). This finding emphasized the importance of high-quality simulation data for facilitating the transfer learning process from simulation to reality. This approach could be expanded to predict fiber orientations as well, given the data in both simulation and more complicated in reality. This would be another interesting quality feature that could be predicted from the way the fluid flows through the textile.

Chapter 7

Control of the Process

Summary. RTM (cf. Subsection 2.1.1) is a composite manufacturing process that uses a liquid polymer matrix to create complex-shaped parts, that yields several challenges. One of the main challenges is ensuring that the liquid polymer matrix is properly distributed throughout the composite material during the molding process. If the matrix is not evenly distributed, the resulting part may have weak or inconsistent properties. Detecting and visualizing these potential weaknesses in form of a digital twin was the main topic of Chapters 5 and 6. To circumvent or avoid these problems is the challenge we tackle with the approach presented in this chapter. We implement an online control using DRL to ensure a complete impregnation of the reinforcing fibers during the injection phase, by controlling the input pressure on different inlets. The work is carried out in simulation due to the necessity of many training iterations that is unfeasible in reality and the absence of a machine to carry out the experiments. This work uses this self-learning paradigm to actively control the injection of an RTM process, which has the advantage of depending on a reward function instead of a mathematical model, which would be the case for MPC. A reward function is more straightforward to model and can be applied and adapted to more complex problems. RL algorithms have to be trained for many iterations, for which we developed a simulation environment with a distributed and parallel architecture. We show that the presented approach decreases the failure rate from 54 % to 27 %, by 50 % compared to the same setup with steady parameters. This is the first approach to actively control this process with RL in contrast to existing related approaches.

Publication. The concepts and results presented in this chapter are published in [Stieber et al., 2023a].

7.1 Reinforcement Learning for the Controller

As described in Sections 1.1 and 1.2, one of the main challenges in manufacturing FRP is the heterogenous nature of the input material, the textile. During the injection of the resin, disturbances that stem from this irregular nature of the textile can occur that lead to an incomplete impregnation of the textile, which reduces the stability of the manufactured

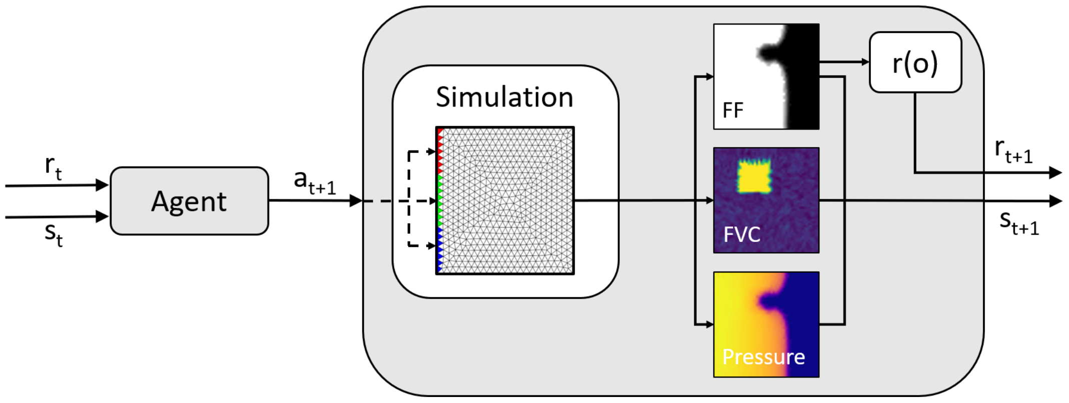


Figure 7.1: Overview of the RL setup for RTM control: The agent receives an observation s_t of the state and a reward r_t and chooses an action a_t. The simulation acts as the environment which takes the action and returns the next observation, containing the flow front image, the FVC map, and the pressure image.

component. This leads to a high reject rate, which can make the production of FRP components uneconomical and unecological. An irregular spreading of the flow front can lead to an air entrapment in the worst case (cf. Section 5.1). By controlling the pressure at which the resin is injected into the mold, the flow front can be influenced to ensure a uniform, complete and fast impregnation of the textile. One difficulty in designing such a controller is the nonlinear and complex relationship between the injection pressure and the spreading of the resin [Wang et al., 2018]. This complicates the classical expert-based controller design typically conducted in control theory, which is based on the formation of a mathematical model of the system under control. Therefore, data-driven approaches using various ML models have been developed in related work [Wang et al., 2018; Demirci and Coulter, 1994a; Nielsen and Pitchumani, 2000]. Initial successes have been achieved, resulting in a more regular distribution of the resin.

In this chapter, the injection pressure in an RTM process is optimized through RL. The process has the form of a linear injection, introduced in Subsection 4.1.2 as *case study B*. A system trained by an RL algorithm is called an agent (cf. Subsection 3.1.3). This agent learns to interact with its environment through trial and error. In doing so, it is driven by rewards it receives in response to its actions, which it tries to maximize [Sutton and Barto, 1998]. The experiments for this work were performed using a simulation of the RTM process. The overall approach is depicted in Figure 7.1. Running a real RTM press would be too expensive and burdensome to perform the number of runs necessary to train the algorithms – a policy that is commonly observed in related high-stakes, real-life physical applications of RL such as autonomous driving or robot manipulation [Zhang et al., 2019; Zhao et al., 2020]. In addition, a distributed and parallel architecture was implemented to take advantage of the available computational resources during training.

7.2 Requirements and Design of Environment

In this section, the requirements and design of environment for our study on optimizing the RTM process using RL are presented. The foundation of our RL environment is built upon the FEM simulation which models the RTM process accurately for agent interaction and learning. To enhance computational efficiency, we employ parallelization techniques, enabling simultaneous training of multiple agents and accelerated learning. A critical aspect of our approach is the design of an effective reward function, which guides the RL agent towards optimizing various factors such as resin flow, pressure, and curing time. Moreover, we discuss the fine-tuning of RL hyperparameters, including those related to the CNN used in our model, to achieve optimal performance. Lastly, the experimental setup for our simulations, describing the specific conditions, configurations, and metrics used to evaluate the performance of the RL agents in optimizing the RTM process, are outlined. This includes details on the scenarios, hardware, and software specifications employed in our study.

7.2.1 FEM Simulation: Basis for the RL Environment

For controlling the RTM process, training in a real-world setup was not feasible, a large number of iterations is required, which would result in many costly experiments and additionally the machinery in CosiMo (cf. Subsection 2.3.4) was not utilizable. Hence, we employed a numerical simulation of resin flow to create a simulated environment that closely resembles the RTM process. However, in contrast to the actual continuous-time state, the state represented by s_t is a simplified abstraction obtained at discrete time intervals. Although our state is more intricate, we only make observations at discrete time points. The action consists of three integer values that control the injection pressure at the three resin inlets, which can be set to five discrete equidistant levels between 0.1 and 5 bar. To be able to execute comparable experiments, some assumptions about process parameters had to be made. Their values were chosen to be constant within realistic magnitudes but would vary, if, for example, other types of resin or textile were used. For the calculations of Darcy's lay (Subsection 2.2.5), we assume viscosity η to be $0,1$ Pas and permeability K to be isotropic, meaning its value is the same in every flow direction. The exact value of K is varied between experiments and will be explained later on. While the simulation requires comparably small time steps to yield numerically stable solutions, we chose a much larger step size of $0,5$ s - or a frequency of 2 Hz - for the RL cycle. This reduces the computational burden, which is necessary to efficiently train agents. Therefore, per RL step, a multitude of simulation steps is executed.

In Subsection 3.2.4 it is described how the simulation was designed to meet the requirements of an RL controller. The exact form of the part and the position of the inlets of the mold is described in detail in Subsection 4.1.2: to reiterate, there are three independently steerable inlets on one side – corresponding to the actions a_t – and the flow front has to flow from left to right.

7.2.2 Feasibility through Parallelization

To implement the interaction of the agent and the simulated environment, the package *Stable-Baselines3* [Raffin et al., 2021] is used, which in turn builds on the package *OpenAI Gym* [Brockman et al., 2016] that provides the algorithms A2C and PPO that are used in this work, with an interface that offers additional parallelization capabilities. Called *vectorized environment*, the extended interface organizes the interaction of an agent with multiple instances of the environment simultaneously. This takes place synchronously so that the agent receives one vector per time step, which contains one observation for each environment instance. Rewards and actions are handled the same way. Figure 7.2 shows the common flow of RL extended with parallel environments. The interaction of the agent with each instance of the RTM process

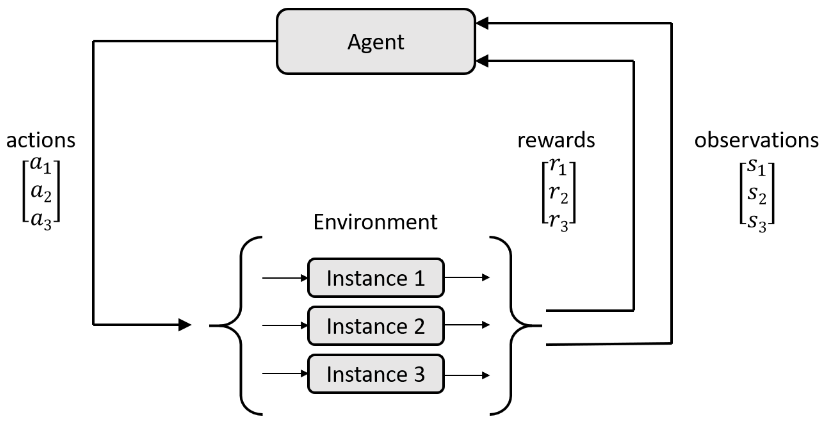

Agent
actions
$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$
rewards
$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}$
observations
$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$
Environment
Instance 1
Instance 2
Instance 3

Figure 7.2: The interaction of an agent with a vectorized environment, exemplified with three parallel instances for a single time step. The actions, observations, and rewards are each grouped into vectors, with entries of the same position belonging to each other. Thus, action a_1 is passed to instance 1, from which observation s_1 and reward r_1 can arise. This mapping is important for the agent to correctly evaluate the effects of its action. For clarity, indexing by the time step was omitted and indices refer to the environments. The sequence of actions, observations, and rewards behaves according to the common sequence of RL (cf. Figure 3.6).

is the same as presented in Section 7.1. The architecture of the training context follows a server-client structure. The client side program contains the agent and the training routine, while multiple servers run the numerical simulation described in Subsection 3.2.4. This client side is implemented in Python and uses resources provided by Stable-Baselines3: algorithms and training methods, plus the ability to record data during training. The `RemoteRtmEnv` class is the core of the architecture and implements the interface of the vectorized environment. It organizes communication with the simulation servers and serves as an interaction point for the agent. The interaction of the agent and the vectorized environment is synchronous, following the typical RL flow shown in Figure 7.2. However, communication with the servers must be asynchronous to take advantage of the parallel computational capabilities. The `RemoteRtmEnv` class resolves this conflict. Its main purpose is initiating the next time step. The flow of this interaction is shown as an adapted sequence diagram in Figure 7.3. The agent passes the vectorized actions, that is, a vector of three entries (corresponding to the three inlet pressures) for each environment instance registered at the beginning of training, to the `RemoteRtmEnv` object. This sends the actions, in the form of Hypertext Transfer Protocol (HTTP) requests, to the servers, asynchronously triggering the simulation of a time step. On the server side, a Julia program facilitates the integration of the simulation solution presented in Subsection 3.2.4. It uses multiprocessing to take advantage of the full processing power of the available compute to provide multiple environment instances per server. A specialized process acts as a coordinator, receiving HTTP requests from the client and delegating them to the rest of the processes. These act as worker processes, each executing one instance of the simulation. As soon as the worker processes receive an action from the coordinator, they simulate a time step of the filling process and subsequently generate images representing their current state. One time step corresponds to many iterations of the numerical simulation, as described in Subsection 3.2.4. Worker processes consistently retain their state between two consecutive time steps. Consequently, it is essential to maintain an accurate mapping of actions to environment instances throughout all

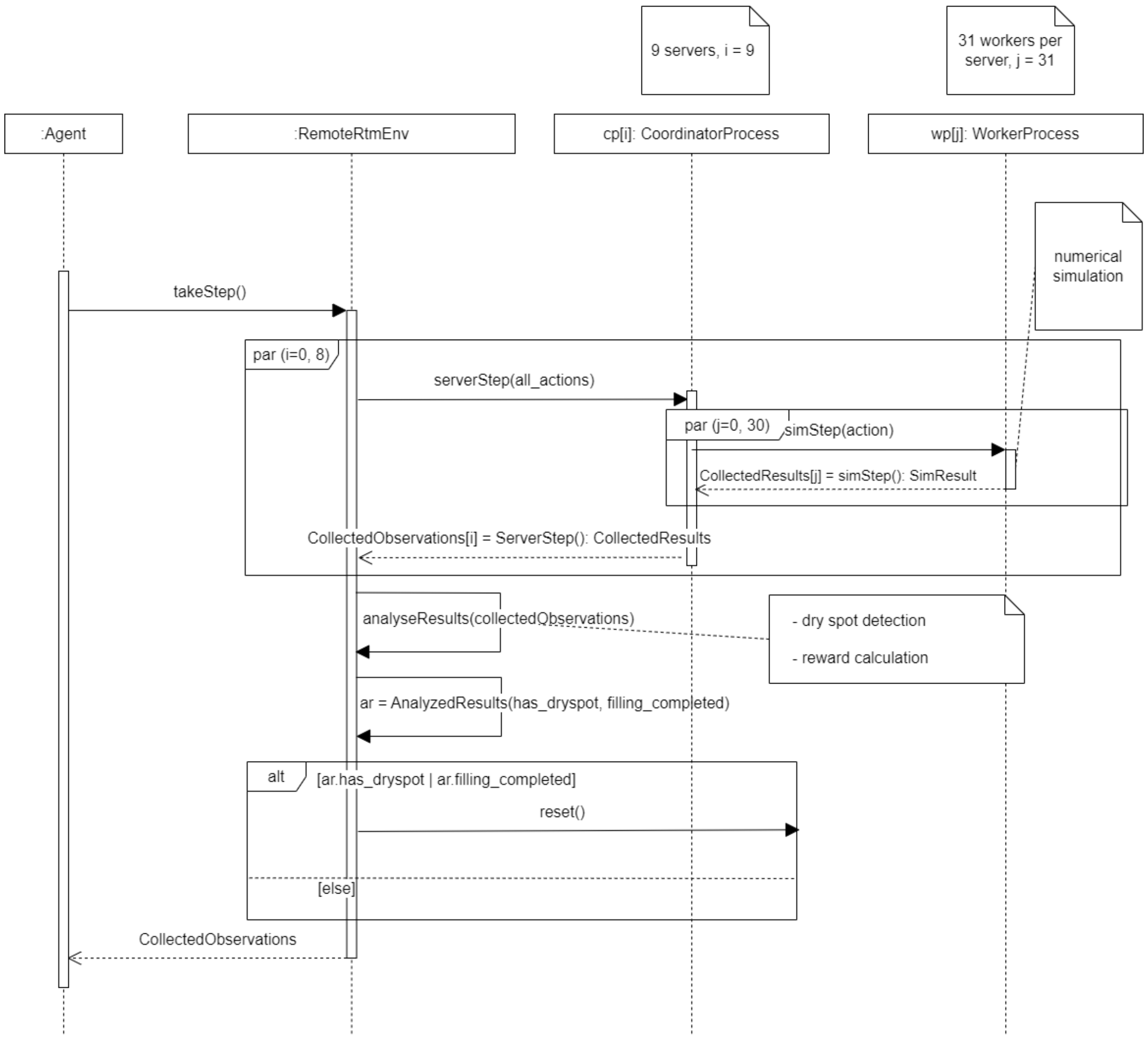

9 servers, i = 9
31 workers per server, j = 31
:Agent
:RemoteRtmEnv
cp[i]: CoordinatorProcess
wp[j]: WorkerProcess
numerical simulation
takeStep()
par (i=0, 8)
serverStep(all_actions)
par (j=0, 30)
simStep(action)
CollectedResults[j] = simStep(): SimResult
CollectedObservations[i] = ServerStep(): CollectedResults
analyseResults(collectedObservations)
- dry spot detection
- reward calculation
ar = AnalyzedResults(has_dryspot, filling_completed)
alt
[ar.has_dryspot | ar.filling_completed]
reset()
[else]
CollectedObservations

Figure 7.3: Communication of the agent and the training context during the execution of a time step, depicted in a sequence diagram-like manner (Added parallel execution (par) to indicate distributed multiprocessing). The training context consists of the `RemoteRtmEnv` class, which satisfies the agent's interface, and several servers, each running a coordinator and several worker processes.

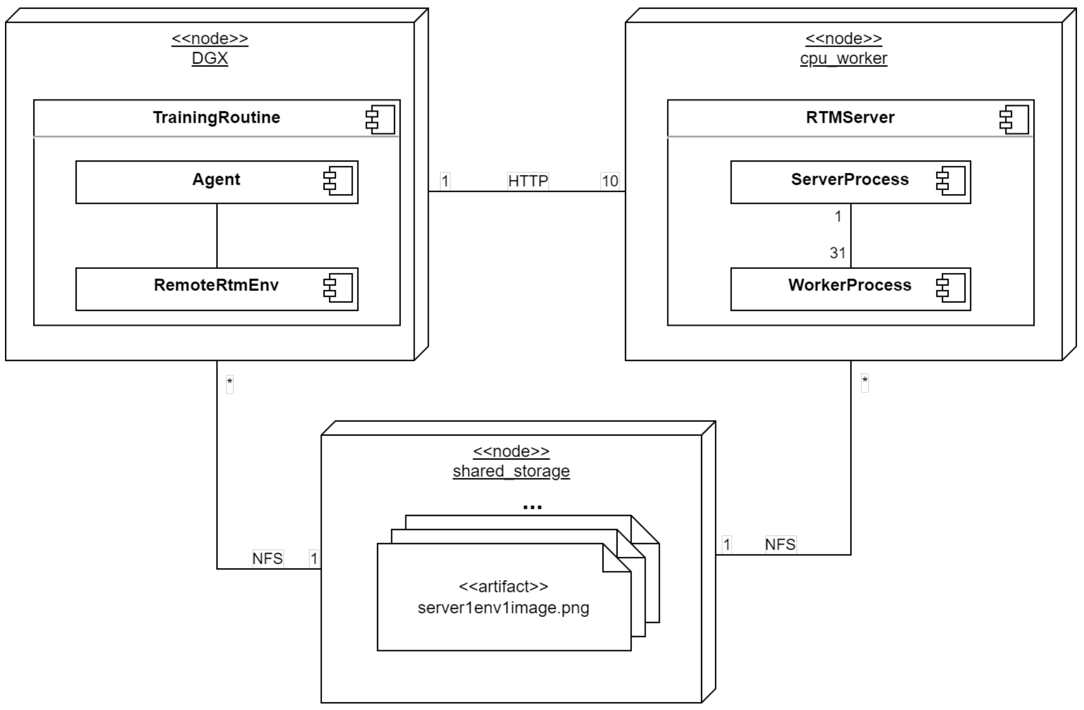


Figure 7.4: Distribution of execution units on the computer network. The `RemoteRtmEnv` component runs on the client, hiding the distributed architecture from the agent. In the background, it communicates with ten simulation servers, which in turn run up to 31 worker processes each. One server is used for validation during training, leaving nine servers for training. The shared file system facilitates the transfer of large amounts of image data.

time steps, ensuring the agent receives the correct trajectories. The coordinator waits for all worker processes to finish their calculations before sending the collected images back to the client. Upon receiving responses from all servers, they are organized to preserve the correct ordering. Reward calculations, along with some image processing, occur on the client side, which determines if an episode should be terminated. This happens when a dry spot is detected or the part is entirely filled. The `RemoteRtmEnv` object then sends this information to the servers, where the corresponding environment instances are reset. The old observations of these environment instances are replaced with the new ones and then the sorted and possibly reset observations are returned to the agent along with the rewards. The agent optimizes its policy based on the rewards received and selects new actions based on the observations received. Then the interaction starts all over again. As indicated, the individual components of the architecture are executed in a distributed manner on different computing capacities to achieve a high degree of parallelism. Figure 7.4 shows how the individual components of the architecture are distributed across different high-performance computers. The client program runs on an Nvidia DGX and can thus use GPU support to train the agent's NN,depending on the algorithm used, value networks and/or policy networks. The simulations do not require GPU support and run on computers without graphics cards. These contain processors with 32 cores each, so minus one delegating process, there are 31 worker processes per server that can run instances of the simulation. A total of ten CPU servers are available to perform this work, one of which was used during training for validation, described in detail in Subsection 7.2.5. Thus, 279 simulated instances of the RTM process are available for the agent to gain experience in parallel. The compute cluster features a shared storage accessible by all computers through the Network File System (NFS) protocol, enabling a unified file system. This facilitates efficient transfer of observations, as image files, from the servers to the client. Smaller amounts of data, such as the actions and other control signals, are sent via HTTP over the local network. In the experiments conducted, a training run including validation required an average of 5 hours and 29 minutes. In the process, 7168 steps of the vectorized environment are executed, which corresponds to almost two million interactions of the agent with an environment. This means that, depending on the average episode length of each trial, between $60,000$ and $100,000$ injection cycles are executed per training.

Summarizing, both the parallelization and a simulation that makes it possible to change parameters (cf. Subsection 3.2.4) make this approach fundamentally possible.

7.2.3 Reward Function for Optimizing the RTM Process

We designed a reward function that depends on the flow front image only. The main purpose of the proposed reward function (7.1) is to reward the timely and complete filling of the part and punish the occurrence of dry spots.

$$r(o) = a \cdot \frac{\mathit{filled}}{\mathit{step\_count}} - b \cdot \mathit{dryspot} - c \cdot \frac{1}{h \cdot R} \cdot \sum_{i=1}^{R-2} \sum_{j=1}^{h} \left((R - i) \cdot (1 - o_{i,j}) \right)^2 \tag{7.1}$$

The variables *filled* and *dryspot* take either values of 0 or 1 and indicate if the respective event has occurred. Both can only be triggered in terminal states because the environments automatically reset in either case. The input to the reward function is the flow front image o, while the notation $o_{i,j}$ refers to the pixel in the i-th row and j-th column. The height $h$of o is 50 in our experiments, and R is the column index of the rightmost pixel that has been reached by the resin. The number of steps elapsed since the beginning of the episode is enumerated by *step_count* and the weighting factors have been chosen as $a = 3000$, $b = 100$ and $c = 10$ in prior experiments. In an episode of usual length - when *step_count* is around 30 - the choice of a and b causes the first two terms to be of the same magnitude.

Apart from the sparse reward mechanism, which only yields a signal - either positive or negative - in terminal states, that come from the first two summands in (7.1), we added an auxiliary goal to guide the agent to the desired behavior in the sense of reward shaping. This introduces the third term, which measures the flow front uniformity, described visually in Figure 7.5. In our setup, the optimal flow front would be an orthogonal line moving from left to right. When evaluating the reward function, we place this target flow front at R. Then the MSE between o and this target, weighted with each pixel's distance to R, is calculated, with the exception that the two columns nearest to the target line are excluded. Thereby, small irregularities have a comparably small impact on the reward signal, whereas deeper bulges are weighted quadratically higher. This motivates the agent to keep the flow front as even as possible, which is a good step toward preventing dry spots.

Two mechanisms implicitly add the incentive to finish episodes quickly. (1) By weighting $filled$ inversely with *step_count*, the agent receives a higher reward at the end of short episodes. (2) Because the deviation from the target flow front is weighted negatively, the reward signal is negative for all states, except for the terminal, which can possibly produce a positive value. To maximize the rewards accumulated over an episode, the agent should finish the episode in as few steps as possible, while still seeking to fill the part completely by avoiding dry spots.

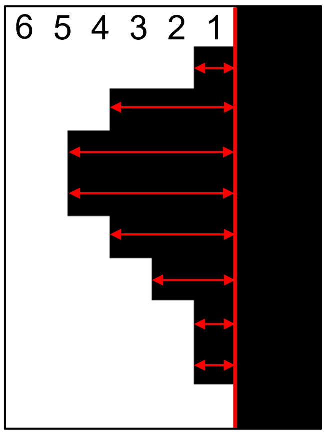


Figure 7.5: Enlarged view of a flow front that advances from left to right, with the vertical red line marking the most advanced point. The distance of a column of pixels to the foremost point of the flow front is plotted in the figure. All pixels to the left of the mark have a nominal value of one, corresponding to a white coloration and a fill level of 100 %. Therefore, the value of these pixels, excluding the columns with the distance of one and two pixels from the calculation, is subtracted from one and enters the reward function averaged and weighted negatively by the squared distance.

7.2.4 Further RL Hyperparameters

We used *Stable-Baselines3*, which, inter alia, provides implementations of A2C and PPO with pre-tuned hyperparameters.We adjusted the hyperparameter `n_steps` to 20, which sets the number of steps to include per policy update. This improved the agents' performance, while for the other hyperparameters, no changes were found to be beneficial.

Regarding the NN trained by the algorithms, we used the same architecture in all experiments. Parts of the network are shared between the policy and the value function. The shared part consists of three convolutional layers followed by a feed forward layer of width 128. The first convolutional layer uses 32 kernels of size 8×8 with stride 4, the second 64 kernels of size 4×4 with stride 2, and the third 64 kernels of size 3×3 with stride 1. This convolutional network architecture was used by Mnih et al. [Mnih et al., 2016] in an influential work on the application of DRL and an implementation is provided by stable-baselines3. Next, the network splits into two heads, each containing a feed forward layer of width 32. The output of the policy contains three values and represents the action of the agent, while the value network predicts one value. In all layers, ReLU is used as activation function and Adam as optimizer.

7.2.5 Experimental Setup

In Section 4.4, the general setup for case study B for controlling the process is described. Here, each series contains six experiments resulting from combining possible observation spaces with two considered algorithms.

The agent interacts with all parallel environments in a synchronous manner and the experiences are batched and accumulated similar to stochastic gradient descent. The interaction with every single environment is similar to how an agent would be integrated into the RTM process as a controller, as can be shown in Figure 7.1. At each discrete step the agent receives an observation and chooses an action.

We experimented with three different observation spaces to evaluate which physical quantities yield the most value for the agent. The considered quantities are the filling state, the preform FVC and the pressure inside the tool. Filling state and pressure are both dynamic, meaning they change over the course of one run while the FVC is a fixed quantity for one run. The filling state represents the spreading of the resin, which is to be optimized and therefore included. The flow speed and thus the flow front is, as stated by Darcy's law, influenced by the permeability and the pressure gradient. We substitute the permeability information with the FVC of the textile, as it is the only factor influencing the permeability, according to our assumptions regarding the textile. In real-world scenarios there is usually a reasonable starting point that contains the plan of the used materials, referred to as ply stacking sequence (cf. Subsection 2.2.4), but misses additional possible deviations of the base material, the textile. Instead of a pressure gradient map, we provide the agents with a simple map of the pressure inside the mold, which is, again, more realistic to measure in a real process. In our simulation, all of those quantities are observed as 50×50 pixel grayscale images that display their spatial distribution. While the FVC remains constant throughout one filling cycle, the filling state and pressure evolve with time. The simplest observation space containing only the flow front image is from now on called **flow front (observation) (Ff)**, adding the FVC map gives the observation space **flow front and fiber volume content (observation) (FfFvc)** and adding the pressure image leads to **flow front, fiber volume content and pressure (observation) (FfFvcP)**.

A filling cycle consists of a series of interactions and ends when one terminal condition is fulfilled. This can be either the occurrence of a dry spot or the complete filling of the part, each of which will trigger a specific reward signal contributing to the reward function described in Subsection 7.2.3. A filling cycle definitely terminates in finite time, rendering this case an episodic problem in terms of RL. During training, finished simulation instances are automatically reset with randomized initial conditions concerning the preform FVC and thus the permeability.

7.3 Evaluation

In the following chapter, the results of these two series of experiments – light and strong perturbations – are presented to assess whether RL can provide an advantage in controlling the RTM process. The performance of different RL control strategies is evaluated using four metrics:

1. The average cumulative reward per episode or *return*. This reflects the overall utility of each episode. The higher the better [Sutton and Barto, 1998].
2. The average episode length in steps. This reflects the efficiency of each episode. The shorter the better. It can also be converted to seconds by dividing by 2.
3. The adjusted average episode length in steps for successful episodes only. This reflects the efficiency of each episode without considering early terminations due to dry spots. Dry spots are undesirable and should be minimized.
4. The dry spot rate for failed episodes. This reflects the frequency of dry spots in the evaluation dataset. The lower the better.

Note that points 1 and 2 are applicable for both experiment setups while 3 and 4 are only useful for the *strong perturbations* scenario. The RL control strategies are compared with a constant

Table 7.1: Results from experiments with slight perturbations.

Algorithm	Observation Space	∅ Reward p. Ep. ⇑	∅ Length ⇓
Static	-	−107.8	**29.52**
PPO	Ff	−51.2	30.59
PPO	FfFvc	−23.2	31.53
PPO	FfFvcP	**−22.2**	32.84
A2C	Ff	−110.2	29.77
A2C	FfFvc	−126.5	29.96
A2C	FfFvcP	−76.7	32.55

baseline policy that applies a fixed injection pressure of 5 bar to each gate in every step. This corresponds to the static version of the RTM process commonly employed industrially [AVK, 2013]. The aim is to demonstrate the superiority of a controlled processed through RL over this static baseline.

7.3.1 Slight Perturbations

In this series of experiments, minor disturbances of the flow front occur during a static injection due to slight perturbations of the preform permeability. Table 7.1 shows the average rewards per episode and the average episode lengths resulting from the different pairings of algorithms and observation spaces.

For comparison, the values obtained by a static injection with maximum injection pressure are given. The high injection pressure leads to the fastest injection times, since a steering signal can only be a decrease of the pressure, which leads to longer injection times. Since no dry spots occur in the slight perturbations setup, the rate of dry spots is not used as a metric. The average episode length, on the other hand, is used as a quality measure of an agent without restriction, since episodes are not terminated prematurely and shorter episodes are desirable. The metrics from Table 7.1 on the test set, show that PPO is better suited for this use case than A2C. While PPO agents indeed achieve a higher average reward than the static injection process in all configurations, the extra information about the FVC map seems crucial. With access to the FVC map, an agent achieves a significantly higher reward, while adding the pressure information provides only a marginal advantage. In contrast, agents trained by A2C only manage to learn a policy that can achieve an advantage over the static process when having access to the full information. Furthermore, it is noticeable that agents achieving higher rewards usually take longer to complete an episode. To reach an even flow front, the agent must match the flow speed in higher permeability areas to the lower speed that is possible when impregnating the lower permeability inserts, which causes the filling cycle to take longer in general. Accordingly, agents behaving similarly to the static process are also nearly as fast.

These observations are illustrated in Figure 7.6 by comparing two differently trained agents: PPO/Ff and PPO/FfFvcP. For this purpose, three snapshots of two filling processes, resulting from applying the respective control strategies to the same test case, are shown. The agent PPO/FfFvcP has an advantage due to its knowledge of the local FVC since it can preemptively steer against flow front disturbances that will happen in later process stages – similar to an MPC approach. In the beginning, agent PPO/FfFvcP applies the maximum possible value of 5 bar relatively constant to gate 3, while slightly lower values are selected for gate 2 and gate 1.

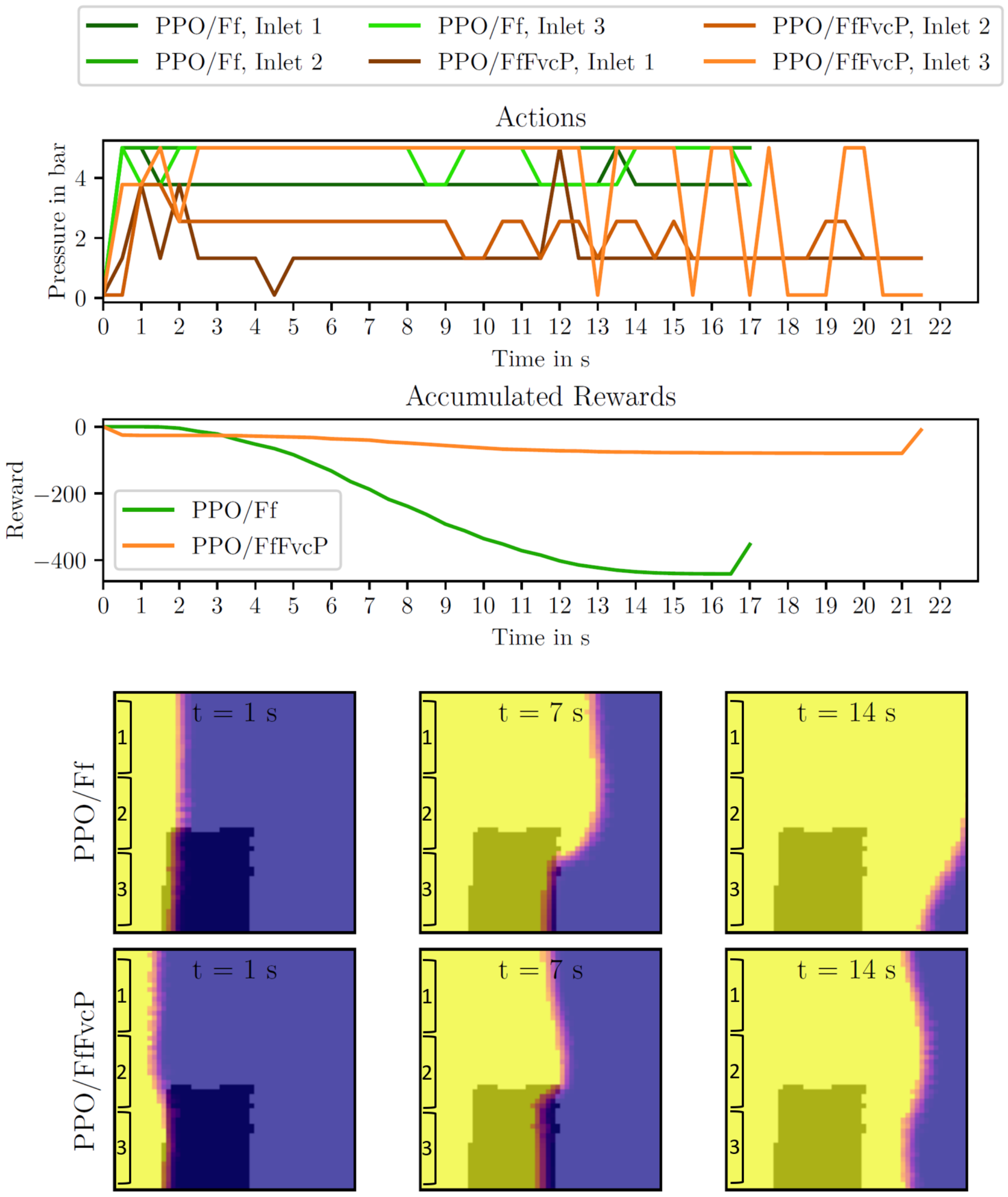


Figure 7.6: The comparison of the control strategies of agents PPO/Ff and PPO/FfFvcP within the slight perturbations regime, using three snapshots. The local FVC of the textile can be seen transparently in the flow front images, with the insert with higher FVC being darker. In addition, the evolution of the cumulative rewards up to the respective time point, as well as the evolution of the pressure values at the individual gates representing the agents' actions, were visualized.

This is displayed in the action plot in Figure 7.6. As a result, the image taken after 1 s shows a curvature towards the insert. This causes a negative reward signal, as can be read from the reward plot. When passing the insert, the flow front is slightly delayed and is subsequently almost straight. In consequence, the reward signals received per step are nearly zero and the cumulative rewards remain almost constant until the positive signal for finishing the episode successfully is triggered. Towards the end of the filling process, there are clear fluctuations in the actions of agent PPO/FfFvcP, but these have little influence on the flow front since it is already far away from the resin inlets. Agent PPO/Ff, in contrast, can only react to disturbances in the flow front that have already occurred. Since the process reacts sluggishly to changes in the injection pressure, especially when the flow front is already more advanced, this agent can often achieve only slight advantages over the static process. Agent PPO/Ff tries to create a straight flow front from the beginning by applying uniform, high-pressure values. However, already after 1 s a slight disturbance can be seen since the flow front has already reached the insert. The agent reacts to this by slightly reducing the pressure at gate 1, which is diagonally opposite to the disturbance. However, this does not prevent the flow front from progressing significantly faster on the upper half of the image. After achieving quite good rewards in the beginning, the cumulative reward drops sharply due to the irregularity of the flow front.

The comparison shows that agent PPO/FfFvcP is able to act far-sighted due to its additional information and accepts suboptimal results in the beginning in order to achieve better ones in the long run. Agent PPO/Ff, on the other hand, makes optimal decisions at the beginning, but since it does not know where the insert is located, it falls behind in terms of cumulative rewards after only a few steps.

In conclusion, it is possible for RL agents to learn strategies that perform better than a static injection in terms of the reward function and in the presence of minor disturbances. The more information is available to the agent, the better is its policy. Especially the FVC map makes a big difference because it allows the agent to steer preemptively against future disturbances. Possible use cases of preemptive steering are described in Section 7.4.

7.3.2 Strong Perturbations

In this series of experiments, the insert placement and the level of permeability perturbation cause strong irregularities of the flow front that can result in the formation of a dry spot. During a static resin injection, 54 % of the test cases were terminated early due to the detection of a dry spot. Although the depicted ratio of failed runs results from an unusually increased usual intensity of material perturbations, it still serves as a striking showcase of the RL controller's capabilities. First, general observations from this experimental campaign are discussed and subsequently, one example run is described.

Table 7.2 lists the results obtained when training the six pairings of algorithms and observation spaces, similar to Table 7.1. The metric of average cumulative rewards per episode is of limited use, because, first the rewards obtained are generally lower than in the first experiment, which can be explained by the stronger perturbations of the flow front. Second, within this experiment, the PPO agents can generate higher rewards than the A2C agents, but the choice of learning algorithm has little influence on the dry spot rate, given the same input information. Third, the A2C agents receive significantly lower rewards on average, in particular even less than the static injection. Nonetheless, agent A2C/FfFvcP is the most successful in terms of preventing dry spots.

Also, the mean length cannot be used unambiguously to compare strategies. The static process requires the fewest steps per episode, this is because over half of the test cases are terminated prematurely, which usually occurs between step 10 and step 15. The conclusion that longer episodes indicate better agent behavior is also incorrect. Agent A2C/FfFvc stands out, producing comparatively long episodes but still having a similar dry spot rate as, e.g., PPO/FfFvc, whose episodes are on average about 9 steps shorter. Thus, the episodes of A2C/FfFvc are not longer because it prevents more dry spots and thus leads more episodes to successful completion, but because the policy of A2C/FfFvc chooses comparatively low-pressure values, causing the flow front to progress slower and the episodes to last longer. This has been found in graphical evaluations of the policy but is not explicitly shown here by figures for reasons of space and relevance. Therefore, Table 7.2 reports the mean length cleared of episodes that have been stopped early as *adjusted mean length.* Due to the occurrence of dry spots, the dry spot rate is used as a metric in this experiment. This has the advantage that preventing dry spots corresponds to a real advantage, while the rewards obtained are subject to the assumptions of the reward function.

All agents achieve a better dry spot rate than the static process. The trend is that, as in the first series of experiments, agents with more information can achieve better results, which means lower dry spot rates in this case. In contrast to the first series of experiments, the best agents do not show major disadvantages in terms of filling speed, yet the static process is still the fastest. This becomes apparent when considering the dry spot rate and the adjusted mean length together, which shows that agents that can prevent the most dry spots still have a comparatively high filling speed. There is no clear trend between the two algorithms, in particular, an agent trained by A2C performs best. Figure 7.7 shows a comparison of the best agent A2C/FfFvcP and the static injection, in which the latter leads to the formation of a dry spot. The actions of the static agent are plotted in the action graph for comparison, with the maximum value of 5 bar always applied to all inlets. The learned policy also behaves almost constantly. At inlet 2, which is on the horizontal line of the insert, 5 bar is applied, while the value at the two outer inlets is reduced to half: 2.5 bar. Due to the overall lower injection pressure, the regulated flow front moves slightly slower than the static one. In the snapshots, although both flow fronts are approximately equally advanced, there are 1.5 s between each. The static flow front advances rapidly at the edges of the component so that the part delayed by the lower permeability of the insert lags. As soon as the advancing parts or “arms” of the flow front have passed the insert, they close again because the textile has a higher permeability there and is penetrated more quickly. The delayed region of the flow front is so far behind that the insert is not completely impregnated before the flow front closes behind it. Thus, air entrapment occurs, and the episode is aborted. This can also be seen in the reward graph in Figure 7.7, as a strong negative reward signal is triggered by the dry spot after 6.5 s. The controlled flow front moves slower, especially at the edges of the component. Although a slight advance on both sides of the insert cannot be prevented, the resin has almost completely penetrated the insert when the two “arms” merge, as can be seen in the snapshot after 6.5 s. Thus, a dry spot does not form, and the filling process can be completed.

We also conducted another experiment to test if pre-training on broader knowledge and inference on a smaller observation improves the performance. The motivation behind that idea was that in simulation, the complete observation is easily obtainable whereas in a real process, only parts of the observation can be retrieved. Pre-training the agent on broader knowledge in simulation could have helped improving performance for the limited observation reality has to offer. To do so, we used a model that was trained on a three dimensional input (FfFvcP)

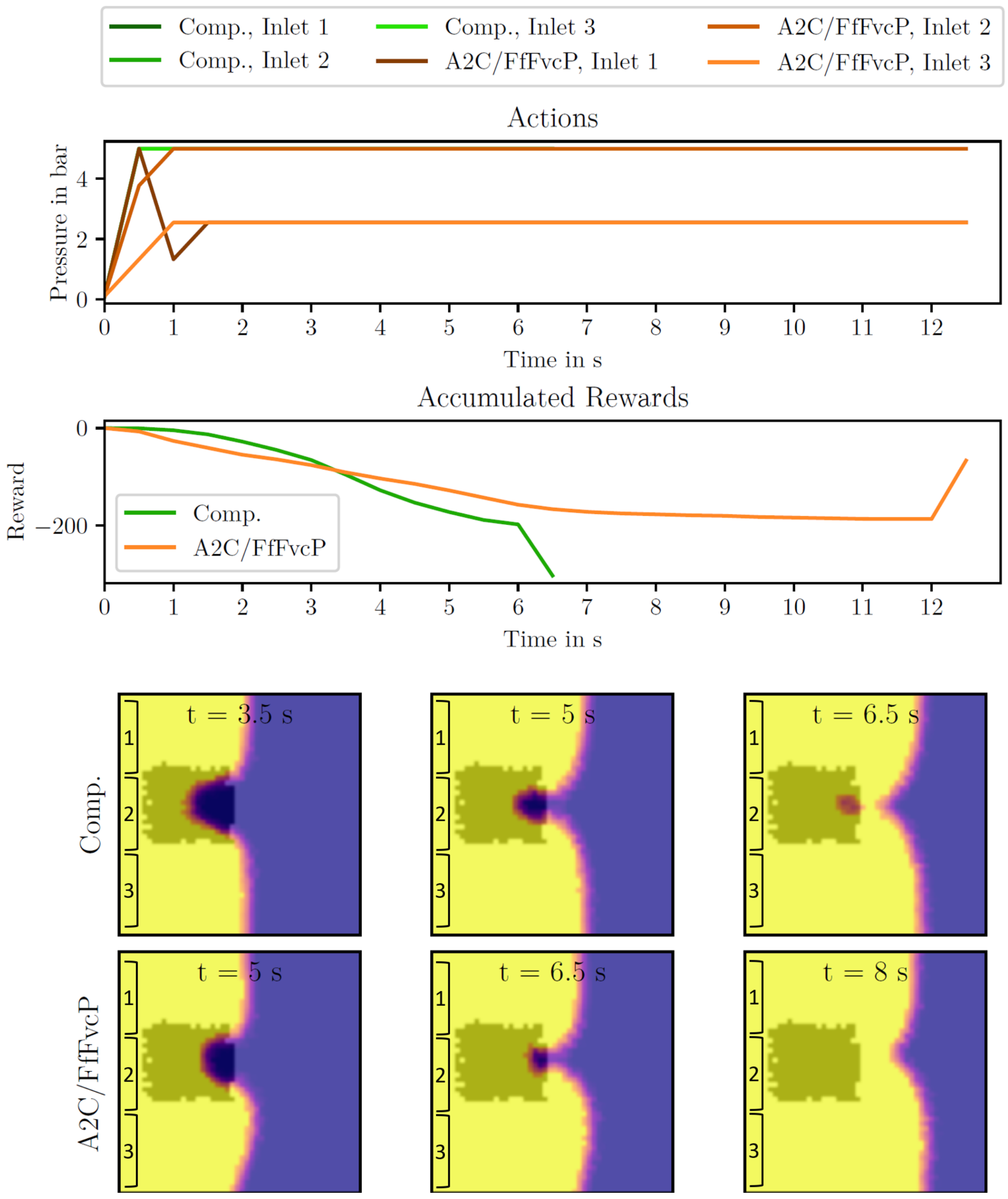


Figure 7.7: Snapshots from a filling cycle with strong perturbations controlled by A2C/FfFvcP. A static injection (*Comp.*) is depicted as a comparison.

Table 7.2: Results from experiments with strong perturbations.

Algorithm	Observation Space	Mean Reward per Episode	Mean Length	Dry Spot Rate [%]	Adj. Mean Length
Static	-	-157.4	21.15	54	**24.91**
PPO	Ff	-163.9	23.72	40	29.27
PPO	FfFvc	$\mathbf{-132.8}$	**20.82**	37	24.95
PPO	FfFvcP	-138.5	22.67	34	26.64
A2C	Ff	-322.8	29.27	41	35.76
A2C	FfFvc	-275.1	29.37	39	35.61
A2C	FfFvcP	-221.5	23.82	**27**	26.82

and give it only the flow front as observation in inference with zeros as inputs for the other dimensions fvc and p. This resulted in worse performance than even the baseline: With PPO the mean reward was -345.2, the mean length 21.4 and the dry spot rate 58 % and with A2C it got even worse, with a mean reward of -420.4, a mean length of 23.9 and a dry spot rate of 70 %. This did not come as a big surprise, since the model was fed many zeros that it was not shown in training, which often leads to underwhelming results.

To summarize, this experiment showed that RL algorithms can learn control strategies that reduce the dry spot rate of the simulated RTM process. In doing so, they do not necessarily have a disadvantage in filling speed. When provided with more information, the agents can preemptively steer against perturbations and thus achieve better results.

7.4 Related Work

In Subsection 2.3.3 the optimization possibilities of LCM methods are described: passive and active. The described approach can be classified as an active method, where the measurement is an image of the flow front, inter alia, and the variable to be controlled is the pressure profile applied to the resin inlets in an RTM process. Several attempts to design such a controller were made so far and will be presented in the following.

Demirci et al. [1997] trained a NN using a numerical simulation to predict the flow front position at the next discrete time step from the current flow front position and the injection pressure applied. They used a plane, but not rectangular mold with two inlet gates for their experiments. They defined an optimal flow front as the control target. The output of the controller is the flow rate profile that minimizes the difference between the flow front position predicted by the model and a predefined target flow front. To determine this solution, they tested several optimization methods, of which the *downhill simplex* method of Nelder and Mead [1965] proved to be the most efficient [Demirci and Coulter, 1996]. Our approach differs from this by using the RL paradigm instead of an control theory architecture. Due to the technical limitations of the time, they used a comparatively coarse measurement of the flow front and a shallow network architecture, while we were able to heavily increase the resolution, the network depth and the number of training samples. Even though the networks serve different purposes, theirs predicts the flow front progressions while ours realizes the agent's policy, a deeper network can be expected to account better for the subtleties of the flow dynamics. In Section 5.2 we discuss forecasting the process with a CNN architecture, which is also close to their task, but does not address readjusting the flow front, only predicting the future flow.

Nielsen and Pitchumani published a series of papers on the design of an MPC for LCM processes. They used the *simulated annealing* method [Kirkpatrick et al., 1983], a heuristic optimization procedure, to determine the optimal flow rate for each inlet following the MPC pattern, e.g. by sampling from a predictive model. The selected flow profile is optimal in a way that it minimizes the Root Mean Squared Error (RMSE) between the desired and the predicted flow front in the next time step. To predict the flow front behavior in a quadratic, plane setup with three resin inlets, they used a NN-based dynamics model [Nielsen and Pitchumani, 2000] as well as a numerical simulation [Nielsen and Pitchumani, 2002b]. In the first case, the training data for the NN was obtained from a numerical simulation. The NN was shallow compared to present-day standards and predicted the flow front position at eight discrete sensing lines. To mimic real perturbations, they used textile inlets and artificial channels. This induced flow front disturbances and race-tracking effects to test the MPC against. In another work [Nielsen and Pitchumani, 2002a], they extended their controller to include a fuzzy logic estimate of the permeability of the textile. In real experimental setups, the resulting flow fronts were found to closely match the respective targets. While the setup is similar to this work, the provoked flow disturbances did not cause the formation of dry spots. Like Demirci and Coulter, they used a NN architecture that is shallow compared to today's technical possibilities. Their approach is an MPC, with a NN for the model, while we apply RL techniques directly to the problem.

[Wang et al., 2018] designed an MPC to control the flow front progression as well as the flow velocity. They used a rectangular, plane mold and added textile inserts to provoke flow front perturbations. Three injection gates are placed on one side of the mold. They use an Autoregressive Model with Exogenous Input (ARX) as a flow front predictor to account for the nonlinear characteristics of the RTM process. The flow front progression is measured and predicted at three discrete points, which lie on horizontal lines evolving from the inlet gates. The parameters of the model can be identified *online*, i.e. at runtime, by a recursive least squares method. The approach was tested in a real setup. A response to disturbances occurring in terms of flow front progression and flow speed was evident in the applied pressure profiles. By this, the controller was able to steer against when the flow front was lagging behind. The ARX model allowed training exclusively on online process data, but comes at the cost of using and providing very coarse information in contrast to image-based DRL approach, presented in this thesis. As pointed out before, our experiments include cases where the formation of dry spots could be prevented, while Wang et al. presented only cases that are similar to the slight perturbations of our setup, shown in Subsection 7.3.1. As the publication lacks information on how the optimization problem at each time step, i.e. the search for the optimal pressure profile, is solved, it is unclear how far sighted the controller can act, which is a general strength of RL. Another advantage that comes with our proposed architecture is the possibility to easily include various physical quantities, e.g. the map of the FVC. Also other information maps could be integrated, i.e. regarding the permeability or the fiber orientation, which remains future work. Wang's approach is not as easily extendable, their model is not specified further than "Matlab Optimization Toolbox". RL yields further advantages, as it is easier adaptable to other (more complex) geometries and the (re-)training can be automated with enough compute. Further it can handle nonlinear dynamics whereas MPC is using often linear models only. Additionally different types of goals can be integrated through the reward function: dry spot avoidance, speed of injection.

The presented approaches have in common that they all rely on a predictive model of the flow front dynamics to determine a near optimal action at each step. Additionally, all of them measure the performance through the difference of the actual flow front from an optimal one, which would be a perpendicular line in the linear injection setup. Our work differs from those approaches because model free RL approximates functions that map measured values directly to the next action. Therefore, *no model* of the process is needed and no elaborate optimization needs to be performed at each step. Further, we do not use the straightness of the flow front as a metric per se, but we optimize to reduce dry spots and thus reduce the failure rate of the process. The correct form of the flow front is a factor in our proposed reward function (cf. Subsection 7.2.3).

Another approach based on control theory is presented by Hergan et al. [2020]: the authors control the flow speed during the manufacturing of hybrid composite metal part, that is carried out with RTM. This work is focused on the process and the material science to optimize bonding between the different materials. They rely on singular pressure values from two sensors inside the mold and model the fluid flow based on Darcy's law and thus the model of the flow front is rather approximately estimated opposed to the approaches presented above, that featured higher resolution representations of the flow front. Nonetheless, this is an interesting work since it represents an approach that was often presented in material science and process engineering literature but also in industrial manufacturing of composites: using a very limited amount sensors to test certain material features.

Szarski et al. published two attempts on the application of RL methods to LCM processes. In a first work [Szarski and Chauhan, 2023] they used RL to optimize the fluid flow by determining the placement of flow enhancers. In another study [Szarski and Chauhan, 2021] they tried to optimize the temperature profiles applied to the tooling with the goal to enhance the resin flow. Both approaches qualify as passive control methods, because a solution to the respective problem is determined prior to the execution of the process, as opposed to our approach, which is an active one.

Chapter Summary and Outlook

In this chapter a way to control the RTM process with DRL was shown. The key factors to make this approach feasible were the simulation, of which the parameters can be adjusted during runtime and the parallelization of the RL environments. The approach showed promising results, the higher the information content from the observation of the environment, the better were the results. In Subsection 2.2.4 it is described, the configuration of preforms is saved in “ply stacking sequences”, which hold a map of the textile information of the product and possible obstacles for the flow front i.e. styrofoam for sandwich components. That means the information about FVC and other obstacles is available and could be used as an information within the RL controller. Other obstacles, i.e. wrinkles from wrong handling still remain as unforeseen changes to the textile and need to be detected or avoided without the information from the ply stacking sequences.

For other processes from the LCM family, it would be necessary to adapt the simulation and the reward function. The simulation would need to be parametrized again with the desired process criteria. The reward function needs to be adapted in a way to fit different injection patterns, which may result from other injection strategies, i.e. injection points in different places. Aside from that, other actuators may be possible to integrate within an RL controller, such as additional vents.

Chapter 8

Conclusion & Outlook

There are various methods for producing CFRP, of which LCM was the focus of the applications presented in this thesis. This family of processes poses challenges such as precise resin flow control and managing process-induced errors. The production of CFRP is further complicated by the inherent characteristics of the base materials, including variability in the textile base material, and the anisotropic nature of CFRP. These factors make CFRP manufacturing a complex task that necessitates careful consideration of process parameters and material behavior, and makes it a good application for ML. This work proposed several ways to tackle these challenges which are summarized in the following.

8.1 Achieved Contributions

Inspired by the successes in other domains [Tan et al., 2018; Zhu et al., 2023; Tercan et al., 2018], we were the first to leverage the possibilities of *sim-to-real transfer learning* for LCM, which uses both simulated and real-world data. Due to the scarcity of real data and the abundance of simulation data, the strategy involves initially training models on a large simulated dataset and subsequently to retraining them with a small amount of real data to adapt the models to real-world conditions and bridge the gap between simulation and reality. This work presents several of the first published *deep* learning approaches to analyze and optimize LCM, which is enabled by the sim-to-real learning approach and the abundance of data from simulation. We also focused on the application of data-driven methods opposed to purely physics-based methods, because we wanted to have the possibility to adapt the trained models on data from the real process and not rely purely on mathematical models, that might not cover the reality in full.

The first application is the *analysis of LCM processes* with the aim of detecting *dry spots* using sensory data, opposed to analyzing the curing rate, as presented in related literature [Pantelelis and Bistekos, 2010; Zhang and Pantelelis, 2011]. This approach was later expanded to incorporate real data for sim-to-real transfer learning. Additionally, we explored the prediction of future fluid flow within the LCM process based on the current injection course, utilizing both simulated and real data and adopting the strategy of learning from simulation data first, followed by real data. Thuerey et al. [2020] predicted air flow around an air foil in simulation, but did not adapt their model on real data.

The second application involves an *ML-driven digital twin of product properties*, which examines the ability to reconstruct 2D maps of textile material parameters using various

ML models. These textile parameters are vital for assessing the quality of the final product, and the 2D maps serve as the digital twin of the product properties. In this endeavor, we implemented sim-to-real transfer learning and conducted an experimental campaign to evaluate the effectiveness of this approach. We investigated the amount of (costly) real data required to achieve a certain level of prediction quality when using larger simulation datasets for pre-training. The best results were obtained using the top-performing model with sim-to-real transfer learning. González and Fernández-León [2020] used CNNs to detect changes in the flow front and derived changes in the permeability as well, but only in a discrete, rectangular fashion, whereas our model is more flexible in producing maps that can contain changes in various forms. Wei et al. [2016] used Darcy's law to predict permeabilities from sensor values, which can defer from reality.

Finally, we explored the potential of *controlling LCM processes using RL methods.* Opposed to Szarski and Chauhan [2023], who optimized fluid flow a priori by means of RL, we conducted experiments on *active* fluid flow control by utilizing an RL agent that actively adjusts the input pressure at different inlets. This method offers the advantage of (1) having the possibility to counteract unforeseen circumstances, i.e. react to changes in the textile and (2) learning from reward functions without relying on an explicit, mathematical environment model, and goes beyond the state-of-the-art of active controllers, that heavily rely on these mathematical models [Wang et al., 2018; Hergan et al., 2020]. Our approach can also handle high-dimensional inputs, such as images or multiple sensor signals simultaneously.

With these ML methods that show how to approach a complex industrial process family such as LCM, that are data-driven in a domain were data is not easily obtainable, one can analyze and optimize a variety of related processes.

8.2 A Short Guide on how to Proceed from here

One open point within the scope of this thesis is the application of the tools and approaches in a real-life industrial setting. This Section is directed towards researchers or engineers that want to adapt the approaches from this thesis to other setups or even other process types within the broader LCM process group. It depends fully on the type of LCM process which measures need to be taken. Closed-mold processes like RTM can be controlled with RL methods, as described in Chapter 7 with the process monitoring tooling and pipeline described in Subsection 2.3.4, based on the flow front reconstruction from Subsection 5.1.1. Also the analysis tools and digital twin from Chapter 5 and Chapter 6 could be used for a new process form. For LCM methods, that feature a visible flow front, such as VARI, the reconstruction is not necessary, but the images of the flow front could be used directly for the RL controller or the analysis of the process. For both types of LCM processes, it is crucial to reimplement the simulation and make it as close to reality as possible. One aspect of this realism would be to include race-tracking (cf. Subsection 4.5.1).

Especially when the process is initially set up, the resulting small data regime will still make sim-to-real transfer learning a valid and useful approach. For both supervised learning approaches, the adaption of the models to real data is already described in the corresponding sections. This works analogously with data from an industrial process. For the control of the process with RL, a decent approach is to train the model in simulation first, since nothing can break there and the parallelization possibilities make training feasible. When a certain quality of the controller is reached, it could be used on the real machine. At first, without training, to see how the controller performs, and then with training and adaptation of the NN's weights to investigate whether the performance can be enhanced. Starting with a low learning rate could help reducing catastrophic forgetting or overfitting on the new (real) data from the industrial process. Before letting the model work on real machinery, several safeguards should be added to avoid damages. These safeguards include constraining the allowed outputs of the controller to values that cause no harm or starting with a controller that only presents suggestions to a human worker that has to acknowledge them. The latter would cause the process to be slower, which could be manageable in processes such as VARI, that take several minutes to complete.

8.3 Outlook and Future Work

As described above, the adaptation of the proposed methods to an industrial setup is an important conceptual extension of this work. Within that measure also falls the utilization of more complex process models, i.e. 2.5- or 3D-based or with different types of (non-)permeable inlays. Another technical extension of the presented work is the prediction of fiber orientations as part of the product properties in the digital twin in Chapter 6, which is another viable quality information. These measures can be summarized as extensions or complexions of the presented methods.

Other aspects that have not yet been taken under consideration is the deployment and maintenance of such models. When deploying an ML model in an industrial setting or in production, additional aspects become important, which fall under the umbrella term *MLOps* [Kreuzberger et al., 2023]: the securing of save, reliable, and efficient deployment and maintenance of ML models. One important aspect is the detection of concept drift in new data and to adapt and retrain the models accordingly. Here it becomes necessary to regularly control the metrics achieved by the model and to retrain it on new data if necessary. To disseminate the applications in a different process within the LCM process group, as described in Section 8.2, we will apply the methods from this thesis, especially the RL method for control on the Vacuum Infusion process, including a real life machine. Here the steps described above will be followed, including the sim-to-real approach with a newly implemented simulation.

Glossary

A2C Asynchronous Advantage Actor-Critic. 41, 44, 45, 103, 108, 110, 112–115, 124

AI Artificial Intelligence. 9, 20, 26, 31

ANN Artificial Neural Network. 28, 40, 43, 44

ARX Autoregressive Model with Exogenous Input. 116

CAD Computer-aided Design. 3

CCC concordance correlation coefficient. 75

CFD Computational Fluid Dynamics. 46

CFRP Carbon Fiber Reinforced Polymer. 1, 3, 5–7, 10, 11, 38, 51, 119

CNN Convolutional Neural Network. 25, 27, 30–32, 37, 71, 74–76, 81, 83, 87, 89, 92, 93, 96–99, 103, 115, 120

ConvLSTM Convolutional Long Short Term Memory. 25, 30–32, 87, 90, 92, 96, 98

CPU Central Processing Unit. 51, 106

CV Computer Vision. 3, 20, 26, 36–38, 48, 69, 91

DEA Dielectric Analysis. 3, 20, 21

DRL Deep Reinforcement Learning. 40, 44, 101, 108, 116, 117

FAM Finite Area Method. 47

FDM Finite Difference Method. 46

FEA Finite Element Analysis. 3, 4

FEM Finite Element Method. 46, 47, 50, 53, 59, 66, 67, 69, 84, 102, 103

Ff flow front (observation). 109–112, 115, 124

FfFvc flow front and fiber volume content (observation). 109, 110, 113, 115

FfFvcP flow front, fiber volume content and pressure (observation). 109–115, 124

φ_f fiber volume content, also abbreviated as FVC. 57, 58, 60, 62–64, 88, 89, 91, 92, 125

FN false negative. 74, 76

FP false positive. 74, 76

FRP Fiber Reinforced Polymer. viii, 1–6, 9–13, 15, 18–21, 49, 57, 64, 87, 101, 102

FVC fiber volume content, also represented as φ_f. 14, 18, 48, 49, 53, 55–58, 63–65, 68, 69, 76, 77, 87, 102, 109–112, 116, 117

FVM Finite Volume Method. 46, 47

GAN Generative Adversarial Network. 72

GMT glass mat thermoplastics. 14

GPT Generative Pre-trained Transformer. 36–38

GPU Graphics Processing Unit. 36, 51, 92, 106, 125

HTTP Hypertext Transfer Protocol. 104, 107

IBE International Benchmark Exercise. 18, 56, 57

IoU Intersection over Union. 6, 87, 91–93, 95–99, 124

LCM Liquid Composite Molding. 1–7, 9–12, 14, 15, 17, 19–21, 25, 38, 45–47, 52–55, 57, 66, 71, 74, 76, 77, 81–85, 87, 96, 98, 115–117, 119–121, 123

LSTM Long Short Term Memory. 25, 27, 30–32, 36, 90, 123

MDP Markov Decision Process. 39–41, 43

ML Machine Learning. 1, 3–7, 21, 25–29, 32, 35, 37, 38, 41, 49–52, 63, 64, 72, 81, 83, 88, 89, 96, 98, 102, 119–121

MLP Multi-layer Perceptron. 75

MPC Model Predictive Control. 76, 101, 110, 116

MSE Mean Squared Error. 27, 30, 78, 79, 82, 107, 124

NCF non-crimp fabrics. 14, 15, 123

NFS Network File System. 107

NLP Natural Language Processing. 26, 32, 36–38

NN Neural Network. 5, 20, 21, 26, 28–31, 40, 44, 45, 72, 83, 84, 89, 91, 92, 106, 108, 115, 116, 121

PAN Polyacrylnitril. 13

PPO Proximal Policy Optimization. 41, 44, 45, 103, 108, 110–113, 115, 124

RANS Reynolds-averaged Navier-Stokes. 83

ReLU Rectified Linear Unit. 77, 90, 108

RENKA Regularising Ensemble Kalman filter Algorithm. 98

RL Reinforcement Learning. x, 4, 7, 15, 21, 25, 26, 38–42, 44, 45, 47, 49, 55, 63, 64, 76, 101–104, 108–110, 112, 115–117, 120, 121, 123, 124

RMSE Root Mean Squared Error. 116

RNN Recurrent Neural Network. 31, 32, 36

RTM Resin Transfer Molding. x, 2–4, 10–12, 20, 21, 23, 38, 44, 45, 47–49, 56, 59, 71, 72, 74, 77, 81, 83, 87, 88, 95, 96, 98, 101–103, 107–110, 115–117, 120, 123, 124

SA SimAniso. 93–96, 124, 125

SAC SimAnisoChannel. 93–97, 124, 125

SMC sheet molding compounds. 14

TD Temporal Difference. 42, 43, 45

TN true negative. 74, 76

TP true positive. 74, 76

TPU Tensor Processing Unit. 51

TRPO Trust Region Policy Optimization. 45

VARI Vacuum Assisted Resin Infusion. 2, 74, 120, 121

ViT Vision Transformer. 36, 37, 75, 89

List of Figures

List of Tables

Bibliography

Achzet, M., Schaller, B., Schlech, T., Linscheid, F., and Sause, M. G. R. (2023). Ultraschallbasierte Untersuchung rheologischer Eigenschaften von Polymeren für die Prozessüberwachung. In *Jahrestagung der Deutschen Gesellschaft für Zerstörungsfreie Prüfung (DGZfP)*.

Achzet, M., Schlech, T., Linscheid, F. F., Faber, J., and Sause, M. G. R. (2022). Ultraschallbasierte Prozessüberwachung am Beipiel eines T-RTM Prozesses. In *Jahrestagung der Deutschen Gesellschaft für Zerstörungsfreie Prüfung (DGZfP)*.

Agarwal, R., Gupta, V., Singh, J., and Jain, V. (2022). Prediction of surface roughness and cutting force induced during rotary ultrasonic bone drilling via statistical and machine learning algorithms. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 236(23):11123–11135.

Al Khawli, T., Anwar, M., Alzaabi, A., Sunda-Meya, A., and Islam, S. (2018). Machine Learning for Robot-Assisted Industrial Automation of Aerospace Applications. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE.

Ali, A., Zhu, Y., and Zakarya, M. (2022). Exploiting dynamic spatio-temporal graph convolutional neural networks for citywide traffic flows prediction. *Neural Networks*, 145:233–247.

Ansys Inc. (2023). Ansys Fluent. Software.

Arnold, M. (2017). *Einfluss verschiedener Angussszenarien auf den Harzinjektionsprozess und dessen simulative Abbildung.* Dissertation, Technische Universität Kaiserslautern.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine*, 34(6):26–38.

AVK, I. V. K. (2013). *Handbuch Faserverbundkunststoffe/Composites: Grundlagen, Verarbeitung, Anwendungen.* Springer eBook Collection Computer Science and Engineering. Springer Vieweg, Wiesbaden, 4 edition.

Babb, D. A., Richey, W. F., Clement, K., Peterson, E. R., Kennedy, A. P., Jezic, Z., Bratton, L. D., Lan, E., and Perettie, D. J. (1998). Resin transfer molding process for composites. Patent.

Baskaran, M., de Mendibil, I. O., Sarrionandia, M., Aurrekoetxea, J., Acosta, J., Argarate, U., and Chico, D. (2014). Manufacturing cost comparison of RTM, HP-RTM and CRTM for an automotive roof. In *Proceedings of the 16th European Conference on Composite Materials, ECCM.*

Bellman, R. (1957). A Markovian Decision Process. *Indiana University Mathematics Journal*, 6(4):679–684.

Bellman, R. E. (2003). *Dynamic Programming.* Dover Books on Computer Science Series. Dover Publications.

Belytschko, T., Liu, W. K., Moran, B., and Elkhodary, K. I. (2014). *Nonlinear finite elements for continua and structures.* Wiley, Chichester, West Sussex, United Kingdom, 2 edition.

Berg, D., Fauster, E., Abliz, D., Grössing, H., Meiners, D., Schledjewski, R., and Ziegmann, G. (2015). Influence of test rig configuration and evaluation algorithms on optical radial-flow permeability measurement: A benchmark exercise. In *20th International Conference on Composite Materials, Copenhagen, Denmark*, volume 19.

Beyrle, M., Endraß, M., Fischer, F., Kühnel, M., Schuster, A., Stefani, T., Glück, R., Braun, G., Jarka, S., Hohenreiter, M., Mayer, M., Brandt, L., Schneyer, S., Rähtz, C., Gänswürger, P., Gerngroß, T., and Kupke, M. (2017). Automated Production and Joining of High Performance Structures out of Carbon Fiber Reinforced Thermoplastics. In *CAMX 2017 - Composites and Advanced Materials Expo.*

Bezanson, J., Karpinski, S., Shah, V. B., and Edelman, A. (2012). Julia: A Fast Dynamic Language for Technical Computing. *arXiv preprint arXiv 1209.5145.*

Bickerton, S., Stadtfeld, H. C., Steiner, K. V., and Advani, S. G. (1998). Active Control of Resin Injection for the Resin Transfer Molding Process. In American Society for Composites, editor, *American Society for Composites*, pages 232–245.

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI Gym. *arXiv preprint arXiv 1606.01540.*

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. *arXiv preprint arXiv 2005.14165v4.*

Brunton, S. L. and Kutz, J. N. (2019). *Data-driven science and engineering: Machine learning, dynamical systems, and control.* Cambridge University Press, Cambridge and New York, NY and Port Melbourne and New Delhi and Singapore.

Brunton, S. L., Nathan Kutz, J., Manohar, K., Aravkin, A. Y., Morgansen, K., Klemisch, J., Goebel, N., Buttrick, J., Poskin, J., Blom-Schieber, A. W., Hogan, T., and McDonald, D. (2021). Data-Driven Aerospace Engineering: Reframing the Industry with Machine Learning. *AIAA Journal*.

Caglar, B., Broggi, G., Ali, M. A., Orgéas, L., and Michaud, V. (2022). Deep learning accelerated prediction of the permeability of fibrous microstructures. *Composites Part A: Applied Science and Manufacturing*, 158.

Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. In *2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561. Association for Computational Linguistics.

Darcy, H. (1856). *Les fontaines publiques de la ville de Dijon: Exposition at application des principes a suivre et des formules a employer dans les questions de: Distribution d'eau ouvrage terminé par un appendice relatif aux fournitures d'eau des plusieurs villes au filtrage des aux et a la fabrication des tuyeaux de fonte, de plomb, de tole et de bitume.* Victor Dalmont, Libraire des Corps imperiaux des ponts et chaussées et des mines, Paris.

Demartini, M., Galluccio, F., Mattis, P., Abusohyon, I., Lepratti, R., and Tonelli, F. (2019). Closed-Loop Manufacturing for Aerospace Industry: An Integrated PLM-MOM Solution to Support the Wing Box Assembly Process. In *APMS 2019. IFIP Advances in Information and Communication Technology*, pages 423–430. Springer, Cham.

Demirci, H. H. and Coulter, J. P. (1994a). Intelligent control of resin transfer molding processes(RTM) utilizing neural networks and nonlinear optimization methods. In Srivatsan, T. S., editor, *Processing, design, and performance of composite materials 1994*, volume 52 of *MD*, pages 13–28. American Society of Mechanical Engineers, New York, NY.

Demirci, H. H. and Coulter, J. P. (1994b). Neural network based control of molding processes. *Journal of Materials Processing and Manufacturing Science*, 2(3):335–354.

Demirci, H. H. and Coulter, J. P. (1996). A comparative study of nonlinear optimization and Taguchi methods applied to the intelligent control of manufacturing processes. *Journal of Intelligent Manufacturing*, 7(1):23–38.

Demirci, H. H., Coulter, J. P., and Güçeri, S. I. (1997). A Numerical and Experimental Investigation of Neural Network-Based Intelligent Control of Molding Processes. *Journal of Manufacturing Science and Engineering*, 119(1):88–94.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, Piscataway, NJ. IEEE.

Denkena, B., Horst, P., Schmidt, C., Behr, M., and Krieglsteiner, J. (2014). Efficient Production of CFRP Lightweight Structures on the Basis of Manufacturing Considerations at an Early Design Stage. In *New Production Technologies in Aerospace Industry*, pages 131–136. Springer, Cham.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv 1810.04805v2*.

Di Fratta, C., Klunker, F., and Ermanni, P. (2013). A methodology for flow-front estimation in LCM processes based on pressure sensors. *Composites Part A: Applied Science and Manufacturing*, 47.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Elkington, M., Bloom, D., Ward, C., Chatzimichali, A., and Potter, K. (2015). Hand layup: understanding the manual process. *Advanced Manufacturing: Polymer & Composites Science*, 1(3):138–151.

ESI Group (2023). Composites Simulation Software: PAM RTM. Software.

Faber, J., Vistein, M., Chaloupka, A., Achzet, M., Linscheid, F., Kurt, S., and Sause, M. G. R. (2021). Sensor-based process monitoring of in-situ polymerization in T-RTM manufacturing with caprolactam. In *SAMPE Europe Conference 2021 Baden/Zürich - Switzerland*.

Fauster, E., Berg, D. C., Abliz, D., Grössing, H., Meiners, D., Ziegmann, G., and Schledjewski, R. (2019). Image processing and data evaluation algorithms for reproducible optical in-plane permeability characterization by radial flow experiments. *Journal of Composite Materials*, 53(1):45–63.

Fauster, E., Berg, D. C., May, D., Blößl, Y., and Schledjewski, R. (2018). Robust Evaluation of Flow Front Data for In-Plane Permeability Characterization by Radial Flow Experiments. *Advanced Manufacturing: Polymer & Composites Science*, pages 24–40.

Fauster, E., Stieber, S., Wilfert, J., Legenstein, A., Schledjewski, R., and Reif, W. (2023). Artificial Intelligence and Optimization Techniques for LCM Process Control: A Literature Review. Submitted to Polymer Composites.

Ferziger, J. H., Perić, M., and Street, R. L. (2020). *Computational Methods for Fluid Dynamics*. Springer International Publishing, Cham.

Fitzgerald, J., Larsen, P. G., and Pierce, K. (2019). Multi-modelling and Co-simulation in the Engineering of Cyber-Physical Systems: Towards the Digital Twin. In *From Software Engineering to Formal Methods and Tools, and Back*, pages 40–55. Springer, Cham.

Fredriksson, T., Mattos, D. I., Bosch, J., and Olsson, H. H. (2020). Data Labeling: An Empirical Investigation into Industrial Challenges and Mitigation Strategies. In Morisio, M., Torchiano, M., and Jedlitschka, A., editors, *Product-Focused Software Process Improvement*, volume 12562 of *Lecture Notes in Computer Science*, pages 202–216. Springer International Publishing, Cham.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030.

Gardner, P., Fuentes, R., Dervilis, N., Mineo, C., Pierce, S. G., Cross, E. J., and Worden, K. (2020). Machine learning at the interface of structural health monitoring and non-destructive evaluation. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 378(2182).

Glaessgen, E. and Stargel, D. (2012). The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. American Institute of Aeronautics and Astronautics.

González, C. and Fernández-León, J. (2020). A Machine Learning Model to Detect Flow Disturbances during Manufacturing of Composites by Liquid Moulding. *Journal of Composites Science*, 4(2):71.

Goodfellow, I., Courville, A., and Bengio, Y. (2016). *Deep learning*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts.

Grauman, K., Westbury, A., Byrne, E., Chavis, Z., Furnari, A., Girdhar, R., Hamburger, J., Jiang, H., Liu, M., Liu, X., Martin, M., Nagarajan, T., Radosavovic, I., Ramakrishnan, S. K., Ryan, F., Sharma, J., Wray, M., Xu, M., Xu, E. Z., Zhao, C., Bansal, S., Batra, D., Cartillier, V., Crane, S., Do, T., Doulaty, M., Erapalli, A., Feichtenhofer, C., Fragomeni, A., Fu, Q., Gebreselasie, A., González, C., Hillis, J., Huang, X., Huang, Y., Jia, W., Khoo, W., Kolář, J., Kottur, S., Kumar, A., Landini, F., Li, C., Li, Y., Li, Z., Mangalam, K., Modhugu, R., Munro, J., Murrell, T., Nishiyasu, T., Price, W., Ruiz, P., Ramazanova, M., Sari, L., Somasundaram, K., Southerland, A., Sugano, Y., Tao, R., Vo, M., Wang, Y., Wu, X., Yagi, T., Zhao, Z., Zhu, Y., Arbeláez, P., Crandall, D., Damen, D., Farinella, G. M., Fuegen, C., Ghanem, B., Ithapu, V. K., Jawahar, C. V., Joo, H., Kitani, K., Li, H., Newcombe, R., Oliva, A., Park, H. S., Rehg, J. M., Sato, Y., Shi, J., Shou, M. Z., Torralba, A., Torresani, L., Yan, M., and Malik, J. (2022). Ego4D: Around the World in 3,000 Hours of Egocentric Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18995–19012.

Grondman, I., Busoniu, L., Lopes, G. A. D., and Babuska, R. (2012). A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307.

Grössing, H., Stadlmajer, N., Fauster, E., Fleischmann, M., and Schledjewski, R. (2016). Flow front advancement during composite processing: predictions from numerical filling simulation tools in comparison with real-world experiments. *Polymer Composites*, 37(9):2782–2793.

Hacotech GmbH (2023). Produkte: Composite materialien. `https://www.hacotech.com/produkte/composite-materialien`. Accessed: 2023-06-28.

Haghighat, E., Raissi, M., Moure, A., Gomez, H., and Juanes, R. (2021). A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379.

Hairer, E. and Wanner, G. (2008). *Solving ordinary differential equations*, volume 8 of *Springer series in computational mathematics*. Springer, Berlin and Heidelberg, 2. rev. ed., corr. 3. print edition.

Hammami, A. and Gebart, B. R. (2000). Analysis of the vacuum infusion molding process. *Polymer Composites*, 21(1):28–40.

Heber, L. (2022). *Optimierung des RTM-Prozesses zur CFK-Herstellung durch Bestärkendes Lernen*. Bachelorarbeit, Universität Augsburg, Augsburg.

Hergan, P., Fauster, E., Perkonigg, D., Pinter, G., and Schledjewski, R. (2020). Flow-speed-controlled quality optimisation for one-shot-hybrid RTM parts. *Advanced Manufacturing: Polymer & Composites Science*, 6(1):29–37.

Heuer, H., Schulze, M., Pooch, M., Gäbler, S., Nocke, A., Bardl, G., Cherif, C., Klein, M., Kupke, R., Vetter, R., Lenz, F., Kliem, M., Bülow, C., Goyvaerts, J., Mayer, T., and Petrenz, S. (2015). Review on quality assurance along the CFRP value chain - Non-destructive testing of fabrics, preforms and CFRP by HF radio wave techniques. *Composites Part B: Engineering*, 77:494–501.

Heywood, J., MacKenzie, D., Akerlind, Bonde, Ingrid, Bastani, P., Berry, I., Bhatt, K., Chao, A., Chow, E., Karplus, V., Keith, D., Khusid, M., Nishimura, E., and Zoepf, S. (2015). On the Road toward 2050: Report Massachusetts Institute of Technology Potential for Substantial Reductions in Light-Duty Vehicle Energy Use and Greenhouse Gas Emissions. *Massachusetts Institute of Technology*.

Higuchi, R., Aoki, R., Yokozeki, T., and Okabe, T. (2020). Evaluation of the in-situ damage and strength properties of thin-ply CFRP laminates by micro-scale finite element analysis. *Advanced Composite Materials*, 29(5):475–493.

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976. IEEE Computer Society.

Janiesch, C., Zschech, P., and Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(3):685–695.

Jiang, S., Zhang, C., and Wang, B. (2001). A process performance index and its application to optimization of the rtm process. *Polymer Composites*, 22(5):690–701.

Jiang, S., Zhang, C., and Wang, B. (2002). Optimum arrangement of gate and vent locations for RTM process design using a mesh distance-based approach. *Composites Part A: Applied Science and Manufacturing*, 33(4):471–481.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Konda, V. and Tsitsiklis, J. (1999). Actor-Critic Algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Konstantopoulos, S., Fauster, E., and Schledjewski, R. (2014). Monitoring the production of FRP composites: A review of in-line sensing methods. *Express Polymer Letters*, 8(11):823–840.

Kreuzberger, D., Kühl, N., and Hirschl, S. (2023). Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *IEEE Access*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Kuhn, T. (2017). Digitaler Zwilling. *Informatik-Spektrum*, 40(5):440–444.

Kurt, S. (2021). *Kalorimetrische und rheologische Untersuchung der anionischen Polymerisation von Epsilon-Caprolactam zu Polyamid-6*. Dissertation, Universität Augsburg.

Landi, D., Vita, A., and Germani, M. (2020). Interactive optimization of the resin transfer molding using a general-purpose tool: a case study. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 14(1):295–308.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551.

Lee, J., Ardakani, H. D., Yang, S., and Bagheri, B. (2015). Industrial Big Data Analytics and Cyber-physical Systems for Future Maintenance & Service Innovation. *Procedia CIRP*, 38:3–7.

Lekanidis, S. and Vosniakos, G. C. (2020). Machine vision support of VARI process automation in composite part manufacturing. *International Journal of Mechatronics and Manufacturing Systems*, 13(2):169.

Li, F.-F. (2023). CS231n: Deep Learning for Computer Vision. `https://cs231n.github.io/neural-networks-1/`. Accessed: 2023–08-01.

Li, L., Aslam, S., Wileman, A., and Perinpanayagam, S. (2022). Digital Twin in Aerospace Industry: A Gentle Introduction. *IEEE Access*, 10:9543–9562.

Linnainmaa, S. (1976). *Taylor Expansion of the accuumulated rounding Error*. Master Thesis, University of Helsinki, Helsinki.

Liu, B., Bickerton, S., and Advani, S. G. (1996). Modelling and simulation of resin transfer moulding (RTM) - gate control, venting and dry spot prediction. *Composites Part A: Applied Science and Manufacturing*, 27(2):135–141.

Lodes, L. (2021). *In-situ Uberwachung von Composite-Herstellungsprozessen mit tiefen neuronalen Netzen auf der Basis von 3D-Repräsentationen*. Masterarbeit, Universität Augsburg, Augsburg.

Luz, F. F., Amico, S. C., Souza, J. Á., Barbosa, E. S., and de Lima, A. G. B. (2012). Resin Transfer Molding Process: Fundamentals, Numerical Computation and Experiments. *Numerical Analysis of Heat and Mass Transfer in Porous Media. Advanced Structured Materials*, pages 121–151.

Magagnato, D., Bernath, A., and Henning, F. (2013). Experimentelle und numerische Untersuchung der Infiltration bei der RTM-Fertigung. *Tagung Verbundwerkstoffe / Werkstoffverbunde 2013*, 2013.

Matsuzaki, R., Kobayashi, S., Todoroki, A., and Mizutani, Y. (2011). Full-field monitoring of resin flow using an area-sensor array in a VaRTM process. *Composites Part A: Applied Science and Manufacturing*, 42(5):550–559.

Matsuzaki, R., Kobayashi, S., Todoroki, A., and Mizutani, Y. (2013). Flow control by progressive forecasting using numerical simulation during vacuum-assisted resin transfer molding. *Composites Part A: Applied Science and Manufacturing*, 45(2):79–87.

Matveev, M. Y., Endruweit, A., Long, A. C., Iglesias, M. A., and Tretyakov, M. V. (2021). Bayesian inversion algorithm for estimating local variations in permeability and porosity of reinforcements using experimental data. *Composites Part A: Applied Science and Manufacturing*, 143.

May, D., Aktas, A., Advani, S. G., Berg, D. C., Endruweit, A., Fauster, E., Lomov, S. V., Long, A., Mitschang, P., Abaimov, S., Abliz, D., Akhatov, I., Ali, M. A., Allen, T. D., Bickerton, S., Bodaghi, M., Caglar, B., Caglar, H., Chiminelli, A., Correia, N., Cosson, B., Danzi, M., Dittmann, J., Ermanni, P., Francucci, G., George, A., Grishaev, V., Hancioglu, M., Kabachi, M. A., Kind, K., Deléglise-Lagardère, M., Laspalas, M., Lebedev, O. V., Lizaranzu, M., Liotier, P.-J., Middendorf, P., Morán, J., Park, C.-H., Pipes, R. B., Pucci, M. F., Raynal, J., Rodriguez, E. S., Schledjewski, R., Schubnel, R., Sharp, N., Sims, G., Sozer, E. M., Sousa, P., Thomas, J., Umer, R., Wijaya, W., Willenbacher, B., Yong, A., Zaremba, S., and Ziegmann, G. (2019). In-plane permeability characterization of engineering textiles based on radial flow experiments: A benchmark exercise. *Composites Part A: Applied Science and Manufacturing*, 121:100–114.

Mendikute, J., Baskaran, M., Aretxabaleta, L., and Aurrekoetxea, J. (2022). Effect of voids on the impact properties of non-crimp fabric carbon/epoxy laminates manufactured by liquid composite moulding. *Composite Structures*, 297.

Mendikute, J., Baskaran, M., Llavori, I., Zugasti, E., Aretxabaleta, L., and Aurrekoetxea, J. (2023). Predicting the effect of voids generated during rtm on the low-velocity impact behaviour by machine learning-based surrogate models. *Composites Part B: Engineering*, 260.

Mendikute, J., Plazaola, J., Baskaran, M., Zugasti, E., Aretxabaleta, L., and Aurrekoetxea, J. (2021). Impregnation quality diagnosis in Resin Transfer Moulding by machine learning. *Composites Part B: Engineering*, 221.

Miyano, Y., Nakada, M., Kudoh, H., and Muki, R. (2000). Prediction of Tensile Fatigue Life for Unidirectional CFRP. *Journal of Composite Materials*, 34(7):538–550.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. In *International conference on machine learning*, pages 1928–1937. PMLR.

Mostowski, A. (1953). A. A. Markov. Téoriá algorifmov (The theory of algorithms). Trudy Matématičéskogo Institute iméni V. A. Stéklova, vol. 38 (1951), pp. 176–189. *Journal of Symbolic Logic*, 18(4):340–341.

Mourtzis, D., Vlachou, E., and Milas, N. (2016). Industrial Big Data as a Result of IoT Adoption in Manufacturing. *Procedia CIRP*, 55:290–295.

Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., Maria, A. D., Panneershelvam, V., Suleyman, M., Beattie, C., Petersen, S., Legg, S., Mnih, V., Kavukcuoglu, K., and Silver, D. (2015). Massively Parallel Methods for Deep Reinforcement Learning. *arXiv preprint arXiv: 1507.04296v2*.

Neitzel, M., Mitschang, P., and Breuer, U. (2014). *Handbuch Verbundwerkstoffe: Werkstoffe, Verarbeitung, Anwendung*. Hanser eLibrary. Hanser, München, 2., aktualisierte und erweiterte auflage edition.

Nelder, J. A. and Mead, R. (1965). A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313.

Nielsen, D. and Pitchumani, R. (2000). Real time model-predictive control of preform permeation in liquid composite molding processes. In *Proceedings of NHTC'00*.

Nielsen, D. and Pitchumani, R. (2001). Intelligent model-based control of preform permeation in liquid composite molding processes, with online optimization. *Composites Part A: Applied Science and Manufacturing*, 32(12):1789–1803.

Nielsen, D. and Pitchumani, R. (2002a). Closed-loop flow control in resin transfer molding using real-time numerical process simulations. *Composites Science and Technology*, 62(2):283–298.

Nielsen, D. R. and Pitchumani, R. (2002b). Control of flow in resin transfer molding with real-time preform permeability estimation. *Polymer Composites*, 23(6):1087–1110.

Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, pages 1520–1528.

Nowoczyn, N. J. (2021). *Optimierung der Fließfrontberechnung im Harzinjektionsprozess mithilfe von GANs*. Bachelorarbeit, Universität Augsburg, Augsburg.

Obertscheider, C. and Fauster, E. (2023). RTMsim - A Julia module for filling simulations in Resin Transfer Moulding with the Finite Area Method. *Journal of Open Source Software*, 8(84):4763.

Obertscheider, C., Fauster, E., and Stieber, S. (2023). Experimental validation of a new adaptable LCM mould filling software. Submitted to Advanced Manufacturing: Polymer & Composites Science Journal.

Ocampo, J. D., Crosby, N., and Millwater, H. R. (2017). Probabilistic Damage Tolerance for Aviation Fleets Using a Kriging Surrogate Model. In *19th AIAA Non-Deterministic Approaches Conference*, Reston, Virginia. American Institute of Aeronautics and Astronautics.

Ogunleye, R. O., Rusnakova, S., Zaludek, M., and Emebu, S. (2022). The Influence of Ply Stacking Sequence on Mechanical Properties of Carbon/Epoxy Composite Laminates. *Polymers*, 14(24):5566.

Okabe, T., Oya, Y., Yamamoto, G., Sato, J., Matsumiya, T., Matsuzaki, R., Yashiro, S., and Obayashi, S. (2017). Multi-objective optimization for resin transfer molding process. *Composites Part A: Applied Science and Manufacturing*, 92.

OpenAI (2023). GPT-4 Technical Report. *arXiv preprint arXiv 2303.08774v3.*

Oya, Y., Matsumiya, T., Ito, A., Matsuzaki, R., and Okabe, T. (2020). Gate optimization for resin transfer molding in dual-scale porous media: Numerical simulation and experiment measurement. *Journal of Composite Materials*, 54(16):2131–2145.

Oyekan, J., Farnsworth, M., Hutabarat, W., Miller, D., and Tiwari, A. (2020). Applying a 6 DoF Robotic Arm and Digital Twin to Automate Fan-Blade Reconditioning for Aerospace Maintenance, Repair, and Overhaul. *Sensors (Basel, Switzerland)*, 20(16).

Pantelelis, N. and Bistekos, E. (2010). Process monitoring and control for the production of cfrp components. In *Proceedings of the SAMPE-10 Conference, Seattle, WA, USA*, pages 17–20.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. (2021). Learning Mesh-Based Simulation with Graph Networks. In *International Conference on Learning Representations*.

Qi, Z., Liu, Y., and Chen, W. (2019). An approach to predict the mechanical properties of CFRP based on cross-scale simulation. *Composite Structures*, 210:339–347.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268).

Rajak, D. K., Wagh, P. H., and Linul, E. (2021). Manufacturing Technologies of Carbon/Glass Fiber-Reinforced Polymer Composites and Their Properties: A Review. *Polymers*, 13(21).

Rodrigues de Oliveira, I., Campos Amico, S., Ávila Souza, J., Ferreira Luz F., Barcella, R., and Gilson Barbosa de Lima, A. (2013). Resin Transfer Molding Process: A Numerical Investigation. *Defect and Diffusion Forum Vols. 334-335 (2013) pp 193-198*, 2013.

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*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer International Publishing, Cham.

Rosenberg, P. (2018). *Entwicklung einer RTM Prozessvariante zur kavitätsdruckgeregelten Herstellung von Faserverbundstrukturbauteilen.* Dissertation, Karlsruher Institut für Technologie (KIT).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Schmidt, T., May, D., Duhovic, M., Widera, A., Hümbert, M., and Mitschang, P. (2021). A combined experimental–numerical approach for permeability characterization of engineering textiles. *Polymer Composites*, 42(7):3363–3379.

Schulman, J. (2016). *Optimizing Expectations: From Deep Reinforcement Learning to Stochastic Computation Graphs.* Dissertation, UC Berkeley, Berkeley.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust Region Policy Optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint arxiv 1707.06347v2.*

Senn-Kalb, L. and Mehta, D. (2023). Artificial Intelligence: in-depth market analysis: Market Insights Report.

Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. (2016). Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 1874–1883.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and Woo, W.-c. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *arXiv preprint arXiv 1506.04214v2.*

Sigl, M. E., Bachmann, A., Mair, T., and Zaeh, M. F. (2020). Torque-Based Temperature Control in Friction Stir Welding by Using a Digital Twin. *Metals*, 10(7):914.

Silver, D., Singh, S., Precup, D., and Sutton, R. S. (2021). Reward is enough. *Artificial Intelligence*, 299.

Simacek, P., Advani, S., and Binetruy, C. (2004). Liquid injection molding simulation (LIMS) a comprehensive tool to design, optimize and control the filling process in liquid composite molding. *JEC-Compos*, 8:58–61.

Singh, M., Fuenmayor, E., Hinchy, E., Qiao, Y., Murray, N., and Devine, D. (2021). Digital Twin: Origin to Future. *Applied System Innovation*, 4(2):36.

Slattery, J. C. (1981). *Momentum, energy, and mass transfer in continua.* Krieger Publ. Co, Huntington, New York, 2 edition.

Sobotta, M. (2022). *Verknüpfung von Realität und Simulation zur Analyse von Harzinjektionsprozessen.* Masterarbeit, Universität Augsburg, Augsburg.

Sorg, C. (2014). *Data Mining als Methode zur Industrialisierung und Qualifizierung neuer Fertigungsprozesse für CFK-Bauteile in automobiler Großserienproduktion.* Dissertation, Technische Universität München, München.

Stieber, S. (2018). Transfer Learning for Optimization of Carbon Fiber Reinforced Polymer Production. *Organic Computing: Doctoral Dissertation Colloquium 2018*, pages 61–73.

Stieber, S., Heber, L., Obertscheider, C., and Reif, W. (2023a). Control of Composite Manufacturing Processes through Deep Reinforcement Learning. Accepted at the International Conference on Machine Learning and Applications.

Stieber, S., Hoffmann, A., Schiendorfer, A., Reif, W., Beyrle, M., Faber, J., Richter, M., and Sause, M. (2020). Towards Real-time Process Monitoring and Machine Learning for Manufacturing Composite Structures. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1455–1458. IEEE.

Stieber, S., Schröter, N., Fauster, E., Bender, M., Schiendorfer, A., and Reif, W. (2023b). Inferring material properties from FRP processes via Sim-to-Real learning. *International Journal on Advanced Manufacturing*, pages 1517–1533.

Stieber, S., Schröter, N., Fauster, E., Schiendorfer, A., and Reif, W. (2021a). PermeabilityNets: Comparing Neural Network Architectures on a Sequence-to-Instance Task in CFRP Manufacturing. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 694–697. IEEE.

Stieber, S., Schröter, N., Schiendorfer, A., Hoffmann, A., and Reif, W. (2021b). FlowFrontNet: Improving Carbon Composite Manufacturing with CNNs. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12460 LNAI, pages 411–426, Cham. Springer International Publishing.

Sundin, E. and Björkman, M. (2016). Development of a Design for Manufacturing and Assembly (DFM/A) methodology concerning products and components made in composites of Carbon Fiber Reinforced Plastics (CFRP) used in the Aerospace Industry. In *Swedish Production Symposium.*

Sutton, R. S. and Barto, A. G. (1998). Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*, 9(5):1054.

Szarski, M. and Chauhan, S. (2021). Composite temperature profile and tooling optimization via Deep Reinforcement Learning. *Composites Part A: Applied Science and Manufacturing*, 142.

Szarski, M. and Chauhan, S. (2023). Instant flow distribution network optimization in liquid composite molding using deep reinforcement learning. *Journal of Intelligent Manufacturing*, 34(1):197–218.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A Survey on Deep Transfer Learning. In *Artificial Neural Networks and Machine Learning – ICANN*, pages 270–279. Springer, Cham.

Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., and Sui, F. (2018). Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, 94(9-12):3563–3576.

Tercan, H., Guajardo, A., Heinisch, J., Thiele, T., Hopmann, C., and Meisen, T. (2018). Transfer-Learning: Bridging the Gap between Real and Simulation Data for Machine Learning in Injection Molding. *Procedia CIRP*, 72:185–190.

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.

Tifkitsis, K. I. and Skordos, A. A. (2020). Real time uncertainty estimation in filling stage of resin transfer molding process. *Polymer Composites*, 41(12):5387–5402.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE.

Toepker, J. (2011). Einsatz von CFK im Fahrzeugbau als Befähiger der Elektromobilität. In *17. Nationales Symposium SAMPE Deutschland (Aachen)*.

Towsyfan, H., Biguri, A., Boardman, R., and Blumensath, T. (2020). Successes and challenges in non-destructive testing of aircraft composite structures. *1000-9361*, 33(3):771–791.

Travnik, J. B., Mathewson, K. W., Sutton, R. S., and Pilarski, P. M. (2018). Reactive Reinforcement Learning in Asynchronous Environments. *Frontiers in robotics and AI*, 5:79.

Trochu, F., Gauvin, R., and Gao, D.-M. (1993). Numerical analysis of the resin transfer molding process by the finite element method. *Advances in Polymer Technology*, 12(4):329–342.

Trusova, K. (2020). Comparison of Two Classification Machine Learning Models of Avionics Systems for Health Analysis. In *2020 IEEE International Conference on Electrical Engineering and Photonics (EExPolytech)*, pages 172–175. IEEE.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention Is All You Need. *Advances in neural information processing systems*, 30.

Verrey, J., Wakeman, M. D., Michaud, V., and Månson, J.-A. (2006). Manufacturing cost comparison of thermoplastic and thermoset RTM for an automotive floor pan. *Composites Part A: Applied Science and Manufacturing*, 37(1):9–22.

Versteeg, H. K. and Malalasekera, W. (2007). *An introduction to computational fluid dynamics: The finite volume method.* Pearson Education Ltd, Harlow England and New York, 2 edition.

Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational intelligence and neuroscience*, 2018.

Wang, K.-H., Chuang, Y.-C., Chiu, T.-H., and Yao, Y. (2018). Flow pattern control in resin transfer molding using a model predictive control strategy. *Polymer Engineering & Science*, 58(9):1659–1665.

Wei, B.-J., Chang, Y.-S., Yao, Y., and Fang, J. (2016). Online estimation and monitoring of local permeability in resin transfer molding. *Polymer Composites*, 37(4):1249–1258.

Wei, Z., Zhang, S., Jafari, S., and Nikolaidis, T. (2020). Gas turbine aero-engines real time on-board modelling: A review, research challenges, and exploring the future. *Progress in Aerospace Sciences*, 121.

Weiler, M. and Cesa, G. (2019). General E(2) - Equivariant steerable CNNs. *Advances in Neural Information Processing Systems*, 32(2).

Weller, H. G., Tabor, G., Jasak, H., and Fureby, C. (1998). A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12(6):620.

Wilfert, J. (2022). Reinforcement Learning für Resin Transfer Moulding. *Seminararbeit, Seminar für Software und Systems Engineering, Universität Augsburg.*

Wilfert, J., Stieber, S., Wilhelm, F., and Reif, W. (2021). Genetic Programming for Fiber-Threading for Fiber-Reinforced Plastics. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE.

Xu, L., Ren, J. S. J., Liu, C., and Jia, J. (2014). Deep convolutional neural network for image deconvolution. *Advances in Neural Information Processing Systems*, 2(January):1790–1798.

Yong, A., Aktas, A., May, D., Endruweit, A., Advani, S., Hubert, P., Abaimov, S., Abliz, D., Akhatov, I., Ali, M., et al. (2021). Out-of-plane permeability measurement for reinforcement textiles: A benchmark exercise. *Composites Part A: Applied Science and Manufacturing*, 148.

Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., and Sang, N. (2018). Learning a Discriminative Feature Network for Semantic Segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE.

Zambal, S., Eitzinger, C., Clarke, M., Klintworth, J., and Mechin, P.-Y. (2018). A digital twin for composite parts manufacturing : Effects of defects analysis based on manufacturing data. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pages 803–808. IEEE.

Zhang, H., Wang, Y., Dayoub, F., and Sunderhauf, N. (2021). VarifocalNet: An IoU-Aware Dense Object Detector. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 8514-8523*, 2021.

Zhang, J. and Pantelelis, N. G. (2011). Modelling and optimisation control of polymer composite moulding processes using bootstrap aggregated neural network models. *2011 International Conference on Electric Information and Control Engineering, ICEICE 2011 - Proceedings*, 1(2).

Zhang, J., Yang, H., Li, X., and Ye, W. (2020). A Method of Reducing Motor Vibration: Natural Frequency, Damping Ratio, and Vibration Analysis of CFRP Motor Frame. *Shock and Vibration*, 2020.

Zhang, P., Xiong, L., Yu, Z., Fang, P., Yan, S., Yao, J., and Zhou, Y. (2019). Reinforcement Learning-Based End-to-End Parking for Automatic Parking System. *Sensors (Basel, Switzerland)*, 19(18).

Zhao, W., Queralta, J. P., and Westerlund, T. (2020). Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE.

Zhu, Z., Lin, K., Jain, A. K., and Zhou, J. (2023). Transfer Learning in Deep Reinforcement Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Zhuang, C., Gong, J., and Liu, J. (2021). Digital twin-based assembly data management and process traceability for complex products. *Journal of Manufacturing Systems*, 58:118–131.