University of Žilina
Faculty of Management Science and Informatics

Blood Diagnostics Using Machine Learning and Simulations

Dissertation thesis

2024

Mgr. Samuel Molčan

University of Žilina

Faculty of Management Science and Informatics

Blood Diagnostics Using Machine Learning and Simulations

Dissertation thesis

28360020243002

Field of study:	Applied Informatics
Department:	Faculty of Management Science and Informatics
Supervisor:	doc. RNDr. Katarína Bachratá, PhD.

Žilina, 2024

Mgr. Samuel Molčan

Acknowledgment:

Many thanks to my supervisor, doc. RNDr. Katarína Bachratá, PhD., for her professional guidance and patience during the writing of this diploma thesis. At the same time, I thank my research group for the hours spent on joint research.

Thanks to my family. Thank you my love. Well, thank you all.

Abstract

The thesis explores innovative methods for diagnosing blood-related disorders using a combination of machine learning techniques and simulations, particularly focusing on the analysis of red blood cells through the ESPResSo simulation module. This approach enables a deep dive into understanding red blood cell behavior under various physiological conditions by generating numerical outputs from simulations. The goal is to employ machine learning models, notably neural networks and random forests, to analyze these outputs for classifying RBCs based on changes in their elasticity, which can indicate the presence of blood disorders. This novel methodology seeks to bridge the gap between traditional diagnostic techniques and the vast potential of computational analysis, offering a more precise and efficient diagnostic tool.

The thesis's core goal is to extend beyond numerical data analysis by transforming the simulation outputs into dynamic visual representations. This involves generating videos to capture the behaviors of red blood cells in blood flow, further processed by machine learning models for classification purposes. This dual approach, encompassing both numerical and visual data, aims to enhance the diagnostic process by providing a comprehensive view of RBC dynamics. By leveraging advanced machine learning techniques, the research promises to significantly improve the accuracy of blood diagnostics, facilitating early detection and treatment of disorders.

The final goal of the thesis revolves around designing and training a machine learning model capable of classifying red blood cells with high efficiency, using video data derived from simulations. This involves employing deep learning strategies to teach the model to recognize complex patterns in red blood cell behavior, laying the groundwork for developing a robust diagnostic framework. Through this innovative integration of computational simulations, visual analysis, and artificial intelligence, the thesis contributes to the advancement of blood diagnostics, opening the door for more personalized and precise healthcare solutions in treating blood-related disorders.

Keywords: neural networks, simulations, red blood cell elasticity

Abstrakt

Práca skúma inovatívne metódy diagnostiky porúch súvisiacich s krvou pomocou kombinácie techník strojového učenia a simulácií, najmä so zameraním na analýzu červených krviniek prostredníctvom simulačného modulu ESPResSo. Tento prístup umožňuje hlboký ponor do pochopenia správania červených krviniek za rôznych fyziologických podmienok generovaním numerických výstupov zo simulácií. Cieľom je využiť modely strojového učenia, najmä neurónové siete a náhodné lesy, na analýzu týchto výstupov na klasifikáciu červených krviniek na základe zmien v ich elasticite, čo môže naznačovať prítomnosť porúch krvi. Táto nová metodológia sa snaží preklenúť priepasť medzi tradičnými diagnostickými technikami a obrovským potenciálom výpočtovej analýzy a ponúka presnejší a efektívnejší diagnostický nástroj.

Hlavným cieľom práce je presahovať rámec numerickej analýzy údajov transformáciou výstupov simulácie do dynamických vizuálnych reprezentácií. Zahŕňa to generovanie videí na zachytenie správania červených krviniek v toku krvi, ktoré sa ďalej spracúvajú modelmi strojového učenia na účely klasifikácie. Tento duálny prístup, ktorý zahŕňa numerické aj vizuálne údaje, má za cieľ zlepšiť diagnostický proces poskytnutím komplexného pohľadu na dynamiku červených krviniek. Využitím pokročilých techník strojového učenia výskum sľubuje výrazné zlepšenie presnosti krvnej diagnostiky, uľahčenie včasnej detekcie a liečby porúch.

Konečný cieľ práce je návrh a trénovanie modelu strojového učenia schopného klasifikovať červené krvinky s vysokou účinnosťou pomocou video údajov odvodených zo simulácií. To zahŕňa použitie stratégií hlbokého učenia, aby sa model naučil rozpoznávať zložité vzorce správania červených krviniek, čím sa položili základy pre vývoj robustného diagnostického rámca. Prostredníctvom tejto inovatívnej integrácie výpočtových simulácií, vizuálnej analýzy a umelej inteligencie práca prispieva k pokroku v diagnostike krvi a otvára dvere pre personalizovanejšie a presnejšie riešenia zdravotnej starostlivosti pri liečbe porúch súvisiacich s krvou.

Kľúčové slová: neurónové siete, simulácie, elasticita červených krviniek

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Microfluidics

Microfluidics studies the behavior of fluids within small channels and the peculiar fabrication of microfluidic devices. These devices, ranging from chambers to pathways, facilitate the controlled flow of fluids [73]. This technology has left an indelible mark on molecular biology influencing areas such as enzymatic analysis [79], DNA research [139], chemical synthesis [38], and protein study [128]. Beyond the limits of laboratories, microfluidic devices have found applications in clinical pathology, swiftly diagnosing diseases and measuring their severity [28]. Notably, they've been instrumental in assessing antibiotic resistance by monitoring microorganism growth rates [98].

Moreover, microfluidics has emerged as a cornerstone in the point-of-care diagnostics. Portable microfluidic devices enable rapid and decentralized testing, enhancing healthcare accessibility in resource-limited settings [105]. This portability, coupled with the ability to process small sample volumes, positions microfluidic technologies as invaluable tools in the fight against infectious diseases, offering timely and accurate diagnostic outcomes.

Microchips, the building blocks of this technology, provide a heterogeneous environment conducive to the success of microorganisms. This application extends beyond biomedicine, where microfluidics takes on a pivotal role in continuous air and water quality monitoring. Serving as an early warning system for toxins, microfluidic devices contribute to environmental safety [96]. The versatility of microfluidics makes it a transformative force, merging into diverse scientific domains and offering solutions that transcend traditional boundaries.

Furthermore, the impact of microfluidics extends to drug discovery and development. Microfluidic platforms play a crucial role in miniaturized tests for drug screening, allowing for more efficient and cost-effective testing of various compounds [7]. Additionally, the controlled microenvironments provided by microfluidic systems enable precise studies of cell behavior and responses to different drugs, anabling advancements in personalized medicine [23]. As the field of microfluidics continues to evolve, its interdisciplinary applications across medicine, environmental monitoring, and pharmaceuticals highlight its significance in advancing scientific research and technological innovation.

1.2 Motivation

In the vast realm of microfluidics, Cell in Fluid, Biomolecular Modeling & Computational Group (CIF) [2], a dedicated research team situated within the Department of Software Technologies at Žilina University's Faculty of Management and Informatics, studies microfluidic devices designed for the capture and sorting of cancer cells from blood samples. Their focus resides in the transformative potential of these devices, particularly in the context of early cancer diagnosis through a straightforward blood sample. The big challenge of metastatic disease in oncology, where tumors seemlessly spread through blood-borne routes, underscores the urgency of their research [75].

Micro-metastatic disease, characterized by the presence of minuscule cancerous cells in critical sites like bone marrow, blood, or lymph nodes, impacts a substantial portion—around 30% to 40%—of solid tumor patients. Herein lies the potential significance of circulating tumor cells (CTCs), tiny entities that may hold the key to unlocking early detection strategies and consequently improving treatment outcomes [66]. The CIF research team's efforts are directed towards the development and optimization of microfluidic devices precisely tailored to capture and analyze these elusive CTCs.

Beyond the singular focus on cancer cell capture, the CIF team dives into the broader exploration of microfluidic phenomena. Their research extends into understanding the complicated impact of microfluidic channels on red blood cells, an essential consideration in the design of efficient and biocompatible devices [62]. Additionally, the collective behavior of cells within the dynamic flow of blood becomes a subject of investigation, offering insights into the complex interplay of cells in these microenvironments. This multifaceted approach positions CIF's efforts not only at the forefront of cancer diagnostics but also as contributors to the broader understanding of microfluidic dynamics within the context of blood-based applications.

In the microfluidic device development, testing physical prototypes presents hard challenges, both in terms of financial investment and time consumption. Recognizing these problems, researchers increasingly turn to numerical models as a pragmatic alternative. Numerical models excel at simulating fluid flow within microfluidic devices, offering a cost-effective means to understand cell behavior under diverse conditions [126].

The computational approach brings notable advantages. It allows for the assessment of device functionality without the significant costs associated with physical prototypes [39]. By employing mathematical representations and algorithms, researchers can emulate complex fluid dynamics within these devices, exploring cell interactions, sorting mechanisms, and other critical functionalities.

However, it's important to acknowledge the challenges inherent in numerical simulations. The computational demands can be substantial due to the complex nature of microfluidic devices and the vast number of modeled cells. Simulating extended experiment durations significantly increases the computational workload, often leaving a big portion of the data untapped.

To unlock the full potential of numerical simulation data, researchers are turning to machine learning. Machine learning algorithms excel at processing extensive datasets, proving invaluable given the huge volume of data generated during preprocessing and simulation. These algorithms not only extract deeper insights from existing data but also accelerate simulations by employing more efficient models. Furthermore, machine learning reveals the limitations of current numerical models, essential for refining simulation accuracy and reliability [5].

The fusion of numerical simulations and machine learning represents a synergistic approach to address microfluidic device design complexity and analyze cell behavior. It optimizes devices by leveraging the advantages of both computational modeling and data-driven artificial intelligence, reducing the financial and temporal costs associated with physical fabrication and experimentation. This integrated approach not only promises cost-efficiency but also unlocks insights from the abundant data generated during the research process.

1.3 Thesis goal

The primary objective of this thesis is to employ the ESPResSo simulation module to model the complex dynamics of red blood cells (RBCs) within the bloodstream. By utilizing sophisticated computational simulations, we aim to gain a comprehensive understanding of the behavior of RBCs under flow conditions. This simulation-based approach will generate numerical outputs that contains information about the cells' responses to different physiological conditions, allowing us to perform machine learning analyses.

The first major goal is to develop and implement machine learning models to classify RBCs based only on the numerical outputs obtained from the ESPResSo simulations which implicitly contain physical information to confirm the hypothesis that elasticity changes can be observed in simulations. Through this phase, we aim to create a classifier that can recognize small variations in red blood cell behavior, contributing to the development of diagnostic tools for blood-related disorders. This initial stage serves as a critical stepping stone, providing insights into the efficiency of numerical data in distinguishing between normal and sick RBC dynamics.

Performing in-depth analyses on the numerical outputs obtained from the ESPResSo simulations is our next goal. This aims to extract hidden patterns and relationships within the numerical data, providing a complementary perspective to the neural network-based classification. [1]

Based on numerical analysis, the subsequent goal involves transforming the simulation outputs into dynamic visual representations. Through video generation, we aim to capture the behavioral patterns exhibited by RBCs in the flow of blood. The following step is the development of a machine learning model capable of classifying RBCs based on their observed behavior in these videos. This approach seeks to enhance the diagnostic capabilities, as it incorporates both numerical and visual information for more comprehensive analysis.

The final goal of the thesis is to design and train a machine learning model capable of effectively classifying red blood cells using video data. Using deep learning techniques, we try to teach the model to recognize complex patterns within the dynamic behavior of red blood cells. This phase is critical to creating a diagnostic framework that harnesses the power of numerical simulations and visual cues, contributing to the advancement of blood diagnostics.

1.4 Thesis overview

This thesis is structured as follows:

- In Chapter 2 we provide a biological overview of red blood cell functionalities and disorders which underscore their significance in human microcirculation, with a focus on their deformability, crucial for oxygen delivery. The study explores techniques like microcapillary flow, microfluidics, and ektacytometry to characterize RBC biomechanical properties. Specific disorders affecting RBCs, such as hereditary and metabolic disorders, are discussed, highlighting tools like ektacytometry and AFM for assessment. Membrane changes and oxidative stress are explored. This chapter underscorse the significance of blood research and justify the whole thesis.

[1] By combining the strengths of multiple machine learning models, we aim to enhance the robustness and interpretability of our diagnostic framework for blood-related disorders, ultimately contributing to the evolution of precision medicine.

- Chapter 3 explores state-of-the-art in red blod cell research and it serves as an overview of the problematic. The impact of neural networks in medical diagnostics, focusing on their application in red blood cell classification using numerical data from flow cytometry and information extracted from video sequences is discussed. We also talk about challenges in incorporating noisy data into machine learning algorithms for simulating physical problems. The principles of Physics-Informed Learning are outlined, involving observational, inductive, and learning influences to guide the learning process toward identifying physically consistent solutions. The chapter concludes with related work, highlighting experiments with Physics-Informed Neural Networks in solving differential equations for a class of problems and a differentiable simulation approach, demonstrating promising results. All of this knowledge is then later used within next chapters.

- Chapter 4 looks into the dynamics of blood flow, introducing the fundamentals of Computational Fluid Dynamics (CDF), describing the mathematical models, geometric modeling, discretization, and numerical methods used to simulate fluid behavior.The applications of CFD in blood flow simulations are sampled, showing its potential in patient-specific simulations, treatment planning, and clinical applications. The chapter discusses the challenges in CFD simulations, such as model validation, computational resource demands, and the modeling of elastic properties of red blood cells.

  The ESPResSo Simulation Tool, a software designed for simulating blood flow dynamics, is introduced, emphasizing its role in personalized medicine and treatment planning. The chapter provides a detailed overview of the general description of the current model, explaining the Lattice-Boltzmann method, RBC modeling, and interactions between fluid and solid components. Model validation and optimization are also discussed. This chapter is particularly important since in the thesis we use simulation data for the analyses and for the training of machine learning models.

- As we use Neural Networks and Random Forest machine learning models, both are described in Chapter 5, emphasizing their foundational role in machine learning and artificial intelligence. It dives into NN components, architecture, learning processes, depth, complexity, and applications. Furthermore, it introduces Recurrent Neural Networks for sequential data processing, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures, Convolutional Neural Networks (CNNs), specialized for grid-like data, detailing their key components, characteristics, applications. The combination of Recurrent Neural Networks and Convolutional Neural Networks in hybrid models enables a complex understanding of data, particularly valuable in image captioning and video analysis.

- Chapter 6 research the determination of red blood cell elasticity through the application of neural networks and numerical data derived from simulations. This chapter contains a detailed description of the utilized data, followed by an evaluation of the classification model with respect to various metrics.

  The study evaluates the predictive ability of LSTM and CNN-LSTM networks for RBC elasticity, highlighting the superior performance of networks utilizing full 3D information compared to 2D projections. The CNN-LSTM architecture, employing 2D convolutional layers, emerges as the top-performing model, showcasing its effectiveness in elasticity prediction and recommending the use of devices capturing progress in linearly independent planes for accurate video recording in biological experiments.

- Chapter 7 details the machine learning models designed for the classification of red blood cells in canal with obsticles. Similar to the previous chapter, it includes a description of the employed data, various models with their parameters, and an assessment of the classification outcomes.

  We explored the classification of red blood cells elasticity through video recordings and CNNs, overcoming limited blood flow data availability by utilizing computer simulations based on a numerical model. The simulation model effectively replicated the behavior of RBCs as elastic entities in fluid flow, producing sample training data for CNN-based classification. Employing various CNN architectures, namely ResNet and EfficientNet, we conducted a classification of RBC elasticity based on the geometric features extracted from video recordings.

  Subsequently, we applied statistical analyses and Random Forest model to the raw numerical output data from simulations.

By adopting this organizational framework, we aim to systematically present our research methodology, numerical models, and the outcomes of our investigations in the domain of red blood cell characterization and classification.

Chapter 2

Biological Rationale for Blood Analysis

Red blood cells (RBCs) play a critical role in human microcirculation, exhibiting a unique ability to deform and navigate through narrow capillaries. They serve as carriers of gases between blood and tissues. Changes in RBC deformability are linked to various diseases, underscoring the importance of measuring RBC biomechanical properties. In the work by Tomaiuolo et al. [121], specific focus is placed on pathological conditions affecting RBCs, such as hereditary disorders, metabolic disorders, and oxidative stress. The study highlights challenges in expressing pathological changes using well-defined cell membrane parameters like elasticity and viscosity. Moreover, it emphasizes the potential of RBC deformability as a biomarker for specific clinical conditions.

The key biomechanical property of healthy RBCs is their deformability, crucial for delivering oxygen to tissues [125]. RBC deformability arises from factors like the cell's biconcave disc shape, intracellular fluid viscosity dominated by hemoglobin, and the viscoelastic properties of the cell membrane. Membrane behavior is described through parameters like shear elastic modulus, area compressibility modulus, and bending modulus.

Traditional methods for assessing membrane properties affecting RBC deformability are limited in terms of throughput and are static in nature. This may not accurately replicate the mechanical stresses that RBCs experience during microcirculation. However, microcapillary flow techniques and microfluidic devices have been developed to overcome these limitations, allowing for high-throughput analysis under flow conditions that are physiologically relevant. By integrating automated image analysis, microfluidic devices enable point-of-care applications, offering insights into cell deformation under various conditions.

Different techniques yield different results, emphasizing the nonlinear mechanical response of RBCs. This review categorizes eight techniques for measuring RBC biomechanical properties, including micropipette aspiration, flickering analysis, viscometry, microcapillary flow/microfluidics, ektacytometry, atomic force microscopy (AFM), optical tweezers, and others. These techniques include innovative approaches like reflection interference contrast micrograph and microscopic holography, providing a comprehensive overview of methods used for biomechanical characterization.

2.1 Hereditary Disorders

Hereditary disorders impacting the erythrocyte membrane, like spherocytosis and elliptocytosis, significantly contribute to inherited hemolytic anemias. These disorders result from deficiencies or dysfunctions in various erythrocyte membrane proteins. Ektacytometry, a common tool for evaluating RBC membrane properties, indicates reduced membrane deformability in hereditary disorders, leading to shifts in the deformability index curve relative to control. Advanced techniques, such as AFM, reveal changes in mechanical properties, providing insights into cytoskeletal alterations. Osmotic gradient ektacytometry remains crucial for diagnosis, offering insights into membrane moduli and deformability.

2.2 Metabolic Disorders

Metabolic disorders, such as diabetes and hypercholesterolemia, significantly impact cellular metabolism, causing disruptions in essential biochemical reactions. Diabetes, characterized by hyperglycemia, exhibits associations with altered blood viscosity and abnormal RBC membrane architecture, leading to impaired deformability. Recent advancements in microfluidic devices and ektacytometry confirm impaired RBC deformability in diabetic patients, emphasizing critical implications for tissue perfusion and complications like microvascular disease. Hypercholesterolemia and obesity also show varying results in RBC deformability studies, warranting further investigation into their correlations with cardiovascular risk factors.

2.3 Membrane Changes and Oxidative Stress

Oxidative stress poses challenges to RBCs due to continuous exposure to reactive oxygen molecules. Under normal conditions, RBCs employ antioxidants to counteract oxidative stress. Abnormal responses can lead to peroxide and free radical production, resulting in damages to the RBC skeleton. Techniques like ektacytometry and microfluidic approaches have been employed to measure impaired RBC deformability in the presence of oxidative stress. These advancements provide insights into changes induced by oxidative stress, offering a cost-effective and high-throughput method for assessment.

2.4 Paroxysmal Nocturnal Hemoglobinuria

Paroxysmal Nocturnal Hemoglobinuria (PHN), a rare acquired clonal disorder, is characterized by a mutation leading to the absence of protective proteins on the RBC membrane. Despite the importance of membrane viscoelastic properties in RBC survival, limited studies have explored RBC membrane properties in PNH. Micropipette techniques have been utilized to measure the mechanical properties of RBCs affected by PNH, revealing impaired mechanical properties compared to normal cells. Further research into the membrane properties of PNH erythrocytes could provide valuable insights into complement-mediated hemolysis mechanisms, contributing to a better understanding of this rare disorder.

2.5 Sickle Cell Disease

Sickle cell disease (SCD) encompasses a collection of hereditary blood disorders that impact the structure and functionality of RBCs. Ordinarily, RBCs possess a pliable and disc-like form, enabling them to effortlessly navigate narrow blood vessels and distribute oxygen across the entire body. Nevertheless, in the case of SCD, a genetic mutation in a specific gene known as hemoglobin brings about alterations in the composition of RBCs.

Consequently, this mutation causes the RBCs to adopt a rigid and sickle-shaped configuration, resembling the agricultural instrument known as a sickle. These sickle-shaped cells possess various drawbacks:

1. **Reduced Flexibility**: Their rigidity makes it difficult for them to navigate narrow blood vessels. This can lead to blockages that restrict blood flow and oxygen delivery to tissues.

2. **Shorter Lifespan**: Sickle-shaped cells are more fragile and break down prematurely. This decrease in healthy RBCs can lead to anemia, a condition characterized by fatigue, weakness, and shortness of breath.

3. **Painful Episodes**: When blockages occur, they can cause excruciating pain, a main symptom of SCD. These episodes can last for hours or even days and often require medical intervention.

4. **The Root Cause**: Genetics and Inheritance

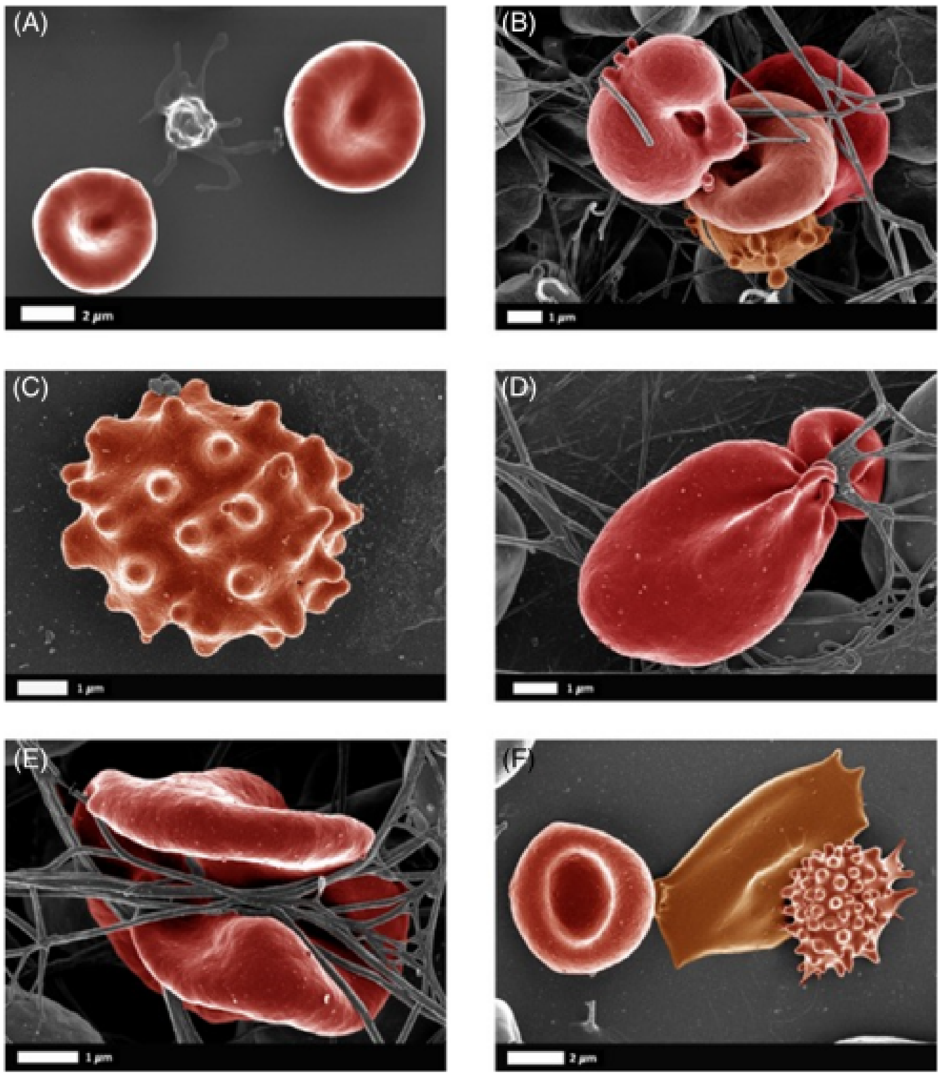


Figure 2.1: Examples of eryptotic red blood cells (RBCs) in inflammation. (A) Healthy RBCs with a platelet; (B) Type 2 diabetes; (C, D) Parkinson's disease; (E) Rheumatoid arthritis; (F) healthy whole blood exposed to interleukin-8 [60]

SCD is an inherited condition. This means that a child inherits two abnormal copies of the hemoglobin gene, one from each parent. If a child inherits only one abnormal copy (from one parent) and one normal copy (from the other parent), they carry the sickle cell trait but typically don't experience symptoms. The trait carriers can still pass the abnormal gene on to their children.

2.5.1 Variants of Sickle Cell

There are various types of SCD, each exhibiting different levels of severity depending on the specific mutations in the hemoglobin gene. Certain types of SCD present with milder symptoms, while others are more severe. Hemoglobin, the protein responsible for carrying oxygen in red blood cells, can be affected by different mutations that alter its structure and function. Consequently, distinct types of SCD arise, each with its own unique characteristics and clinical manifestations.

One of the most prevalent forms of SCD is sickle cell anemia, which occurs when an individual inherits two copies of the hemoglobin S gene, one from each parent. This particular form typically leads to the most severe symptoms, including frequent pain crises, anemia, organ damage, and increased vulnerability to infections. Individuals with sickle cell anemia often require regular medical care and interventions to manage their symptoms and prevent complications.

Other types of SCD include hemoglobin SC disease and sickle beta thalassemia, which result from inheriting one copy of the hemoglobin S gene along with another abnormal hemoglobin gene variant. These forms of SCD may exhibit a less severe clinical course compared to sickle cell anemia, but individuals with these conditions can still face significant health challenges, such as pain episodes, anemia, and organ damage.

In addition to these well-known forms of SCD, there are also rare variants and combinations of hemoglobin mutations that can lead to different clinical presentations. Some individuals may experience fewer symptoms and complications, while others may have more severe disease manifestations. Factors such as the presence of additional genetic modifiers, environmental influences, and access to healthcare can also impact the severity and progression of SCD.

2.5.2 Future Directions

While there's currently no cure for SCD, researchers are actively exploring potential gene therapies that could permanently alter the abnormal gene and allow for the production of healthy RBCs. Additionally, continuous research aims to improve existing treatment options and management strategies to help people with SCD live longer and healthier lives.

SCD is the main motivator for the research carried out in this thesis.

Chapter 3

State-of-the-Art Review

3.1 Utilizing Neural Networks for Blood Diagnostics

The integration of NNs in medical diagnostics has started a transformative era in healthcare and their benefits and advantages are studied in [13]. Leveraging the power of deep learning, NNs analyze complex medical data, ranging from diagnostic images to patient records, with a remarkable capacity to identify compolex patterns and anomalies. In the medical diagnostics, NNs play a pivotal role in tasks such as image classification, disease detection, and risk prediction. Furthermore, NNs can be trained on diverse datasets to recognize subtle variations of various medical conditions, offering healthcare professionals valuable insights for early detection and personalized treatment strategies.

In recent years, the integration of artificial intelligence and machine learning, particularly NNs, has opened new doors for automating RBC classification [30]. The application of NNs in the classification of RBCs using numerical data obtained from flow cytometry and information extracted from video sequences is actual hot topic to study.

Numerical Data from Flow Cytometry

Flow cytometry [107] is a widely used technique for characterizing RBCs based on their size, shape, and optical properties. Numerical data, including size distribution, hemoglobin content, and fluorescence intensity, are collected from these analyses. NN, such as CNN, has been employed to process this data, making it possible to classify RBCs into various subtypes. By training on a large dataset of annotated RBC samples, CNNs can learn to recognize distinctive patterns in the numerical data, enabling high-accuracy classification and providing valuable insights into the RBC population.

Visual Sequencies for Dynamic Analysis

In addition to numerical data, the analysis of RBCs can be enhanced by considering their dynamic behavior captured in video sequences. Microscopic video recordings enable the tracking of individual RBCs' motion, deformation, and interaction with other blood components, used in the paper [47]. Recurrent NNs (RNNs) and long short-term memory networks (LSTMs) have been employed to process video data. These networks can learn temporal dependencies in RBC behavior and classify them based on parameters like velocity, deformability, or agglomeration, which can provide insights into conditions like sickle cell disease or malaria.

Combining Numerical and Video Data

An integrative approach harnessing both numerical data and video sequences offers a comprehensive perspective on RBC classification. By combining CNNs for numerical data and RNNs for video data, hybrid networks can identify RBC subtypes with higher precision. This synergy allows for the assessment of RBCs based on both their static and dynamic characteristics, providing a more complete diagnostic framework and facilitating the detection of abnormalities that might be missed by using either data type alone.

Challenges

The application of NNs for RBC classification has shown significant promise, butit is not without challenges. Data preprocessing, the acquisition of high-quality video sequences, and the creation of diverse and comprehensive datasets for training and validation remain important issues to address. Future research might also focus on improving the interpretability of NNs for better clinical adoption. Additionally, there is potential for real-time RBC analysis, allowing for immediate feedback in clinical settings, which could revolutionize the field of hematology.

The utilization of NNs for RBC classification from numerical data obtained through flow cytometry and video sequences offers a powerful tool for medical diagnostics. These techniques can enhance the precision of RBC analysis and improve our understanding of various hematological conditions. As technology and machine learning algorithms continue to advance, the potential for more accurate RBC classification becomes increasingly promising, ultimately benefiting patient care and medical research.

3.2 Learning with Physical Information

Despite significant progress in simulating physical problems using numerical discretization of partial differential equations (PDEs), incorporating noisy data into existing machine learning algorithms remains a challenge which is detailly described in [27]. Moreover, solving complex physical problems is often computationally expensive and requires various formulations and sophisticated computer programs due to missing or noisy physical conditions.

Machine learning has emerged as a promising alternative because it can identify multidimensional correlations and handle ill-defined problems. In [80] authors show that they are superior for inverse problems that cannot even be solved with standard techniques. Deep learning approaches naturally provide tools for automatically extracting features from vast amounts of observational data. They can also help connect these features with existing approximate models and leverage them in creating new prediction tools.

However, training deep NNs requires a large amount of data. Despite the volume, speed, and diversity of available (collected or generated) data streams, it is still challenging in many real cases to integrate data into existing physical models. The ability to collect and create observational data far exceeds the ability to sensibly collect them, and current machine learning approaches cannot extract interpretable information and insights from this flood of data. There are two possible solutions to this problem.

First one is the integration of fundamental physical laws and domain knowledge into machine learning models, which can provide added information in the form of strong theoretical constraints. For this purpose, physics-informed learning of the specific physical process is necessary. Previous knowledge derived from observational, empirical, physical, or mathematical understanding of the world can be used to improve the performance of the learning algorithm. The main motivation for developing these algorithms is that such added information or constraints can yield more interpretable machine learning methods that remain robust even when using imperfect data (such as missing or noisy values, outliers, etc.) and can provide accurate and physically consistent predictions.

In the second approach, a sufficient amount of data for training a machine learning model is obtained through physically accurate simulations. If these simulations are physically accurate, the information about physics is implicitly embedded within the data. [1]

Figure 3.1 schematically depicts three possible categories of physical problems and the associated scope of available data. In the case of a small amount of data, it is

[1]But, data-only models may have high accuracy for observations, but their predictions can be physically inconsistent, leading to poor generalization.

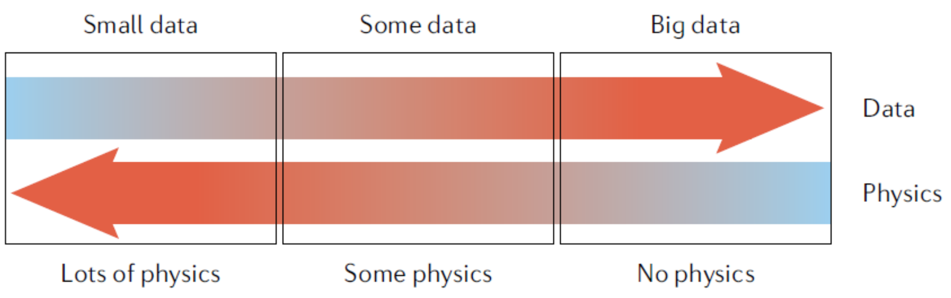


Figure 3.1: Various situations that occur during the training of a physics-informed models. (Figure adapted from [58])

assumed that we know the entire physics, and data is provided for initial and boundary conditions, as well as coefficients of the partial differential equation. The middle option represents the most common case in applications where we know some data and some physics, or we are missing some parameter values or even an entire term in the partial differential equation. Finally, there is the case with a large amount of data, where we may not know any physics, and a data-driven approach may be the most effective. Physics-informed machine learning can integrate data and governing physical laws, including models with partially missing physics, in a unified manner.

3.2.1 Principles of Physics-Informed Learning

Creating a physics-based learning algorithm involves introducing suitable observational, inductive, or instructional influences that can guide the learning process towards identifying physically consistent solutions.

• Observational Influence is incorporated through data representing the fundamental physics. Training a machine learning system on such data allows it to learn functions, vector fields, and operators that reflect the physical structure of the data. A large amount of data is required, and the cost of obtaining a sufficient amount of data must be considered.

• Inductive Influence comprises assumptions that can be incorporated into the model architecture so that the desired predictions implicitly satisfy a set of physical laws, usually expressed as certain mathematical conditions.

• Learning Influence is manifested through the choice of an appropriate loss function, conditions, and inference algorithms that influence the training of the model to explicitly encourage convergence towards solutions that adhere to fundamental physics. By applying such mild constraints, basic physical laws can be approximately satisfied, allowing us to introduce a wide range of physical influences. In practice, this involves adding penalties based on physical conditions to the loss function.

The ways to influence the learning algorithm towards physically consistent solutions are not mutually exclusive and can be combined. Various hybrid learning approaches have been proposed, as seen in [45]. The authors combine the influence of observations and learning using a large number of simulated data and a NN with a conditional training method to create a Navier-Stokes model for turbulent fluid flow.

Physics-informed learning can easily combine physical information and noisy data, even when both are imperfect, and still find meaningful solutions, as demonstrated in [100].

3.3 Previous Research of RBC Using Simulations and Neural Networks

Advancements in the scientific understanding of RBC classification have been marked by notable progress in computational modeling and data analysis. Researchers have turned to computer simulations to simulate and study the behavior and elasticity of RBCs in blood flow. The integration of these simulations with advanced data analysis techniques has led to more accurate classification of RBC properties.

The classification of healthy and diseased blood cells in blood flow is discussed in [31], emphasizing the loss of elasticity in diseased cells, leading to displacement by healthy cells towards the canal's edge. This behavior is observed in microfluidic devices as well. Authors present tests of various NN models for classification accuracy. The models learn based on the temporal sequence of cell positions to identify the type of blood cell. Models incorporating information about extreme cell positions obtained from simulations. The most powerful NN model in terms of accuracy and learning speed is identified, and evaluated for its generalization ability to various input data types.

Machine learning analysis of simulation outputs exposes certain disadvanteges in the simulation model, such as periodic channel issues, providing insights for potential improvements. A significant difference in the classification accuracy of healthy and diseased cells (approximately 10 %) highlights the importance of considering extreme cell positions.

3.4 Segment Anything Model

The field of image segmentation has traditionally relied on complex algorithms and significant manual effort for accurate object identification and isolation. Meta's Segment Anything Model (SAM) [64] presents a novel approach, offering a user-friendly and powerful tool for this task.

SAM leverages deep learning, a subfield of artificial intelligence inspired by the structure and function of the human brain. This approach enables the model to learn from vast quantities of labeled image data, establishing relationships between specific objects and corresponding pixel patterns. Consequently, SAM can accurately segment intricate scenes based on user instructions provided through points, bounding boxes, or even user gaze within an augmented reality environment.

The key to SAM's efficiency lies in its ability to analyze extensive datasets of labeled images. Through this process, the model learns to differentiate between objects and their backgrounds, allowing for precise segmentation. This functionality translates into valuable applications across various fields.

Within the realm of biological research, SAM offers significant potential for analyzing microscopic images of cells and tissues. For RBC, SAM can quickly segment individual RBCs from a complex image, enabling researchers to efficiently analyze their size, shape, and other features. This capability holds particular significance in studying diseases that affect RBC morphology, such as sickle cell disease.

Furthermore, SAM's capabilities extend beyond static images, encompassing video recordings as well. Consider a video depicting traffic on a busy street. SAM can segment individual vehicles within the video, aiding researchers in analyzing traffic flow patterns or assisting autonomous vehicle developers in identifying obstacles. As research progresses, we can anticipate even more innovative applications for SAM, further expanding its impact on various disciplines.

Chapter 4

Simuations Using Computational Fluid Dynamics

This research utilizes a simulation tool to generate data for training machine learning models. The tool allows us to simulate complex phenomena, resulting in a large amount of data for training neural networks. By creating and studying various scenarios, we gain insights into how systems behave under different conditions. The details of the simulation tool are described in this chapter.

The simulated data serves as a valuable resource for training NNs. By exposing them to this diverse range of scenarios, we enable NNs to learn and make predictions more accurately. This approach has two benefits: it improves our understanding of complex systems, and it allows NNs to solve real-world problems by leveraging the knowledge gained from the simulations.

The simulations of bloodflow are enabling us to research hemodynamics, the study of blood flow within the vascular system. This dynamic process ensures continuous circulation of oxygen, nutrients, and waste products throughout the body. We can visualize the vascular system as a complex transportation network, where RBCs)act as carriers for this vital fluid. The heart, functioning as the central pump, propels blood through the network, creating a never-ending cycle of delivery and return.

Understanding the intricate details of blood flow within the network is crucial for diagnosing and treating cardiovascular diseases. Computational Fluid Dynamics (CFD) [32] offers a sophisticated approach to studying hemodynamics. CFD uses numerical methods and algorithms to create digital simulations of fluid flow. This functions as a digital laboratory, allowing researchers to explore the complexities of blood circulation without the need for complex mathematical models. By simulating various scenarios, CFD provides valuable insights into the behavior of blood flow within our vascular system.

4.1 Fundamentals of Computational Fluid Dynamics

The core of CFD lies in mathematical models capturing fluid dynamics principles, with the Navier-Stokes equations forming the theoretical foundation [136]. These equations describe the conservation of mass and momentum in fluids, providing the mathematical basis for CFD simulations. In simpler terms, they explain how fluids behave and interact.

Geometric modeling translates the complex world of RBCs into a digital one. Medical imaging data (MRIs, CT scans) are used to represent RBC shapes and branching patterns. Imagine creating a digital landscape mirroring the intricate details of the human vascular system. This geometric modeling provides the virtual canvas for CFD simulations.

Fluid dynamics, governed by continuous mathematical equations, undergoes a transformation in CFD through discretization. This involves breaking down smooth curves into manageable segments. Meshing, or creating a grid over the geometric model, facilitates this process. Discretization translates the continuous language of fluid dynamics into a series of calculable steps, allowing computers to perform mathematical computations.

CFD relies on numerical methods like finite difference, finite volume, and finite element methods to convert mathematical abstractions into tangible simulations. Each method offers a unique approach to approximating solutions to the discretized equations. Essentially, these methods enable computers to understand and simulate complex fluid phenomena.

Simulating blood flow in the digital domain requires setting the stage with boundary conditions. These define the rules for how simulated blood flow interacts with virtual RBC walls. Parameters like blood velocity, pressure distributions, and RBC wall properties act as guiding principles. This section explores how these boundary conditions are established, influencing the behavior of the simulated fluid and reflecting real-world interactions within RBCs.

4.2 Components of Computational Fluid Dynamics Simulations

Well-defined boundary conditions are essential for accurate CFD simulations [44]. These conditions specify the interaction between simulated blood flow and the virtual walls representing red blood cells (RBCs). Parameters like blood velocity, pressure distributions, and RBC wall properties guide the behavior of the simulated fluid, reflecting real-world interactions within the vascular system.

CFD simulations generate a wealth of hemodynamic parameters that provide insights into blood flow behavior. These parameters include:

- Blood velocity: quantifies the speed of blood flow.
- Pressure distributions: visualize pressure variations within the simulated blood flow.
- Shear stress: represents the force experienced by RBCs due to blood flow.

By analyzing these parameters, researchers gain a comprehensive understanding of how blood interacts with RBC walls and the forces exerted on them. This section explores the significance of these hemodynamic parameters in unraveling the complexities of blood circulation.

CFD simulations extend beyond simulating healthy blood flow. They allow researchers to create digital models of cardiovascular diseases like atherosclerosis, stenosis, and aneurysms [113]. These can be incorporated into the virtual laboratory, transforming it into a diagnostic tool. By introducing pathologies into a digital representation, researchers can analyze their impact on blood flow dynamics. This section explores how CFD simulations act as a virtual microscope, offering insights into deviations from normal blood flow patterns and providing a diagnostic lens into cardiovascular diseases.

Advancements in medical imaging and computational techniques enable patient-specific CFD simulations. These simulations are no longer limited to generic models; they can now incorporate individual anatomical and physiological variations. This allows for simulations tailored to the unique characteristics of a specific patient. This section delves into the evolution towards personalized medicine, exploring how patient-specific simulations enhance our understanding of individual variations in blood flow dynamics and pave the way for tailored medical interventions.

CFD simulations can act as a virtual treatment planner, offering insights into potential outcomes of medical interventions like stent placements and bypass surgeries [108]. This section navigates the role of CFD simulations in treatment planning, elucidating how they contribute to decision-making processes in clinical settings. By predicting the hemodynamic consequences of interventions, CFD becomes an invaluable tool for clinicians, aiding in devising personalized and effective treatment strategies.

Beyond research, CFD simulations are finding their way into practical clinical applications. Doctors can potentially access virtual dashboards that provide insights into a patient's unique blood flow dynamics. CFD is transcending its research origins and offering tangible benefits in diagnosis support and disease management. By bridging the gap between theoretical knowledge and real-world applications, CFD emerges as a transformative force in the clinical landscape, enhancing our ability to understand and address cardiovascular challenges.

4.3 Applications of Computational Fluid Dynamics in Blood Flow Simulations

CFD simulations in blood flow analysis offer valuable insights for personalized medicine. These simulations move beyond generic models by incorporating a patient's unique anatomy using medical imaging data like MRI or CT scans [29, 35]. By creating a tailored digital representation of the patient's vasculature, CFD provides clinicians with a nuanced understanding of blood flow patterns, facilitating more precise treatment planning.

In treatment planning for cardiovascular diseases, CFD simulations act as a powerful tool for predicting outcomes of interventions like stent placements and bypass surgeries. Clinicians can virtually rehearse these procedures, gaining insights into potential hemodynamic consequences. This allows for tailored interventions based on a patient's specific hemodynamic profile.

The integration of CFD from research to clinical applications is transformative. It offers real-time insights into a patient's blood flow dynamics, supporting diagnosis and disease management. As CFD becomes an useful tool in clinical workflows, it enhances diagnostic accuracy and facilitates the development of personalized treatment strategies for cardiovascular challenges [103].

The applications of CFD in blood flow simulations open exciting possibilities for future research. The evolution of personalized medicine, driven by patient-specific simulations, fosters further exploration. Researchers are investigating the integration of CFD with emerging technologies like artificial intelligence and machine learning to enhance predictive capabilities. However, challenges like model validation and computational resource demands remain at the forefront of research. As CFD continues to evolve, its role in shaping the future of cardiovascular research and clinical practice becomes increasingly vital, offering a dynamic landscape for exploration and innovation.

Model validation is crucial for ensuring the reliability of CFD simulations [134]. Experimental data, obtained through techniques like Doppler ultrasound or MRI, serves as the benchmark for comparing simulation results. This iterative process of validation ensures that virtual representations align with physical reality, instilling confidence in the accuracy and predictive capability of CFD simulations.

The pursuit of high-fidelity simulations presents a challenge in terms of computational resources. Increased simulation accuracy demands greater computational power. Researchers are continuously seeking to optimize algorithmic efficiency and parallel processing techniques to navigate the delicate balance between simulation accuracy and the feasibility of computational resources.

4.3.1 Modeling Red Blood Cells Properties

Blood flow interacts with the mechanical properties of red blood cell (RBC) walls, introducing complexity known as fluid-structure interaction (FSI) [51]. Modeling this interaction accurately presents a significant challenge. RBCs are not rigid; they exhibit elasticity and deform in response to fluid forces. Capturing this two-way influence between blood flow and RBC walls necessitates sophisticated modeling techniques [42]. FSI simulations aim to harmonize the fluid dynamics of blood with the structural mechanics of RBCs, contributing to a comprehensive understanding of cardiovascular dynamics.

A defining characteristic of RBCs is their elasticity, which influences their response to varying pressures and blood flows. Incorporating this elasticity into CFD simulations increases complexity. Accurate representation of RBC properties requires understanding how they deform under the pulsatile nature of blood flow. This challenge involves not only modeling elasticity but also integrating it seamlessly into simulations. Researchers strive for a balance between computational efficiency and capturing realistic RBC elasticity, a crucial consideration for precise hemodynamic simulations.

Modeling RBC elasticity presents challenges. Balancing computational efficiency with capturing realistic behavior is a key consideration. Simulations need to accurately represent how RBCs deform under pulsatile blood flow while remaining computationally feasible.

4.4 ESPResSo Simulation Tool

The ESPResSo Simulation Tool [46] is a software designed for simulating blood flow dynamics within the human cardiovascular system. Developed as a computational tool, it employs advanced numerical methods to simulate and analyze the behavior of blood within RBCs. This tool plays a crucial role in enhancing our understanding of hemodynamics, providing insights into the complexities of blood circulation.

The primary purpose of the ESPResSo Simulation Tool is to contribute to the field of CFD with a specific focus on blood flow. It is significant for its ability to virtually model and simulate the movement of blood, allowing researchers and medical professionals to gain valuable insights without resorting to traditional, more invasive methods.

This simulation tool is equipped with features that enable accurate representation of RBCs and their dynamics. It can consider factors like fluid-structure interaction, capturing how RBCs deform under the influence of blood flow. The tool also accounts for the elasticity of RBCs, an essential characteristic in understanding their response to varying pressures and flows.

4.4.1 Applications in Patient-Specific Simulations

A notable application of the ESPResSo Simulation Tool is its capacity for patient-specific simulations [93]. By incorporating individualized anatomical and physiological data from medical imaging, the tool can create simulations tailored to a specific patient. This personalized approach is valuable for clinicians in devising treatment strategies that consider the unique characteristics of each individual.

In treatment planning, the ESPResSo Simulation Tool becomes a virtual laboratory, allowing practitioners to predict the outcomes of medical interventions. Whether it's stent placements or bypass surgeries, the tool facilitates a digital rehearsal, offering insights into how these procedures might impact blood flow. This predictive capability aids clinicians in making informed decisions about the most suitable interventions for their patients.

Despite its advantages, the ESPResSo Simulation Tool faces challenges. Model validation is crucial, emphasizing the need for comparison with experimental data to ensure the accuracy of simulations. Additionally, the demand for computational resources, especially in high-fidelity simulations, and the consideration of fluid-structure interaction and vessel elasticity present ongoing challenges that researchers aim to address for further improvements.

The ESPResSo Simulation Tool represents an important step in the evolution of blood flow simulations. Future directions may involve the integration of artificial intelligence and machine learning to enhance predictive capabilities. Researchers continue to explore ways how to validate and optimize the model, ensuring that the tool evolves to meet the increasing demands of precision medicine and contributes to advancements in cardiovascular research and clinical practice.

4.5 General Description of the Current Model

In fluid dynamics simulations, ESPResSo employs a Lattice-Boltzmann method [9] to model the behavior of fluids. In this method, the fluid is represented using a fixed Eulerian grid, where the movement of the fluid is computed as a flow at each point within this Eulerian grid. Objects immersed within the fluid are represented using a Lagrangian grid, which moves in tandem with the simulated fluid. The fluid exerts forces on the immersed objects, represented by the Lagrangian grid points, originating from the Eulerian grid, and these forces are calculated as averages of the Eulerian grid points. The interaction between immersed objects and the fluid is bidirectional, meaning that the forces acting on the immersed objects by the fluid are evaluated in a similar manner to the forces exerted by the immersed objects on the fluid [34].

A critical component of the ESPResSo model involves the modeling of red blood cells (RBCs). RBCs constitute approximately 45% of the volume fraction, making their accurate representation crucial for the model's proper functioning.

Within the ESPResSo framework, RBCs are modeled as a membrane filled with fluid. The membrane itself is represented as a triangulation of points interconnected by elastic bonds. The elasticity of these bonds is defined by experimentally obtained elastic coefficients.

The surface and volume properties of RBCs incorporated into the model are complemented by parameters that define the interaction between macroscopic objects immersed in the fluid. There are two primary types of interactions at play. First, there is a repulsive potential between two bodies, which prevents overlap of the grids of two immersed bodies during collisions and compels them to bounce off each other. The second type is adhesion, representing the ability to form bonds between two objects in the bloodstream - either between two RBCs or between a RBC and another immersed object.

In the context of this research, ESPResSo, an open-source software for molecular dynamics simulations primarily used in the fields of chemistry, physics, and molecular biology, serves as the numerical model. The research group CIF has extended ESPResSo with a module for modeling elastic objects. The core computations are carried out in C++, while the user interface for simulation execution is programmed in Python.

Blood is represented as a suspension of fluid with embedded solid particles. The flow of blood in microfluidic devices is simulated using the Lattice-Boltzmann method [114], with the continuous fluid discretized into discrete points located on a fine Eulerian grid. This approach offers the advantage of relatively low computational demand. Embedded objects are depicted using a Lagrangian grid that moves along with the modeled object, and interactions between the fluid and embedded objects are taken into account. This means that the forces exerted by an embedded object on the fluid are evaluated in a similar manner to the forces exerted by the fluid on the embedded object.

Research in the domain of modeling elastic objects in blood has primarily focused on RBCs, which constitute the largest solid component of blood, approximately 45% [67]. The RBC model is discrete, involving the triangulation of the object's surface (RBCs). The elastic properties of RBC are determined by five elastic moduli: the elastic modulus, bending modulus, local area preservation modulus, global area preservation modulus, and volume preservation modulus, each with its specific stiffness values, calibrated through a stretching experiment.

The numerical model also encompasses settings for various interactions, including the interaction between cells and the channel surface, self-interaction of embedded objects, and interactions between embedded objects. The details of the interaction of solid objects are elaborated in the relevant research work.

Additionally, a model of a cell with a nucleus, containing cancer cells, is currently in the development and validation process.

In the simulation experiments utilized, the microfluidic device is represented as a rectangular channel, with blood flowing from left to right along the horizontal x-axis. In this direction, the channels are periodic, meaning that once a cell exits the simulation channel, it re-enters from the other side.

Various experiments require different configurations of the numerical model depending on the studied properties and phenomena. The basic parameter settings for RBC and the fluid in each NN model are provided in the section describing the dataset.

All simulation experiments are executed through the open-source software ESPREesSo [14], employing its Lattice-Boltzmann and Object-in-fluid [CIF] modules. Blood flow simulations typically encompass two core components: the fluid and the cell membrane. These elements are interrelated and interact via forces.

The Lattice-Boltzmann method governs the fluid component, while the cell membrane is modeled using a spring network, with their connection maintained through a dissipative version of the Immersed Boundary Method (IBM). A schematic representation of the RBC model and the individual elastic forces are depicted in Figure 4.1.

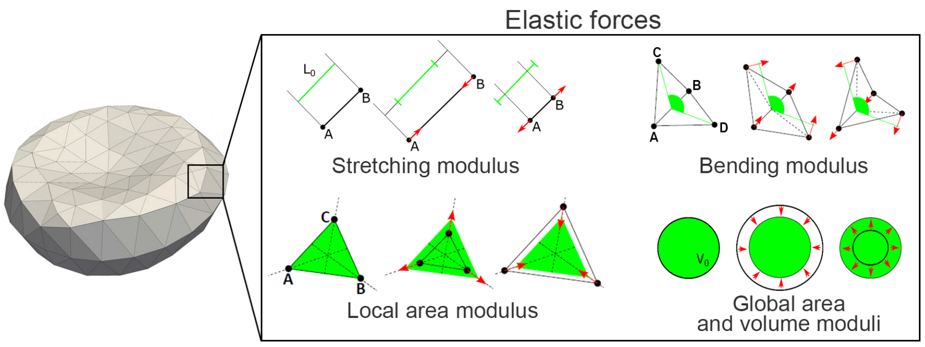


Figure 4.1: Schematic representation of the cell membrane. Each individual cell is modeled by a spring network of boundary points bound by elastic interactions.

The red blood cells were represented as a surface mesh with 374 nodes. In their relaxed state, they assumed the typical biconcave shape with dimensions of $7.82 \times 7.82 \times 2.58\,\mu m$ and a volume of $90.75\,\mu m^3$. These cells were filled with the same fluid as their surroundings. To model the elastic properties of the cell membrane, five distinct types of elastic forces were employed, each corresponding to one modulus of elasticity and its corresponding parameter: modulus of elasticity (preservation of edge length), modulus of bending (preservation of angles between adjacent triangles), local modulus of area preservation, global modulus of area preservation, and volume preservation modulus. A schematic representation of the RBC model and the individual elastic forces is illustrated in Figure 4.1. The calibration of elastic coefficients for healthy and well-deformable RBCs and their subsequent validation are detailed in [56].

The simulation model utilized the lattice-Boltzmann method to represent the fluid, a spring network model to simulate the cell membrane, and a dissipative version of the Immersed Boundary Method (IBM) to connect them. In addition to the fluid force, elastic forces were exerted on the cell mesh points, evaluated from the deformation of the cell. The resulting force $\vec{F}_{tot}$ acted as the driving force, governed by Newton's equation:

$$m\frac{\partial^2 \vec{x}(t)}{\partial t^2} = \vec{F}_{tot},$$

where m represents the mass of the mesh points. The sources of these forces included the elasto-mechanical properties of the cell membrane, fluid-cell interaction, and possibly other external stimuli.

4.6 Using Simulation Model in Thesis

Unfortunately, there is a shortage of available blood flow recordings, and the limited video quantity is insufficient for effective NN training, even with data augmentation methods. An alternative approach is to employ computational simulations based on a numerical model tailored to the experimental requisites, including desired blood flow characteristics and structure - ESPREsSo. Computational simulations provide a more comprehensive understanding of the behavior of the examined blood flow compared to restricted video recordings. Consequently, the incorporation of computational simulations into the study of hemodynamics and the advancement of novel diagnostic techniques holds substantial significance. The simulation model employed in this research extends ESPResSo, representing blood cells as deformable entities within a fluid flow. The model's validity is consistently assessed by comparing simulation results with laboratory experiments [70]. Utilizing simulations enables the generation of a substantial dataset for NN training, particularly from video analyses of such experiments which we laverage in Chapters 6 and 7.

Despite being a simulation, this model allows us to mimic real experiments and even adjust computational outputs to match simple image recordings from in vitro experiments.

Chapter 5

Neural networks for sequential data

5.1 Neural Networks

Neural Networks (NNs) are a foundational paradigm within machine learning and artificial intelligence [112]. They are inspired from the complicated architecture and functioning of the human brain, making them versatile tools capable of addressing a wide array of complex tasks. At their core, Ns consist of interconnected artificial neurons, each serving as an information processor and transformer. The versatility and applicability of NNs extend across diverse domains, including image and speech recognition, natural language processing, and reinforcement learning, thereby establishing their central role in modern technological landscapes.

In our investigation, by employing NNs, we aim to predict and class. Our research focused on comparing the capabilities of models with 3D information and its 2D projections based on video recordings view.

To leverage the power of NNs, in our research we use Tensorflow library [3] in Python as a main tool to implement our NN models.

5.1.1 Neural Network Components and Architecture

The fundamental unit of a NN is the artificial neuron, designed to emulate the basic functionality of biological neurons. These neurons receive input, perform mathematical operations, and generate outputs. A NN is structured hierarchically into layers, each with its distinct function:

Serving as the network's initial point of interaction, the input layer is responsible for receiving and processing raw input data. Each neuron in this layer represents a specific feature or attribute of the input data, acting as the sensory gateway to the network's processing.

NNs typically incorporate one or more hidden layers, positioned between the input and output layers. These hidden layers play a critical role in enabling the network to learn and model complex relationships within the data. The term "hidden" signifies that the outputs of these layers remain hidden and serve as intermediary constructs in the network's information processing cascade.

The ultimate layer in the hierarchy is the output layer, where the NN produces its final predictions or classifications. The number of neurons within the output layer adapts to the specific requirements of the task at hand.

5.1.2 Learning and Training Process

The NN functionality revolves around its capacity to adapt and enhance its performance through the process of learning [127]. During the training phase, NNs undergo a critical process known as backpropagation. The training process unfolds through several steps.

The training process begins with the initialization of the parameters of each neuron, including weights and biases. These parameters are set to random values.

With the parameters initialized, data traverses the network in the forward direction. Neurons within each layer execute a weighted summation of their inputs, followed by the application of an activation function. The output from each neuron proceeds to the subsequent layer, facilitating the cascade of information processing.

The network's final output is compared to the actual target values, enabling the computation of an error or loss. This error serves as a quantifiable measure of the disparity between the network's predictions and the ground truth.

Subsequently, the error is propagated backward through the network, leading to the adjustment of the weights and biases of each neuron. This iterative process is designed to minimize the error and improve the network's predictive accuracy. Optimization algorithms, such as gradient descent, are commonly employed to drive this weight adjustment process.

The training continues until the network's performance, as indicated by the error, reaches an acceptable level or converges to a minimum. This convergence signifies that the network has effectively learned the underlying patterns in the data.

5.1.3 Depth and Complexity

NNs exhibit varying levels of depth, with deep networks being commonly referred to as Deep Neural Networks (DNNs). The depth of a network is determined by the number of hidden layers it incorporates. Deep networks are capable of modeling complicated and nonlinear relationships within data, making them particularly suitable for tasks requiring complex feature extraction and abstraction.

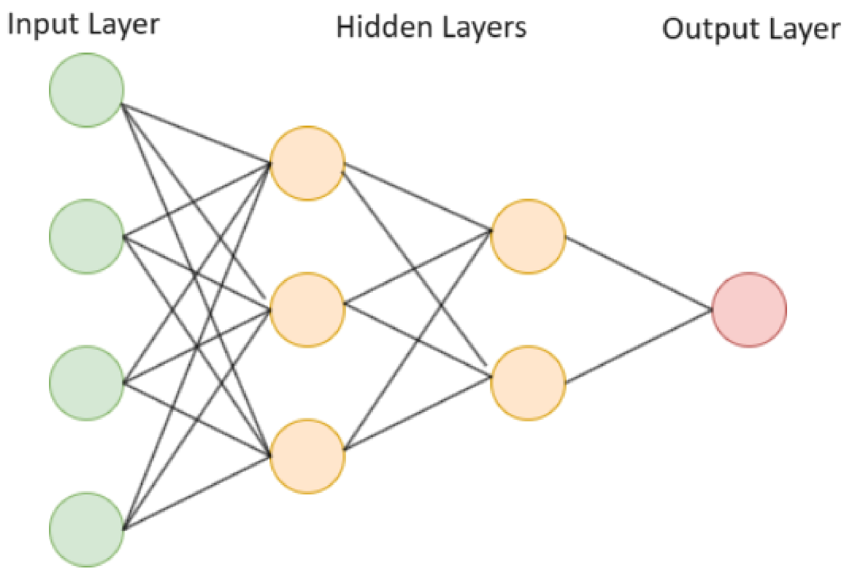


Figure 5.1: Simple NN schema.

5.1.4 Applications of Neural Networks

The versatility of NNs is vividly manifested in their widespread applications [4, 12]:

NNs, particularly deep learning models, exhibit remarkable proficiency in discovering complex patterns and features embedded within images. This capability roots from the natural hierarchical structure of NNs, where layers of interconnected neurons progressively extract and abstract information. In the context of image recognition and classification tasks, NNs excel at hard complex details, shapes, and contextual relationships within visual data. The convolutional layers of CNN, for instance, are designed to automatically learn hierarchical representations of visual features, starting from simple edges and textures and progressing to more complex object parts and configurations. This hierarchical feature extraction allows NNs to not only identify individual objects in images but also to grasp the broader context and relationships among different elements. This adaptability makes NNs a powerful tool in various domains, from computer vision applications like facial recognition and autonomous vehicles to medical imaging for diagnosing diseases based on visual data. The capacity of NNs to excel in image analysis underscores their versatility and effitiency in addressing diverse challenges related to object identification and pattern recognition within the visual domain.

In natural language processing (NLP), NNs are instrumental in a multitude of tasks, including language modeling, text generation, sentiment analysis, and machine translation. They exhibit an unparalleled ability to capture the contextual nuances of words and sentences, rendering them invaluable in the understanding and generation of human language.

NNs play a pivotal role in converting spoken language into text, enabling voice assistants, transcription services, and numerous other applications. Their adeptness lies in their capacity to navigate the intricacies of sequential audio data. This area is called speech recognition.

In the domain of reinforcement learning, NNs form the foundation for training intelligent agents to make sequential decisions. This underpins significant advancements in robotics, autonomous systems, and game playing.

NNs represent the cornerstone of current machine learning and artificial intelligence, mirroring the neural processes of the human brain. Their hierarchical architecture, the intricacies of the training process, and their ability to handle depth render them useful tools in the modern technological world.

5.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [135] form a specialized and dynamic category purpose-built for the complex processing of sequential data. In contrast to the traditional feedforward NNs, RNNs introduce a transformative element through the incorporation of recurrent connections. These looped connections anaible RNNs the unique capability to maintain a form of memory or state, making them performing very well in an array of tasks, particularly the analysis of sequential data. This extends to domains as diverse as natural language processing, time series prediction, and speech recognition.

5.2.1 Fundamental Characteristics of RNNs

RNNs demonstrate an proficiency in handling sequential data, wherein the order of data points is of primary significance. Such sequences encompass a wide spectrum, spanning sentences in natural language, time series data, and audio signals. RNNs stand as versatile instruments for analyses the temporal evolution of information natural to these data forms.

The characterisric feature of RNNs is their recurrent connections. These interconnections provide a channel through which information can flow from one time step in the sequence to the next. This cyclical data flow mechanism enriches RNNs with the invaluable ability to maintain a state or memory of prior inputs, allowing them to establish temporal contextual quality necessary in the analysis of sequential data.

RNNs are designed to reveal the complex dependencies embedded within data. They are focused on recognizing patterns and relationships that evolve and develop over time – a fundamental characteristic critical to solving a variety of real-world problems, from sentiment analysis to stock market forecasting.

5.2.2 Challenges and Progress in RNNs

Despite their effectiveness, RNNs face challenges, as discussed in [10], with a notable hurdle known as the *vanishing gradient problem.* This challenge becomes evident when training RNNs on lengthy sequences, leading to excessively small gradients during backpropagation for weight updates. The consequence is a hindered learning process, making it tough for the network to grasp long-range dependencies within the data.

To address this issue, advanced RNN architectures have been introduced, including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU).

LSTMs are equipped with specialized mechanisms designed to manage information over extended time periods. They incorporate three essential gates—the input gate, output gate, and forget gate—that work together to control information flow within the network. LSTMs excel in capturing long-term dependencies, gaining recognition in various applications such as machine translation and speech recognition.

GRUs present a more streamlined rendition of the LSTM, engineered to deliver comparable results while conserving network parameters. They incorporate reset and update gates, playing a pivotal role in governing information flow. GRUs are particularly well-suited for applications requiring a delicate balance between memory capacity and computational efficiency.

5.2.3 Long Short-Term Memory

LSTM, which stands for Long Short-Term Memory [115], is a complex NN architecture designed to overcome the limitations of traditional RNNs when dealing with sequential data. Understanding LSTM can be challenging, especially for the concept of gates and cell state. This discussion aims to shed light on LSTM and its essential components.

Various issues that RNNs encounter are solved by LSTM, notably the challenges of long-term dependencies and the vanishing or exploding gradient problems. At the core of LSTM lies its unique cell state, which acts as a form of memory for the network, enabling it to retain important information from the past.

The fundamental structure of LSTM includes three crucial gates:

- **Input Gate**: This gate controls what new information will be stored in the cell state (4.1).
- **Forget Gate**: The forget gate determines what information should be discarded from the cell state (4.2).
- **Output Gate**: The output gate regulates the activation for the final output of the LSTM block at a given timestamp (4.3).

These gates in LSTM employ sigmoid activation functions, which output values between 0 and 1. In most cases, these values are either 0 or 1. The choice of the sigmoid function for gates is strategic because it ensures that the gates yield only positive values, offering a clear decision regarding whether to retain or discard a particular feature. In this context, "0" means the gates block everything, while "1" signifies that the gates allow everything to pass through. The equations and explanation of LSTM is described in 8:

The whole proces can be graphically seen in Figure 5.2

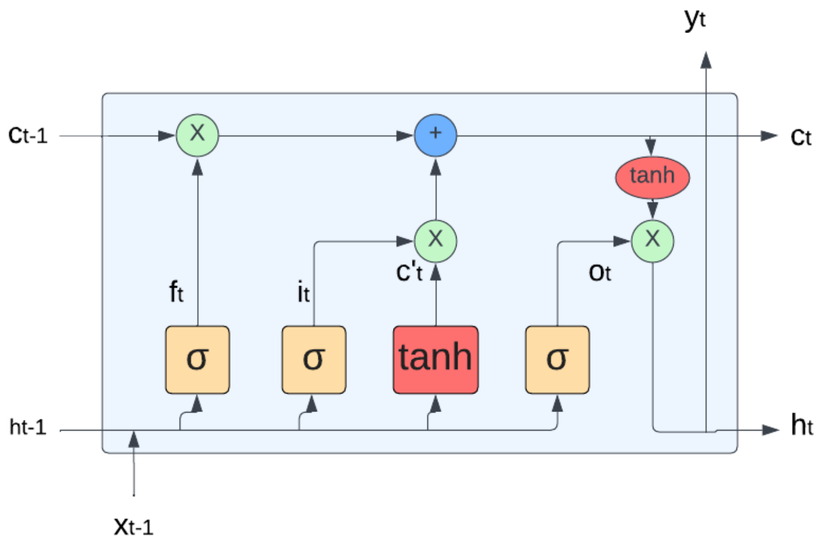


Figure 5.2: LSTM block at timestamp t

In Chapter 6, RNNs are employed to take advantage of their capability to capture long-term dependencies in sequential data. This choice allows us to leverage the strength of RNNs in preserving information over time, making them effective tools for analyzing and predicting patterns in sequences of data.

5.3 Convolutional Neural Networks

In NNs, CNNs [74] are a specialized tool tailored for the processing of grid-like data, with images and videos standing as its most prominent use cases. These networks represent a profound innovation deeply ingrained in their exceptional ability for image recognition, object detection, and a vast spectrum of computer vision tasks.

5.3.1 Key Components and Characteristics of CNNs

The fundamental advantage of CNNs resides in the application of convolutional layers, a transformative mechanism responsible for automatic feature learning from input data.

These layers engage in convolution operations, a process that entails the systematic deployment of filters, also known as kernels 5.3, across the input data. This strategic deployment of kernels is instrumental in revealing complex patterns, structures, and distinctive features embedded within the data. By orchestrating these convolution operations, CNNs progressively assemble a hierarchical representation of the data, progressively transitioning from basic features to increasingly complex and abstract structures.

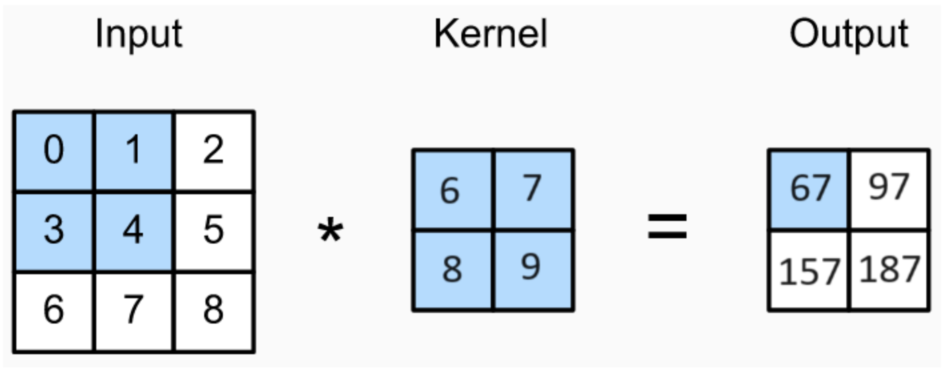


Figure 5.3: Simple CNN kernel.

CNNs have the capability to create a hierarchical representation of input data. The lower layers, positioned at the beginning of the network, specialize in the capture of low-level features such as edges, corners, and textures. These basic components constitute the building blocks of more complex visual constructs. Subsequently, the higher layers synthesize these low-level features, fusing them into more advanced and abstract structures. The hierarchical nature of CNNs mirrors the human visual processing system, where elemental visual components evolve into coherent and complex percepts.

In their architecture, CNNs often incorporate pooling layers, a strategic component engineered to downsample the feature maps produced by the convolutional layers. Pooling operations involve selecting the most characteristic information while systematically discarding redundant or less relevant details. This process holds profound implications for computational efficiency, as it trims the computational load, rendering the network more manageable while still retaining the important information necessary for accurate analysis.

The culmination of the CNN architecture is manifested through the deployment of fully connected layers. These layers receive the extracted feature representations acquired through the network's previous layers. Their role assumes particular significance when it comes to tasks such as classification, regression, or any form of decision-making grounded in the knowledge encapsulated within the learned features. These layers essentially synthesize the acquired knowledge into actionable decisions, be it classifying an image into predefined categories or making predictions based on the collected insights.

5.3.2 Applications of CNNs

The versatility of CNNs finds expression in a wide-range of applications [41], due to their efficiency in processing grid-like data.

At the image recognition, CNNs stand as the foundation upon which systems for identifying and classifying objects, patterns, and structures within images are built. They have emerged as the base of computer vision, revolutionizing image processing.

In image classification tasks, CNNs seamlessly segregate images into predefined categories. Applications ranging from species differentiation in the natural world to product classification in the retail sector benefit significantly from their categorical power.

The capability of CNNs to detect and localize objects within images or video frames is a game-changer. This technology has expanded across domains such as autonomous vehicles, surveillance systems, and robotics, significantly enhancing object tracking and recognition.

CNNs exhibit remarkable utility in the complex domain of image segmentation, a process aimed at partitioning an image into distinct regions or segments. This technology is of primary importance in the fields of medical imaging, scene understanding, and image manipulation, offering unprecedented capabilities for image analysis and annotation.

The dynamic ability of CNNs extends to video analysis, particularly in tracking objects, recognizing actions, and comprehending events within video sequences. This multifaceted capacity empowers video surveillance systems, content summarization algorithms, and video indexing technology.

Natural capability of CNNs to automatically extract hierarchical features from grid-like data forms the foundatio upon which applications such as image recognition and object detection rest. The enduring evolution of CNN architectures and techniques promises to further accentuate their utility, forging new frontiers of application across diverse domains and industries. These networks are a will to the enduring symbiosis of science, engineering, and creativity in artificial intelligence and machine learning.

5.3.3 Layers Utilized in Constructing CNNs

CNN architecture, comprises a sequence of layers, with each layer effecting a transformation from one volume to another through a differentiable function. These layers, which can be categorized into distinct types, are integral components in the construction of CNN (Figure 5.4). To illustrate their functionality, consider running a CNNS on an image with dimensions $h * w * d$.

Input Layer: Serving as the initial point where input is fed into the model, this layer typically receives images or sequences of images. It takes in the raw input, such as an image with a width of w, a height of h, and a depth of d.

Convolutional Layers: These layers are responsible for feature extraction from the input dataset. They apply learnable filters, known as kernels, to the input images. These kernels are typically smaller matrices, such as 2×2, 3×3, or 5×5 in size. The filters traverse the input image data, computing the dot product between kernel weights and the corresponding input image patch. The result of this process is referred to as feature maps.

Activation Layer: By introducing an activation function to the output of the preceding layer, activation layers introduce nonlinearity to the network. These layers apply element-wise activation functions to the output of the convolution layer. Common activation functions include Rectified Linear Unit (RELU): max(0, x), Tanh, Leaky RELU, and more.

Pooling Layer: Periodically incorporated within CNNs, pooling layers serve to reduce the volume size, leading to faster computations, reduced memory usage, and the prevention of overfitting. Two prevalent types of pooling layers are max pooling and average pooling.

Flattening: Following the convolution and pooling layers, the resultant feature maps are flattened into a one-dimensional vector. This transformation enables them to be forwarded to a fully connected layer for classification or regression tasks.

Fully Connected Layers: These layers receive input from the preceding layer and are responsible for computing the final outcome for the classification or regression task at hand.

Output Layer: The output from the fully connected layers is subsequently subjected to a logistic function, such as sigmoid or softmax. This conversion process assigns probability scores to each class, serving classification tasks by determining the likelihood of the input belonging to a specific class.

In Chapter 6, we use CNN due to their remarkable capability to recognize and understand diverse features and patterns present in image data. This choice allows us to harness the power of CNNs to enhance our understanding and interpretation of the details within the images under consideration.

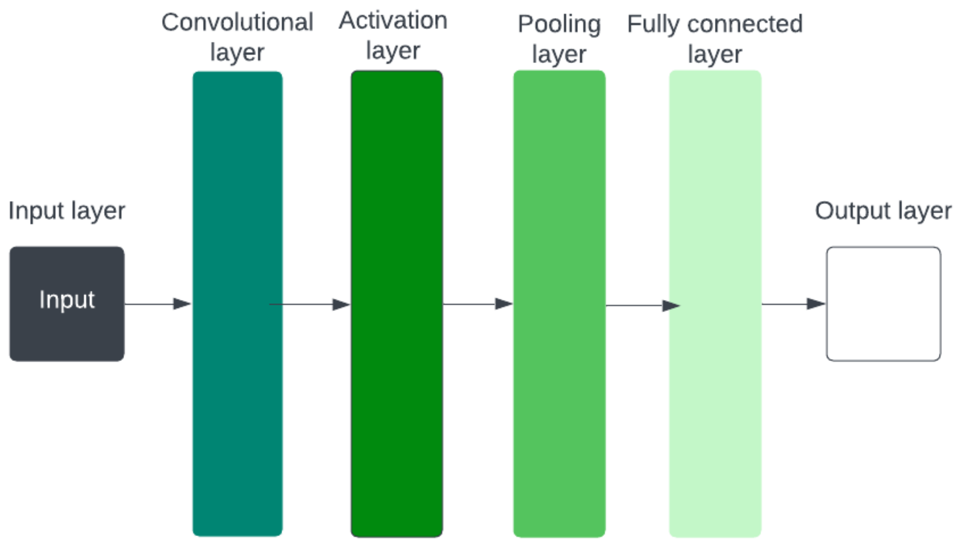


Figure 5.4: Simple CNN architecture

5.4 Combination of RNN and CNN

The synthesis of RNN and CNN [55] represents a potent architectural paradigm poised at the confluence of advanced deep learning techniques. This synergistic fusion of NNs combined with the respective strengths of RNNs and CNNs, creating a dynamic framework capable of effectively addressing the multifaceted challenges characteristic in processing a diverse array of data types.

5.4.1 Image Captioning

One of the most important applications of CNN-RNN hybrids is image captioning. Within this area, a CNN takes center stage as it undertakes the formidable task of image analysis. The CNN performs an complex examination of the image, extracting and encoding its visual features. These features, constituting the visual essence of the image, are then seamlessly propagated to a RNN architecture.

The RNN, equipped with its complex capacity for sequential data processing, subsequently assumes the role of generating textual descriptions of the image. This dynamic generative process is inspired by the contextual understanding of the image and its components. The utilization of an RNN in this context eceeds the mere act of labeling an image and ventures into the domain of automated image annotation, where the interplay between the visual and textual domains enriches the interpretation of images. This innovation finds applications in content-based retrieval systems, adaptive educational tools, and aids for the visually corrupt.

5.4.2 Video Analysis

In the complex landscape of video analysis, CNN-RNN hybrids rise to the occasion to use their capacity to seamlessly combine spatial and temporal analysis. This harmonious combination of CNNs and RNNs is especially characteristic in the domains of action recognition and video captioning.

The CNN component of the hybrid model assumes the role of a spatial feature extractor, parsing each individual video frame to uncover complex patterns and structural information. These spatial features encompass objects, scenes, and contextual elements within the frame. This spatial understanding serves as a critical foundation for the subsequent temporal analysis.

The RNBN component excels at capturing temporal dependencies within the video sequence. It processes the spatial features, sequentially modeling their evolution across time. As a result, the hybrid model not only recognizes actions and events within the video but is also well-poised to generate descriptive and contextually relevant captions for the video sequences.

This innovation bears profound significance in the domains of video surveillance, automated content summarization, and content indexing. It enables the automation of the cumbersome task of video annotation and indexing, providing insights into actions, events, and scenes without human intervention.

5.4.3 Neural Networks for Video Classification

Video classification entails attributing one or more labels to a video based on its content. The use of NNs for video classification [59] has gained prominence owing to their capacity to distinguish complex data patterns. CNNs prove particularly apt for video classification tasks, as they effectively capture both spatial and temporal features. CNNs comprise multiple layers of neurons trained to extract relevant features from images or video frames.

Traditionally, video classification NNs undergo training on an extensive dataset of labeled videos. These videos are divided into training and validation sets to measure model performance. Diversifying the training data is achieved by applying various data augmentation methods, such as flipping, rotation, and scaling. Furthermore, transfer learning accelerates the training process by leveraging pre-trained models previously trained on distinct datasets.

However, video classification poses challenges due to variations in lighting conditions, camera perspectives, and object appearances. Domain adaptation techniques address these issues by transferring knowledge from a source domain to a target domain.

video classification with NNs spans across various domains, including surveillance, entertainment, and healthcare. In the healthcare sector, video classification aids in the analysis of medical videos, such as endoscopic videos, assisting in the detection of anomalies and diagnosis. Assessing the performance of video classification using NNs can be measured through metrics like accuracy, precision, and recall. The selection of evaluation metrics hinges on the specific application and the relative importance of different types of errors. Video classification using NNs is a dynamically evolving field full with promising applications and associated challenges, spanning domains such as surveillance, entertainment, and healthcare.

5.4.4 Strengths and Significance

The integration of RNNs and CNNs within the CNN-RNN hybrid framework is an embodiment of the concept of synergy and complementarity. The strengths of CNNs in spatial feature extraction are harmonized with the RNN's ability to model temporal dependencies, leading to a holistic understanding of data that is invaluable in diverse applications.

This synergy allows hybrid models to develop a holistic understanding of data by fusing spatial features and temporal context. This facilitates a deeper and more context-aware analysis of complex information, especially in applications involving sequential and spatial data.

The cross-modal capabilities of CNN-RNN hybrids bridge the abyss between visual and textual domains, allowing for a comprehensive synthesis of both modalities. This is particularly beneficial in image captioning, where textual descriptions are generated directly from visual data, leading to enhanced human-computer interaction.

In the dynamic domain of video analysis, the hybrid models demonstrate their power in recognizing actions and events, offering insights critical to domains such as security monitoring, content analysis, and automated indexing.

The integration of RNNs and CNNs within the CNN-RNN hybrid model represents a remarkable advancement in artificial intelligence and machine learning. Their potential lies at the combination of deep learning and data analysis. The continued enhancement of these hybrid architectures is ready to uncover new dimensions in data processing and interpretation across a spectrum of applications in the fields of natural language processing, computer vision, and beyond.

In our thesis, the combination of RNNs and CNNs is leveraged in Chapter 6.

5.4.5 ResNet

Residual Networks (ResNet) [49] are a class of CNN architectures, originally designed for image classification, as illustrated in Figure 5.5. However, the versatility of ResNet extends to effective utilization in video classification by adapting it to process multiple frames within a video sequence.

Conventional CNNs involve incremental processing of each layer's output to extract increasingly complex features from the input image. Nevertheless, as network depth increases, the training process can become more formidable, potentially leading to the vanishing gradient problem. This issue surfaces when gradients used for weight updates dwindle significantly, hindering effective learning.

ResNet ingeniously addresses this concern by introducing residual connections between layers. These connections allow the network to effectively "bypass" certain layers, enabling information to traverse these layers with minimal alteration. This ingenious design circumvents the vanishing gradient problem, fostering more efficient learning.

In the context of video classification with ResNet, the network is applied to each frame within the video sequence, and subsequently, the outputs from these frames are merged to produce a final classification. The fusion of these outputs can be achieved through methods like averaging or by implementing an attention mechanism to emphasize the most salient frames.

A widely adopted implementation of ResNet for video classification is the Two-Stream ResNet. This architecture encompasses two distinct ResNet networks: one dedicated to processing spatial information, capturing the visual appearance of objects in the video, and the other tailored for processing temporal information, encapsulating the motion of objects within the video. The spatial network independently processes each frame, while the temporal network operates on pairs of frames to capture motion-related information. The outputs from both networks are subsequently integrated to yield a definitive classification.

Employing ResNet for video classification offers a effective approach due to its capability to extract various features and its incorporation of residual connections, effectively mitigating the vanishing gradient problem.

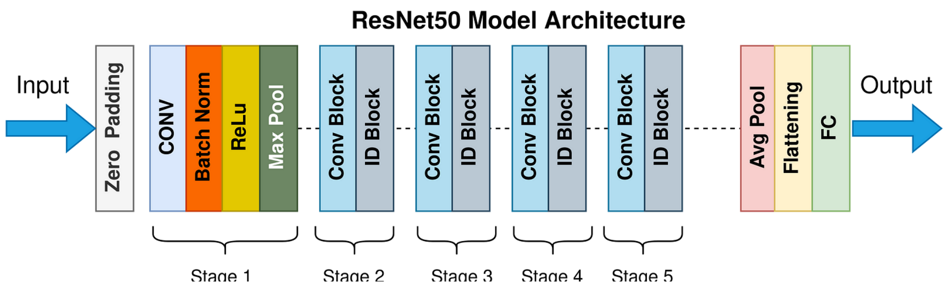


Figure 5.5: Architecture of ResNet50 [1].

5.4.6 EfficientNet

EfficientNet [119, 120] represents a family of CNNs meticulously engineered to maximize computational efficiency and parameter utilization in comparison to their predecessors. This achievement is accomplished through the adoption of an innovative scaling technique, which adapts the network architecture (as shown in Table 5.1) based on the available computational resources.

EfficientNet can be effectively employed for video classification by adapting it to process multiple frames within a video sequence. One approach involves the utilization of a 3D CNN architecture, capable of capturing both spatial and temporal information from the video frames.

To harness EfficientNet for video classification, we apply the 3D CNN to each frame within the video sequence and subsequently consolidate the outputs from these frames to arrive at a definitive classification. The consolidation process may involve techniques such as output averaging or the implementation of an attention mechanism to prioritize the most relevant frames.

One of the compelling advantages of using EfficientNet for video classification lies in its exceptional computational efficiency and efficient parameter utilization. This aspect is particularly crucial in scenarios with resource constraints, such as mobile devices or real-time video analysis. Additionally, EfficientNet demonstrates outstanding accuracy across a wide range of image classification tasks, promising applicability in video classification.

However, several challenges are associated with leveraging EfficientNet for video classification. Video classification often requires processing a significant number of frames, which can be computationally demanding. Furthermore, EfficientNet may not be as adept at capturing temporal information as specialized CNN architectures explicitly tailored for video classification, like the Two-Stream CNN or the 3D ResNet.

Employing EfficientNet for video classification presents an enticing strategy due to its computational efficiency and remarkable accuracy in image classification tasks. Nevertheless, fine-tuning and optimization may be necessary to achieve peak performance in video classification tasks.

Both of the architectures, ResNet and EfficientNet, are used in section 7 for video classification.

5.5 Random forest

Random Forest [22, 110], a powerful machine learning model, stands out as a prominent representative of ensemble learning techniques. It is particularly renowned for its ability to handle complex classification and regression tasks with a high degree of predictive

Stage	Operator	Resolution	Channels	Layers
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Table 5.1: EfficientNet v2 B0 architecture [119].

accuracy. At the heart of Random Forest's appeal lies its capacity to aggregate the predictions of multiple decision trees, resulting in a collective, consensus-driven model that overcomes the limitations of individual classifiers. The structural and operational principles of Random Forest are grounded in several key aspects. Decision trees serve as the foundational building blocks within the Random Forest model. Each decision tree is constructed through a recursive partitioning of the data into subsets, achieved by evaluating the values of input features. These partitioning decisions are strategically made to maximize the purity or homogeneity of each subset concerning the target variable.

Random Forest excels in forming an ensemble of decision trees, creating a diversified collection of classifiers. What sets this ensemble approach apart is its inherent randomness. Notably, both the data subsets and the feature subsets used for training are randomly selected with replacement, a technique commonly known as bootstrapping. In classification tasks, the individual predictions of decision trees are harmonized through a majority voting scheme, while in regression tasks, the predictions are collectively averaged. The final prediction of the Random Forest model emerges as the output of this voting or averaging process, benefiting from the wisdom of the ensemble.

A pivotal characteristic of Random Forest is the introduction of feature randomness. The model considers only a random subset of features at each split during the tree-building process. This feature selection approach significantly contributes to decorrelating the constituent trees within the ensemble, enhancing the overall model's robustness.

5.5.1 Algorithm: Sequential Steps

The Random Forest algorithm involves a series of steps to construct an ensemble of decision trees for both classification and regression tasks.

Step 1: Subsampling Data and Features

Within the Random Forest model, a random subset of data points and a random subset of features is chosen for the creation of each individual decision tree. Specifically, n random records and m features are selected from the original dataset, which comprises k data points.

Step 2: Construction of Individual Decision Trees

Each decision tree within the ensemble is independently constructed using the unique subset of data and features chosen in Step 1. These individual trees are designed to capture distinct patterns within the data.

Step 3: Output Generation from Decision Trees

Every decision tree in the ensemble generates an output or prediction based on its unique training subset. These outputs collectively form the basis for the ensemble's final prediction.

Step 4: Final Prediction via Majority Voting or Averaging

The ultimate prediction made by the Random Forest algorithm depends on the task at hand. For classification tasks, the final output is determined through majority voting, where the most frequently predicted class across the ensemble is selected. In the case of regression tasks, the final prediction results from averaging the outputs from all individual decision trees (Figure 5.6).

Random Forest exhibits several essential characteristics that distinguish it from other machine learning approaches.

- *Diversity*: Random Forest leverages diversity by not considering all attributes, variables, or features when constructing individual trees. This diversity ensures that each tree captures unique aspects of the data.

- *Immunity to the Curse of Dimensionality*: The algorithm is immune to the curse of dimensionality as each tree works with a reduced feature space, mitigating issues associated with high-dimensional data.

- *Parallelization*: Random Forest takes full advantage of parallelization, allowing each tree to be independently generated using different data and attributes. This parallel approach optimally utilizes computational resources.

- *Train-Test Split*: In Random Forest, there's no need for explicit data segregation into training and testing sets. Approximately 30% of the data is never seen by any individual decision tree, ensuring robust generalization.

- *Stability*: The stability of Random Forest stems from its reliance on majority voting or averaging. By aggregating predictions from multiple trees, the final output becomes more resilient to noise and fluctuations in the data.

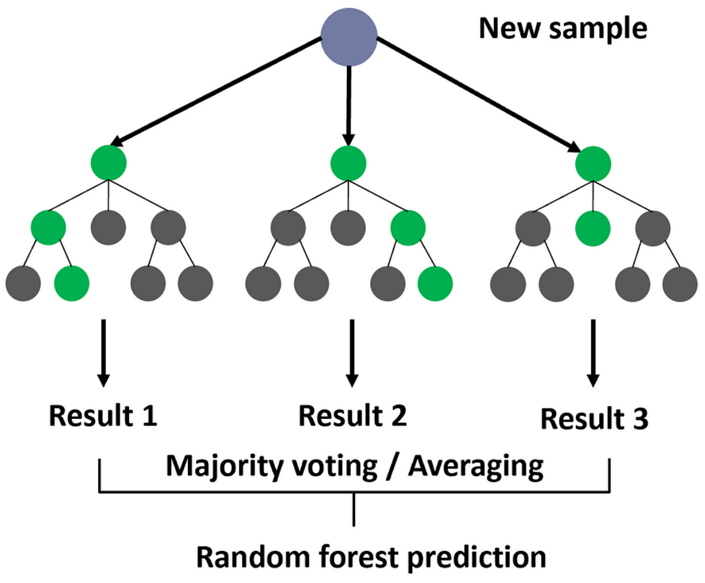


Figure 5.6: Random forest architecture

5.5.2 The Advantages of Random Forest

Random Forest's enduring popularity and effectiveness can be summarized to several keys [97]. The ensemble nature of Random Forest models is a natural defense against overfitting, ultimately leading to elevated predictive accuracy across a spectrum of diverse datasets.

The inherent randomness in feature and data selection, coupled with the diversity of constituent decision trees, bestows upon Random Forest models a notable level of robustness. This robustness enables the model to navigate the complexities of noisy and outlier-ridden data. Random Forest models contribute to the interpretability of machine learning results by providing feature importance scores. These scores offer insights into the contribution of each feature to the model's predictive power, facilitating model understanding and feature selection.

Random Forests have a remarkable aptitude for capturing non-linear and complex relationships in the data. This versatility makes them suitable for a wide array of tasks, transcending the boundaries of linear models. Random Forest models excel in their ability to efficiently handle missing data, maintaining robust predictive performance even when faced with data incompleteness.

5.5.3 Applications and Utility Across Domains

The versatility of Random Forest models translates into several applications across diverse domains and industries. Random Forests are harnessed for a wide spectrum of classification tasks, spanning image recognition, disease diagnosis, sentiment analysis, and more. Their consensus-based approach enhances classification accuracy. In regression problems, Random Forest models find application in real estate price prediction, financial forecasting, ecological modeling, and various contexts where precise numerical predictions are required.

Random Forests are instrumental in anomaly detection and outlier identification. They are commonly used in network security, fraud detection, and quality control scenarios. The model's ability to assign feature importance scores makes it a valuable tool for feature selection, streamlining the process of identifying relevant and influential features.

Bioinformatics, genomics, and proteomics are also the areas in which Random Forests are used. They aid in the analysis of biological data, contributing to insights in fields such as genetics and molecular biology. Within environmental sciences, Random Forest models serve in ecological modeling, climate modeling, and remote sensing data analysis. They are instrumental in understanding and predicting environmental phenomena.

Random Forest stands as a robust and adaptable machine learning model renowned for its high predictive accuracy and ability to address complex tasks. The ensemble approach, the introduction of feature randomness, and the model's versatility have positioned it as a preferred choice across a wide spectrum of applications in diverse industries.

5.5.4 Random Forests vs. Neural Networks

Random forests and neural networks are two of the most popular machine learning algorithms. While both can be used for classification and regression, they have key differences:

Random Forest	**Neural Network**
High accuracy	Potentially higher accuracy than RF in some cases
Reduced overfitting	Complex non-linear relationships
Interpretability	Suitable for high-dimensional data
Fast training	Black box nature - less interpretable
Linear relationships only	Slow and computationally expensive traing

Table 5.2: Characteristics of Random Forest and Neural Network models

Choosing between a random forest and a neural network depends on the specific task and available data. Random forests are a good option when interpretability and speed are crucial. Neural networks are better suited for handling complex non-linear relationships and high-dimensional data.

Beyond random forests and neural networks, other machine learning algorithms exist, like XGBoost and CatBoost. These algorithms share similarities with random forests but use different techniques to enhance accuracy and reduce overfitting.

CatBoost is used when dealing with a lot of categorical features. In our work, we use either numerical or visual data and therefore, this algorithm is not a good choice. While XGBoost excels at handling numerical data, it offers less interpretability compared to Random Forests. Since a key consideration in choosing a model is interpretability to counter the "black box" nature of neural networks, XGBoost might not be the optimal choice in this case. Nevertheless, XGBoost is one of the best decision trees based algorithms, therefore, we use it in this thesis.

Chapter 6

Determination of elasticity in straight canal

The advantage of computational simulation models lies in the access to comprehensive information about the motion of RBCs in the bloodstream, surpassing what can be recorded during experiments in physical devices. RBCs, being elastic entities suspended in blood plasma and constituting the predominant component of blood, are pivotal for accurate blood flow simulations.

In this chapter, we employ the position data of surface points on red blood cells in straight cannal as an input dataset for training neural networks. We present the results of utilizing neural networks to assess the elasticity of RBCs within the bloodstream from the numerical outputs of the simulation model.

The endeavor of determining the elasticity of RBCs using simulation experiments was a subject of exploration in the research by [16]. Their primary objective was the categorization of RBC elasticity into predefined classes. In this chapter, we extend this line of inquiry, transforming it into a regression task. Here, our goal is to determine the value of the regression function characterizing the elasticity of each individual RBC. Our ultimate, albeit distant, ambition is to ascertain the elastic properties, or elastic coefficients, of RBCs from video records of laboratory experiments. Within our numerical model, the elasticity of RBCs is governed by five parameters, meticulously detailed in Section 6.1.

For several reasons, including their robust physical interpretations and substantial influence on the overall elastic properties of RBCs, we single out the triangulation edge elasticity parameter (represented by the stretching coefficient k_s) as a pivotal factor. You can find the precise values of these coefficients in Table 6.1. In this chapter, we present a novel approach that employs machine learning methods to deduce the elastic parameters of the cell membrane.

Our innovative approach leverages data derived from computer simulations based on laboratory experiments. This data offers exceptional accuracy, rendering it apt for supervised learning. A notable advantage is that this dataset negates the need for manual annotations, as the computer can autonomously handle this task. By harnessing simulations, we can harness data suitable for input into neural networks. While we have the option to use exclusively the data acquired from video analyses of these experiments, we choose to also incorporate comprehensive simulation data for methodological comparisons. The aim is to evaluate the quality of elasticity coefficient estimations acquired from limited data compared to those obtained using the entire simulation dataset.

6.1 Simulation Input Settings

Current research explores the application of a physically informed neural network, as highlighted in prior works [69, 68]. In our thesis, we harness the implicit information of machine learning models, ensuring alignment with physical principles.

The source data for estimating the k_s coefficient via machine learning stem from several simulation experiments. These simulations are meticulously aligned with real experiments and have seen prior application in the study by [116].

In these simulations, a standardized channel was utilized, characterized by a cuboid shape with four walls measuring $60 \times 40 \times 40\, \mu m$. Periodic properties of the liquid flow were established in the direction of the x-axis. The fluid was discretized into a three-dimensional grid with a spatial resolution of $1\, \mu m$. The liquid possessed a kinematic viscosity of $1.3 \times 10^{-6}\, m^2/s$ and a density of $1.025 \times 10^3\, kg/m^3$. Interaction between the fluid and objects was maintained through a friction coefficient of $1.414\, N$, and external forces induced flow at values guaranteeing a maximum velocity of approximately $0.03\, m/s$.

Interaction among red blood cells, known as *cell-cell interaction*, was modeled using the *membrane_ collision* potential, characterized by parameters $mc_a = 0.01$, $mc_n = 1.0$, and $mc_{cut} = 0.4$. Interactions between the cells and the channel walls were represented by the $soft_{sphere}$ potential, defined by parameters $soft_a = 0.00035$, $soft_n = 1.0$, and $soft_{cut} = 0.5$.

The geometry of red blood cells was captured using a surface mesh, comprising 374 nodes. In their relaxed state, these cells assumed the characteristic biconcave shape, with dimensions of $7.82\, \mu m \times 7.82\, \mu m \times 2.58\, \mu m$, and a volume of $90.75\, \mu m^3$. Notably, the cells were filled with the same fluid as in their immediate surroundings.

To model the elastic properties of the cell membrane, we incorporated five distinct types of elastic forces. Each force corresponds to a specific modulus of elasticity and an associated parameter. These moduli of elasticity encompass the preservation of edge length, preservation of angles between adjacent triangles (modulus of bending), local modulus of area preservation, global modulus of area preservation, and volume preservation modulus. For detailed values of the coefficients, please refer to Table 6.1.

Table 6.1: Overview of used simulation parameters.

coefficient	value
coefficient of elasticity (k_s):	various
bending coefficient (k_b):	$3 \times 10^{-19}\, Nm$
local area conservation coefficient (k_{al}):	$2 \times 10^{-5}\, N/m$
global area conservation coefficient (k_{ag}):	$7 \times 10^{-4}\, N/m$
volume conservation coefficient (k_v):	$900\, N/m^2$

The same channel and fluid flow parameters were used in all simulations. The simulation channel had the shape of a cuboid with four walls of dimensions $60 \times 40 \times 40 \mu m$. To ensure periodic properties of the liquid in the direction of the x-axis, the fluid was discretized into a three-dimensional grid with a spatial step of $1 \mu m$. The kinematic viscosity of the liquid was $1.3 \times 10^{-6}\, m^2/s$, and the density was $1.025 \times 10^3\, kg/m^3$. To facilitate interaction between the fluid and objects, a coefficient of friction of $1.414\, N$ was utilized. External forces were applied to set the flow in motion, ensuring a maximum velocity of approximately 0.03 *m/s*.

Cell-to-cell interactions, referred to as *cell-cell interaction*, were modeled using the *membrane_collision* potential with parameters $mc_a = 0.01$, $mc_n = 1.0$, and $mc_{cut} = 0.4$. Interactions between cells and the channel walls were described using the *soft_sphere* potential with parameters $soft_a = 0.00035$, $soft_n = 1.0$, and $soft_{cut} = 0.5$. Elastic coefficients for healthy RBCs used are listed in Table 6.2.

With malaria disease in mind, we focused on nine levels of RBC elasticity. First level represent healthy RBCs, with the most elastic RBCs having a stiffness coefficient (k_s) of 0.005, and the least elastic RBCs with $k_s = 0.3$, which represent super-sfiff RBCS. Malaria-infected cells at stage 3 of the disease have a k_s value of 0.03 (The value of $k_s = 0.03$ was chosen based on the reduced elasticity observed in malaria-infected cells at stage 3, as determined by an optical tweezers stretching experiment [118]). The remaining two levels of RBC elasticity were distributed between healthy, malaria-infected and super-stiff RBCs.

6.2 Description of Obtained Simulation Data

The initial arrangement of the platelets was random in the simulations, and the number of simulation steps recorded was 3400 for *Sim*3*a*, *Sim*3*b*, and *Sim*3*c*, and 1240 steps

Table 6.2: Overview of k_s values in each simulation.

Simulation name	values of elasticity coefficients (k_s)
*Sim*3*a*	0.3, 0.005, 0.03
*Sim*3*b*	0.15, 0.015, 0.009
*Sim*3*c*	0.225, 0.1, 0.05
*Sim*9*a*	0.005, 0.009, 0.015, 0.03, 0.05, 0.1, 0.15, 0.225, 0.3
*Sim*9*b*	0.005, 0.009, 0.015, 0.03, 0.05, 0.1, 0.15, 0.225, 0.3

To note

for *Sim*9*a* and *Sim*9*b*, which corresponded to RBC movement in the channel of about 5.5 mm. The recorded steps corresponded to every 2000 steps of the simulation, an internal step of the simulation detailed enough for method training. Due to the need to stabilize the flow in the run-up part of the experiment, the first 300 records of the simulation outputs were not used. To create a balanced dataset, each type of red blood cell, categorized by elasticity, was represented by an equal amount of data. In the *Sim*3 experiments, 9 types of elasticity were simulated, each with 6 red blood cells, and each type had 3100 records. For *Sim*9, there were also 9 different types, with 6 blood cells of each type and 940 records. In total, 54 red blood cells were simulated in both cases, *Sim*3 and *Sim*9. A similar procedure was employed in the study [16].

In each internal simulation step of the ESPreSso module, the current position of each non-stationary point in the blood flow, and therefore all red blood cell triangulation points, are calculated. Given the vast range of this data, basic information about the position and velocity of vital points of each blood cell was usually stored, including the simulation step (cycle) number, the coordinates of the center of the simulated cell $[x, y, z]$, the velocity of the center of the cell determined by its components in the direction of the axes x, y, z, coordinates x, y, z representing the extreme points of the triangulation of the cell (according to minimum and maximum coordinates along each axis, as shown in 6.1), and the velocities of the extreme points of the cell determined by its components in the direction of the x, y, z axes, as well as the volume or surface area of the cell.

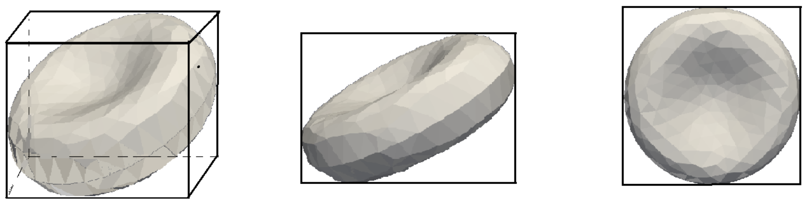


Figure 6.1: 3D RBC covering cube from the simulation and two 2D RBC covering rectangles from the video.

When selecting and editing the data, we considered what information we can obtain from the actual video footage. While the information contained in the simulation outputs allows us to determine the extreme points of the red blood cell in a real experiment, other parameters such as the center of the cell, the speed of its movement, the movement of the extreme points, the volume, or the total surface area of the cell are much more challenging to obtain, if at all. Therefore, in our work, we focused on using data that represents the projection of information from 3D into two-dimensional space along the xy and xz axes. By subtracting the positions of the extreme points in each axis's direction, we could also derive information about the size of the rectangle "bounding box" (or cube) that encloses the blood cell at each time step 6.1.

In the individual computational experiments intended for training machine learning methods, we generally employed the following types of simulation data modifications for input to the neural network:

- data to 2D according to xy axes
- data to 2D according to xz axes
- double projection of data into 2D along the xy and xz axes
- use of full 3D data along all three xyz axes

Table 6.3: Overview of used data sets.

dataset_xyz	dataset_xy_xz	dataset_xz	dataset_xy
cuboid_x_min (x,y,z)	cuboid_x_min (x,y,z)	cuboid_x_min (x,z)	cuboid_x_min (x,y)
cuboid_x_max (x,y,z)	cuboid_x_max (x,y,z)	cuboid_x_max (x,z)	cuboid_x_max (x,y)
cuboid_y_min (x,y,z)	cuboid_y_min (x,y)		cuboid_y_min (x,y)
cuboid_y_max (x,y,z)	cuboid_y_max (x,y)		cuboid_y_max (x,y)
cuboid_z_min (x,y,z)	cuboid_z_min (x,z)	cuboid_z_min (x,z)	
cuboid_z_max (x,y,z)	cuboid_z_max (x,z)	cuboid_z_max (x,z)	
x_x_size	x_x_size	x_x_size	x_x_size
y_y_size	y_y_size		y_y_size
z_z_size	z_z_size	z_z_size	

In each option, data along the x-axis was utilized because the fluid in the channel flows primarily along this axis, making the effect of cell elasticity most observable. However, data along all three axes was also used to verify the accuracy of the neural network models and to determine how effectively we could extract information about RBC elasticity from full simulation data.

6.2.1 Data Preprocessing and Augmentation

The dataset obtained from simulation experiments needed transformation to be suitable for training the machine learning model.

Given that the simulation data aims to replace real video information, cell velocity information is implicitly included based on the change in x-coordinate values in consecutive records. Thus, the resulting dataset was divided into time windows, equivalent to a sequence of consecutive simulation records in video processing. The sequence length, represented by the window size w, becomes a crucial hyperparameter for the model.

In computational experiments involving the training of individual neural networks, window sizes from the set $w = \{5, 10, 20, 30, 40, 50\}$ were used. After selecting w, the entire dataset was divided into time sequences of size w, resulting in the training data (x_{wi}, y_i), where x_{wi} represents the time sequence, and y_i is the actual value of the elasticity coefficient k_s for that sequence.

To expedite the model training process, we applied standardization and normalization techniques, as described in [85]. Standardization was performed on the values of the extreme points along the y and z axes and the dimensions of the bounding rectangle (or cube). This transformation ensured that the data had a mean of 0 and a standard deviation of 1. We then adjusted attributes separately for each training example containing information about the coordinates of the extreme points along the x axis. For each record within the time window, we subtracted the minimum value of the x_i attribute from the x_i attribute, resulting in the following normalization equation:

$$x_{ij\_transformed} = x_{ij} - \min(x_i) \tag{6.1}$$

This approach allowed us to normalize the training examples while implicitly preserving information about the speed of red blood cell movement.

The data created underwent subsequent division into three parts:

- *Training data* - used for model training.
- *Validation data 1* - utilized for model validation after each epoch.
- *Validation data 2* - data used to compare different models.

This partitioning was performed to avoid data leakage. Data leakage occurs when information used for training a machine learning model is also used for validation or final testing. To prevent this, the model was trained on one set of data, while validation and testing were carried out on separate datasets.

In order to augment the training data for machine learning, the original positions were transformed by adding noise. Noise (n) was generated from a random normal distribution with a mean of 0 and standard deviation of 1, and was multiplied by a constant of 0.1 for each component of the training example. Additionally, a random shift (s) was generated from a uniform distribution within the interval $(-0.25, 0.25)$ for each training example. The number of expansions (a) was determined as follows:

$$a = \frac{10000}{\frac{3100-50}{w}} - 1 \tag{6.2}$$

and then a is rounded to an integer. The amount of training data thus increased to approximately 380,000 examples.

6.2.2 Types of Neural Network Architectures Tested

LSTM

The LSTM (Long-Short Term Memory) architecture [135] is typically employed for input data in the form of time sequences. Our network using this architecture consisted of four layers with 512, 64, 32, and 10 hidden neurons, respectively. It utilized a hyperbolic tangent activation function and a recurrent sigmoid activation function. Each LSTM layer was followed by a dropout layer in which 10% of the neurons were dropped out. The output was then flattened (*keras.layers.Flatten()*) and passed through a pair of fully connected layers, one with 1024 neurons, another with 512 neurons, both using a ReLU activation function. Finally, a fully connected output layer with a single neuron and a linear activation function served as the output layer.

CNN-LSTM

The CNN-LSTM architecture combines CNN convolutional layers for feature extraction from input data with LSTM for sequence prediction. The combination of these layers is motivated by research [95] and studies such as [63] that suggest improved LSTM performance with this architecture. It is suitable for problems that involve data with spatial structures (e.g., image pixels) or time-structured inputs/outputs (e.g., video frames or text).

We implemented two versions of the CNN-LSTM network. The first, *CNN-LSTM Conv1D*, used 1D convolution while traversing the time sequence. The second, *CNN-*

LSTM Conv2D, used 2D convolution. To reduce input data variance, we passed the input through two convolutional layers with 256 filters, a step size of 1, and applied ReLU activation. For *CNN-LSTM Conv1D*, the filter size for both layers was three times the width of the time window. For *CNN-LSTM Conv2D*, the filter size was three times the width of the time window for the first layer and 4x3 for the second layer. Following the convolutional layers, the result was adjusted by pooling with a 2x2 filter size. The multivariate intermediate result was flattened (*keras.layers.Flatten()*) and passed to an LSTM layer with 256 hidden neurons and a ReLU activation function. Finally, the output from the LSTM layer was connected to a fully connected layer with 512 neurons using ReLU activation, and the network was terminated with a linear output layer containing one neuron.

The network architectures used in our experiments are illustrated in 6.2.

All experiments were conducted using Python 3.8, with the Tensorflow (Keras) [3] library utilized to construct the neural networks. Training was performed on a computer equipped with an AMD Ryzen 55600H processor, Radeon graphics card, 16 GB of RAM, and an NVIDIA GeForce RTX 3060 Laptop graphics card.

6.3 Result Examples of Using CNN-LSTM Networks for Simulating Three RBC Types

In total, we trained 84 different neural network models for each combination of architecture type (3 types), data (4 data sets used), and w window size (7 options) with $Sim3$ data. The mean absolute percentage error (MAPE) was used as the loss function, calculated as:

$$MAPE(y_{\text{true}}, y_{\text{pred}}) = 100 \times \left| \frac{y_{\text{true}} - y_{\text{pred}}}{y_{\text{true}}} \right| \tag{6.3}$$

Figure 6.3 displays the MAPE values for each possible combination of the options mentioned. The graph illustrates that as the size of the training time window (w) increases, the MAPE also increases, possibly due to noise and distortion introduced. The selection of MAPE as the loss function is selected over other loss functions due to prioritized capturing of the relative difference between true and predicted coefficients. Unlike Mean Absolute Error (MAE) or Mean Squared Error (MSE) which deal with absolute differences, MAPE expresses error as a percentage of the actual target value. This makes it scale-invariant, meaning the error is independent of the overall magnitude of the target variable.

The MAPE falls within the 20% range for data simulating information from video recordings, but combining these data can halve the error. Data from all three xyz axes provides the lowest error, at 5%, validating the hypothesis that RBC elasticity information can be obtained from the used data. Table 4 lists MAPE values by architecture and data subset for the w hyperparameter with the lowest error. The most accurate neural network model was *CNN-LSTM Conv2D* for data along the xy, xyz, xy_xz axes with w values successively 3, 3, 5. For xz axes, LSTM with $w = 5$ was the best. *CNN-LSTM Conv2D* with $w = 3$ achieved the lowest MAPE of all models, at 4.58%. For 2D projection, LSTM with $w = 5$ performed the best for data along the xz axes, with a MAPE value of 20.46%. When combining two 2D projections, *CNN-LSTM Conv2D* with $w = 5$ resulted in a MAPE value of 7.97%.

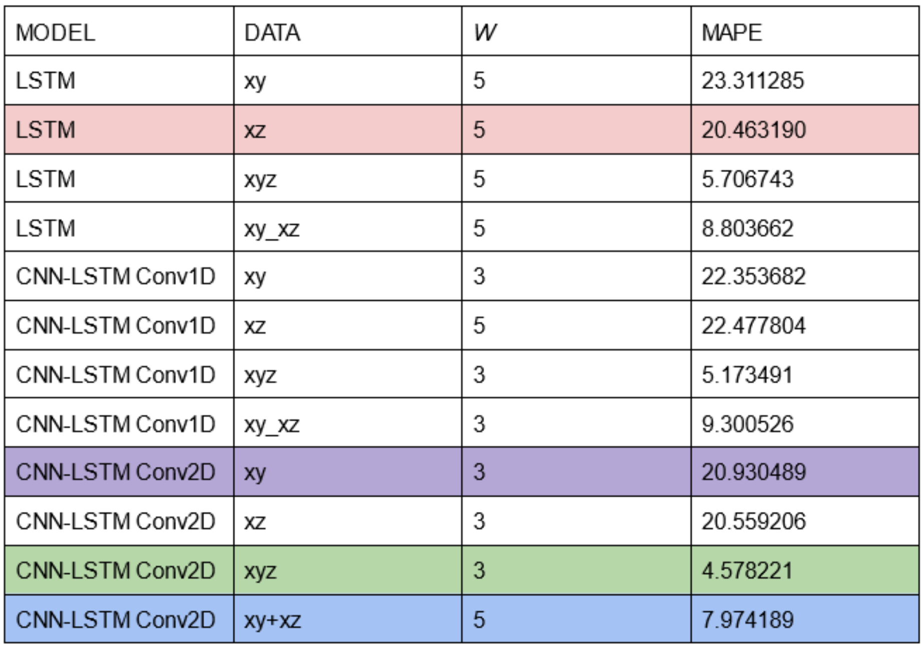
The distribution of MAPE for all simulated types of elastic blood cells can be observed in 6.4 for *CNN-LSTM Conv2D* xyz with $w = 3$ and 6.5 for *CNN-LSTM Conv2D* xyz, $w = 3$ and *CNN-LSTM Conv2D* xy_xz with $w = 5$. The green triangles represent the average MAPE for each value of the elasticity coefficient k_s, and the yellow line indicates the median. Outliers are also shown in figures (b) and (d).

The largest mean MAPE values are observed at elasticity values of 0.03 and 0.1 for *CNN-LSTM Conv2D* models xyz with $w = 3$ and xy_xz with $w = 5$, as shown in 6.6 and 6.7. For blood cells with elasticity values of 0.03, the predicted value was around 0.009, and similarly, for elasticity values of 0.1, a substantial number of predictions fell in the range $(0.005, 0.03)$.

Table 5 lists MAPE values by architecture, data subset, and the value of the hyperparameter w for which the resulting model achieved the lowest error. The most accurate neural network model was *CNN-LSTM Conv2D* for data along the xy, xyz, xy_xz axes, with w values of 3, 3, and 5, respectively. For data from the xz axes, the best model was *LSTM* with $w = 5$. *CNN-LSTM Conv2D* with $w = 3$ achieved the lowest MAPE value among all the trained models, at 4.58%. For 2D projection, *LSTM* with $w = 5$ achieved the best performance for data along the xz axes, with a MAPE value of 20.46%. When combining two 2D projections, *CNN-LSTM Conv2D* with $w = 5$ resulted in a MAPE value of 7.97%.

The results of the experiment described in subsection 6.3 indicate that it might be possible to determine the elasticity of the red blood cell from the data we can obtain from video recordings of blood flow in microfluidic devices. However, if we use only one view of blood flow, e.g. along the xy axes, the resulting prediction will contain a relatively large error, more than 20% on average. If an image from two sides, xy and xz, is used, the prediction achieves an average error of less than 8% of the true value. Such a result would be difficult to obtain from data measured in a real blood flow experiment. In the simulation experiment, we have all values of coordinates x, y, z for all discretization points of the RBC surface. Thanks to this, we can estimate how much the real blood cell deviates from its video recording. By substituting an estimate of the elasticity parameter k_s we can estimate the error we made compared

Table 6.4: MAPE values by architecture, subset of data used for training, and hyperparameter w values. The lowest (best) value for each data subset is highlighted.



MODEL	DATA	W	MAPE
LSTM	xy	5	23.311285
LSTM	xz	5	20.463190
LSTM	xyz	5	5.706743
LSTM	xy_xz	5	8.803662
CNN-LSTM Conv1D	xy	3	22.353682
CNN-LSTM Conv1D	xz	5	22.477804
CNN-LSTM Conv1D	xyz	3	5.173491
CNN-LSTM Conv1D	xy_xz	3	9.300526
CNN-LSTM Conv2D	xy	3	20.930489
CNN-LSTM Conv2D	xz	3	20.559206
CNN-LSTM Conv2D	xyz	3	4.578221
CNN-LSTM Conv2D	xy+xz	5	7.974189

to a situation where we would have had complete information. Our results show that when solving the k_s parameter estimation problem, we get better results for videos from multiple symmetry axes of the monitored channel.

6.4 Using a Regression Neural Network to Classify Red Blood Cell Elasticity

As discussed in the introduction, determining RBC elasticity as a classification problem has been previously explored. To compare the results, the output of the regression method was transformed into a simple classifier using the same simulation data as in Section 6.3 (*Sim3a-c*).

Two approaches were used to classify blood cells: training a classification neural network and directly using the output of the regression neural network. The latter method involved averaging the elasticity coefficients and assigning the result to the appropriate category. Categories were created by establishing eight thresholds among the nine elasticity values, with each boundary located midway between two adjacent coefficients in ascending order. This neural network is referred to as *RegToClass*.

The classification neural network model shared the same architecture as the regression neural networks, with the only alteration being in the output layer. Here, one output neuron was replaced by nine to represent the nine categories, and the activation function was switched from linear to *softmax*. The loss function was also changed to categorical cross-entropy, and the target variables were converted to one-hot encoding.

Table 6.5 presents the classification performance for the best models, as determined in the 6.4. Overall, better classification performance was achieved for the models optimized directly for the classification task. The largest improvement was almost 11.75% for the model trained on data representing the projection to 2D along the xz axes, while the smallest improvement was 5.84% for the model with xyz. A noticeable bias was observed for blood cells with an elasticity coefficient value of 0.03, where the predicted class was 0.009, particularly for the model trained on the xz data. Lower classification success was also seen for coefficient values of 0.225 and 0.3, indicating that predicting the class is challenging for very rigid blood cells with small differences in elasticity.

Table 6.5: Comparison of classification accuracy of RegToClass and Classification neural networks.

Window Size w	Data Subgroup	RegToClass	Classification
3	xy	69.83%	80.92%
3	xz	71.84%	83.59%
3	xyz	93.73%	96.87%
5	xy_xz	88.83%	94.67%

6.5 Validation of Models on Different Simulations

In previous experiments, three simulations were utilized, each divided into training, validation, and testing parts. To further investigate model errors and potential overfitting, a different dataset compilation was employed in this experiment:

- Training: One simulation with 9 types of cells based on elasticity coefficient values ($k_s = \{0.005, 0.009, 0.015, 0.03, 0.05, 0.1, 0.15, 0.225, 0.3\}$).
- Validation: One simulation with 9 cell types (same as model training but with different initial seeding) and the same simulation parameters.

The previous experiments indicated that the best-performing data subsets for the model were those along the xy and xz axes simultaneously, excluding the xyz subset, which is not practical to obtain in real experiments. For this specific combination of the model and data subset (xy_xz), models were trained with varying window lengths w. As in previous experiments, data preprocessing was conducted, and data was augmented for the neural network. The resulting MAPE values can be observed in 6.5.

The validation results revealed a significant deterioration in model performance, despite efforts to mitigate overfitting by adding extended noise and dropout data. This decline could be attributed to the data, specifically the components containing information about the y and z axes. The neural network may have overfit by focusing on this portion of the data and predicting the elasticity coefficient based on the cell's position in the channel. In the prior experiment, the entire trajectory of red blood cells was divided into segments with a specified number of records equal to the parameter w and then split into subsets for random training and validation. This information was present in both data sets, which accounted for the superior model performance.

A noteworthy finding was the significantly improved performance of the *CNN_ LSTM _ Conv2D* model in all experiments. Unlike the original *CNN_ LSTM_ Conv1D* architecture from the article [132], this model incorporated a convolution filter of size (4, 3), as opposed to (number of features, 3). This outcome reinforces the notion that CNNs are among the most potent neural network architectures today.

The code is available at [84].

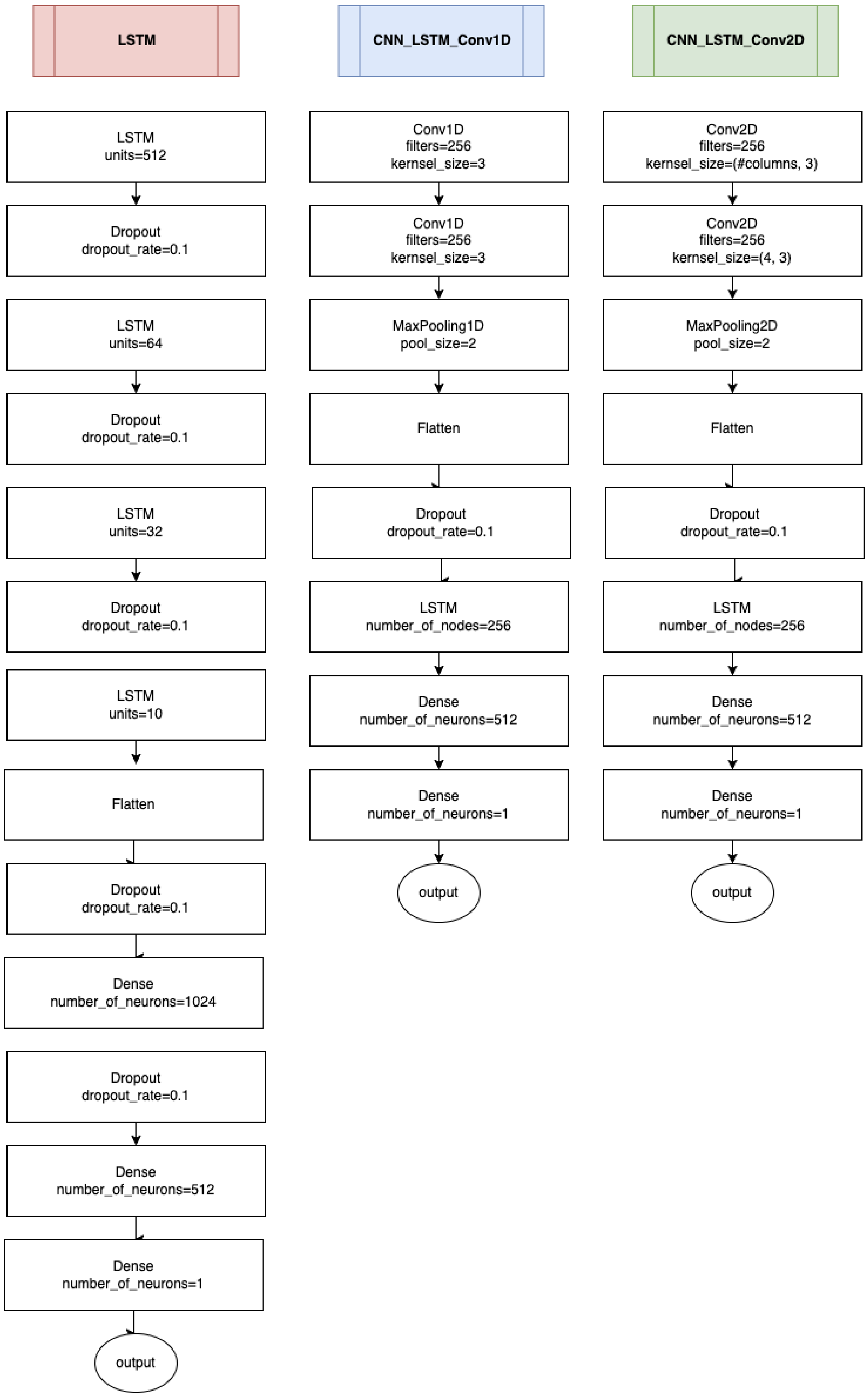


Figure 6.2: Diagrams of the neural network architectures used in the experiments.

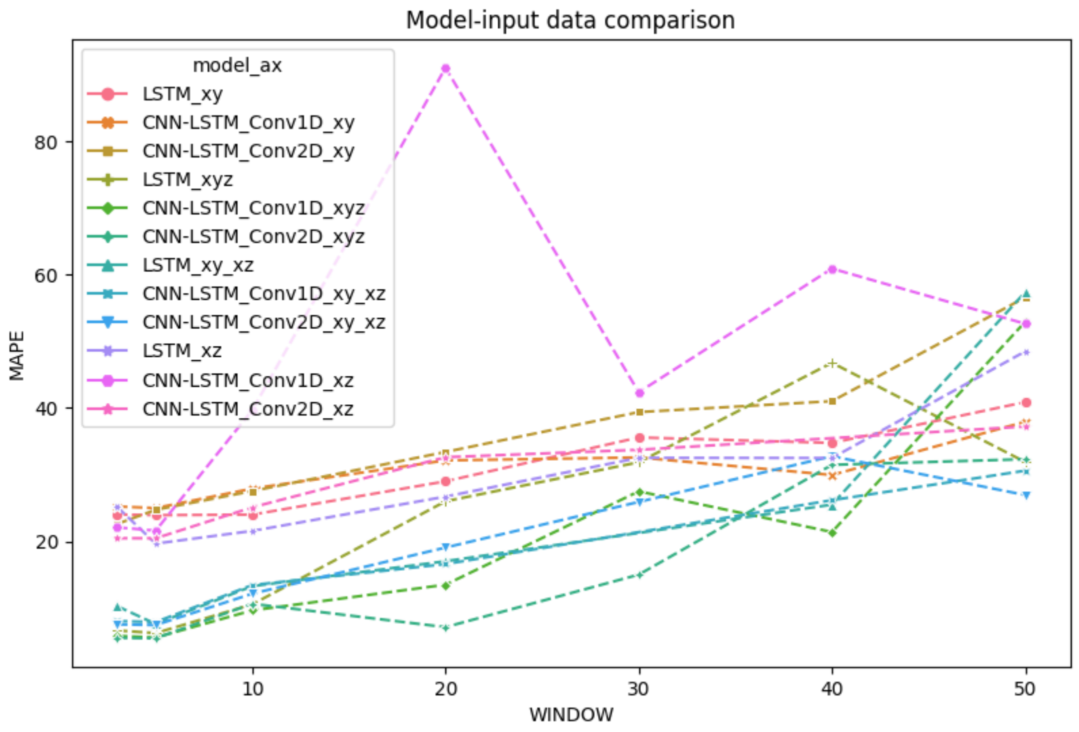


Figure 6.3: Comparison of training MAPE for each combination of w, data subset, and NN model type.

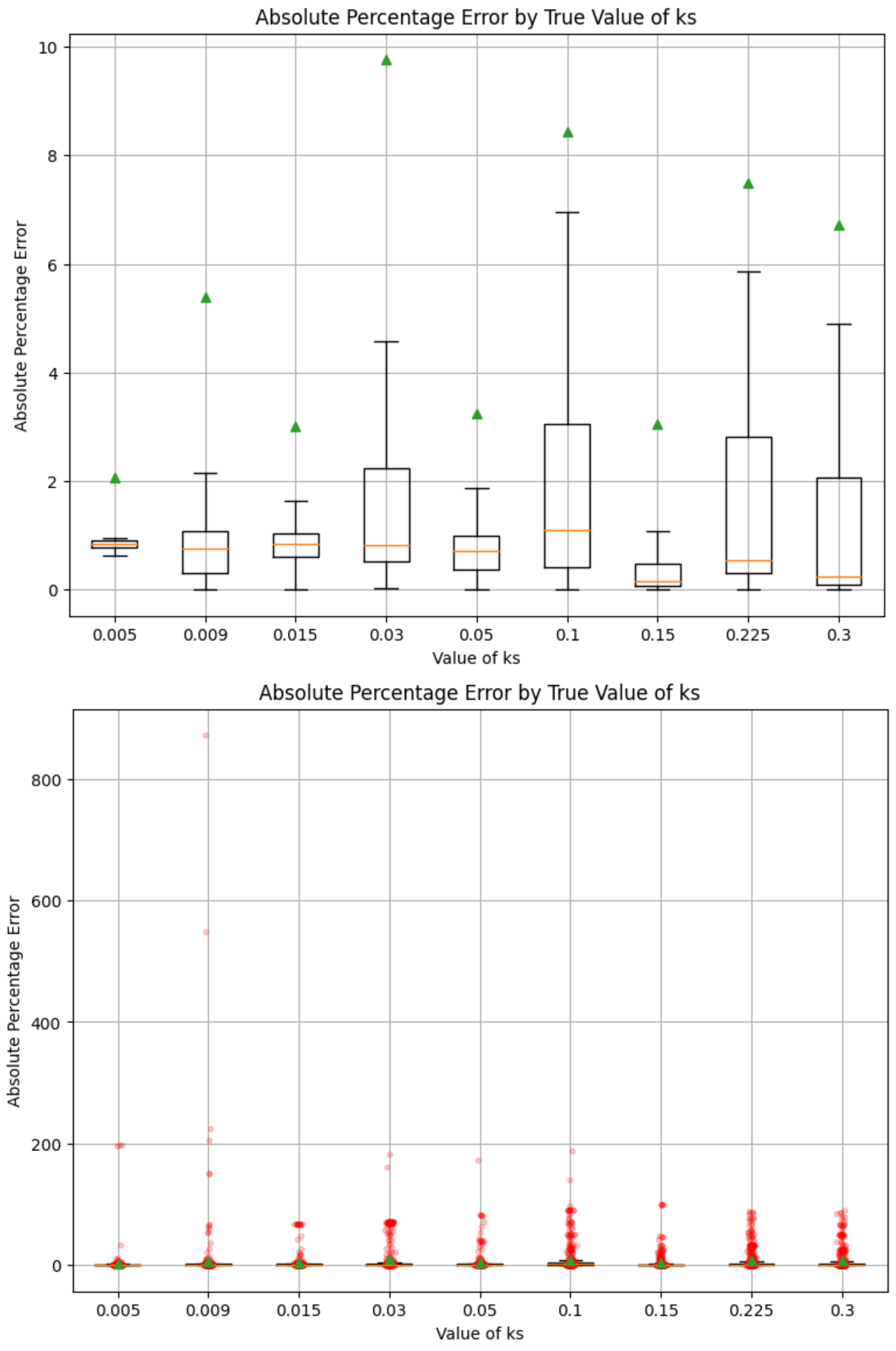
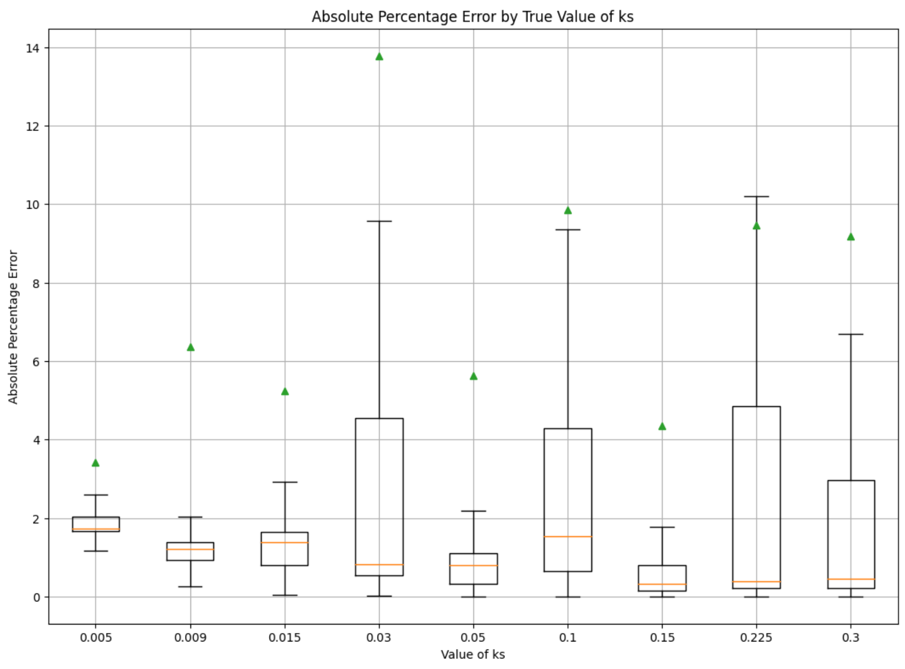


Figure 6.4: MAPE boxplots for each RBC elasticity type for the *CNN-LSTM Conv2D* architecture, xyz data subset used with $w = 3$ without (up) and with outliers (down)



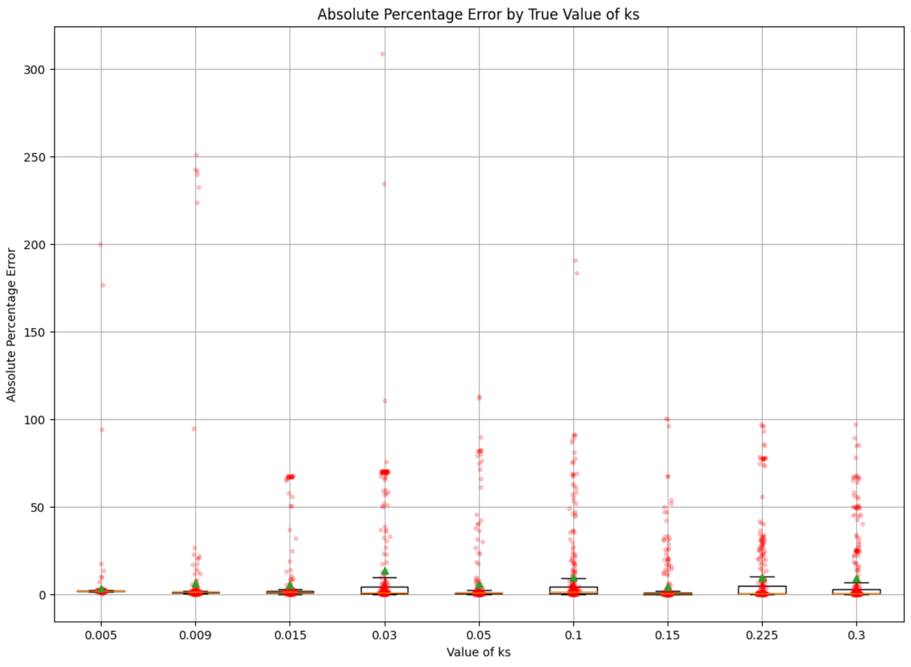


Figure 6.5: MAPE boxplots for each RBC elasticity type for the *CNN-LSTM Conv2D* architecture, xy_xz data subset used with $w = 5$ without (up) and with outliers (down)

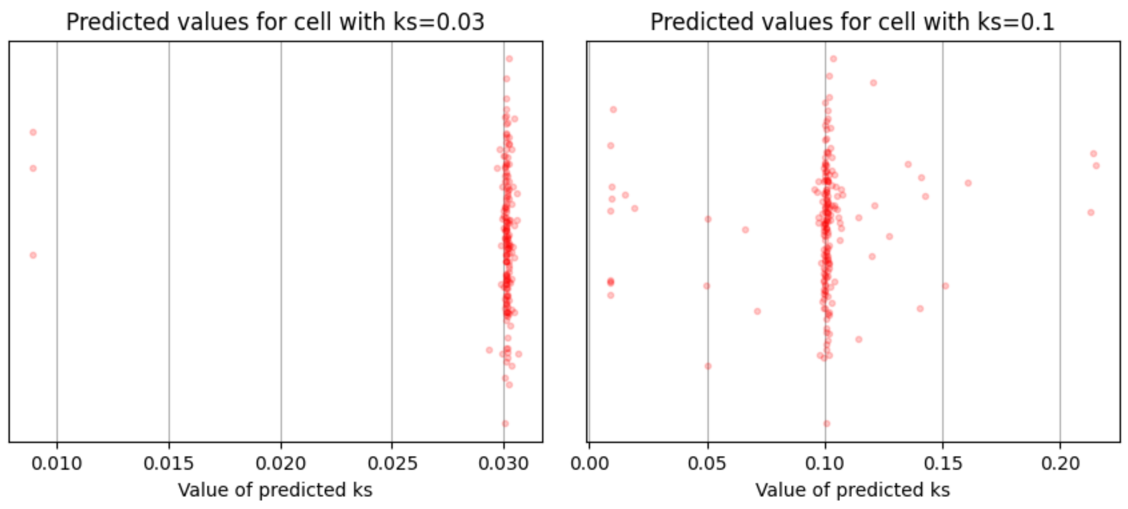


Figure 6.6: Highest average MAPE for elasticity 0.03 and 0.1 for architecture *CNN-LSTM Conv2D*, used data subset xyz with $w = 3$.

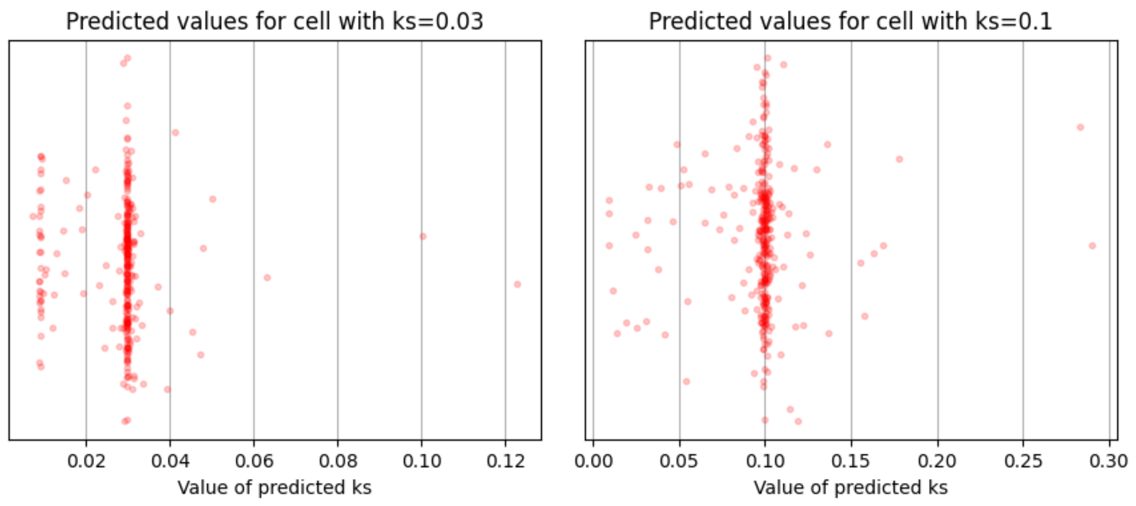


Figure 6.7: Highest average MAPE for elasticity 0.03 and 0.1 for architecture CNN-LSTM_Conv2D, used data subset xy_xz with $w = 5$.

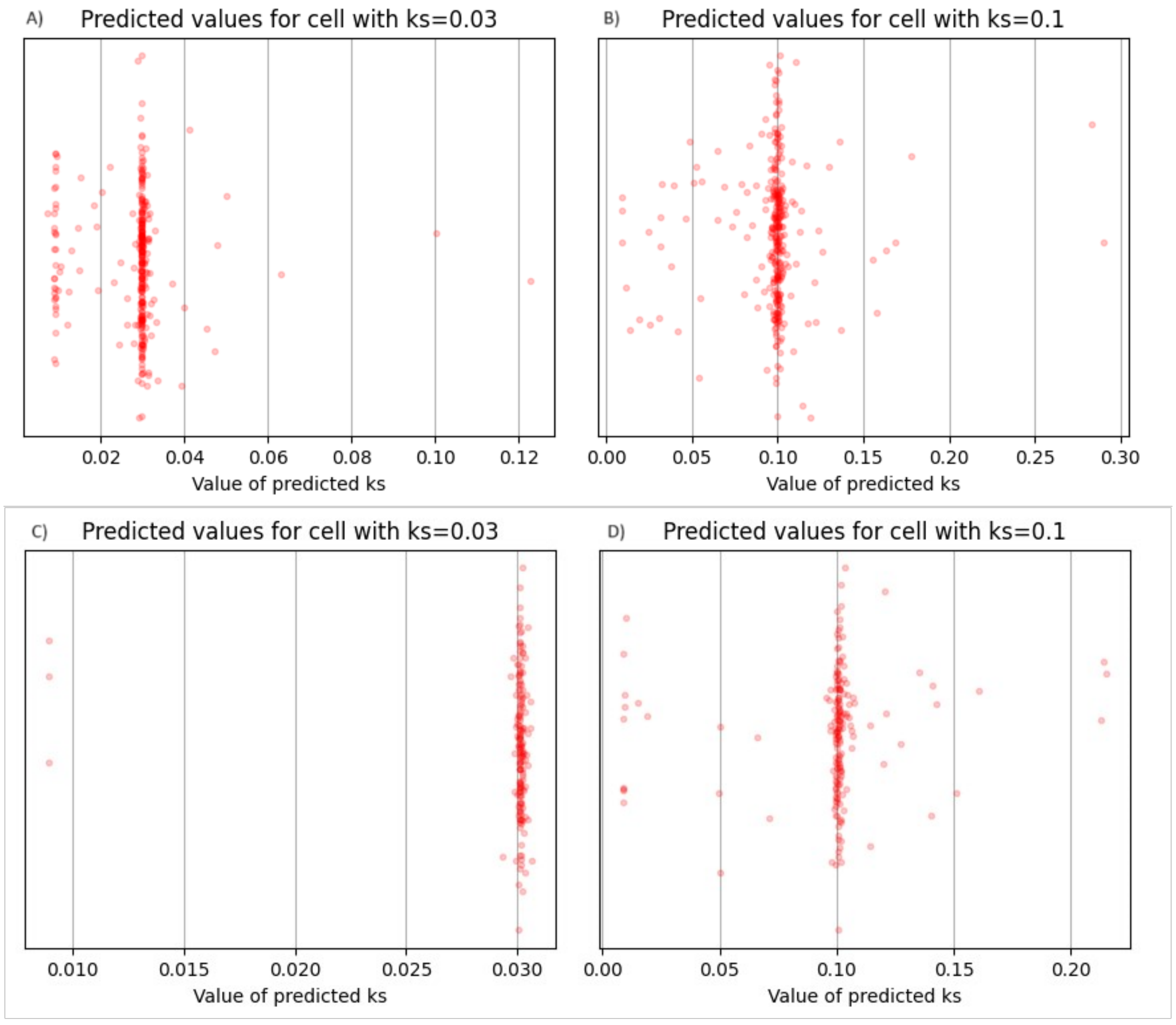
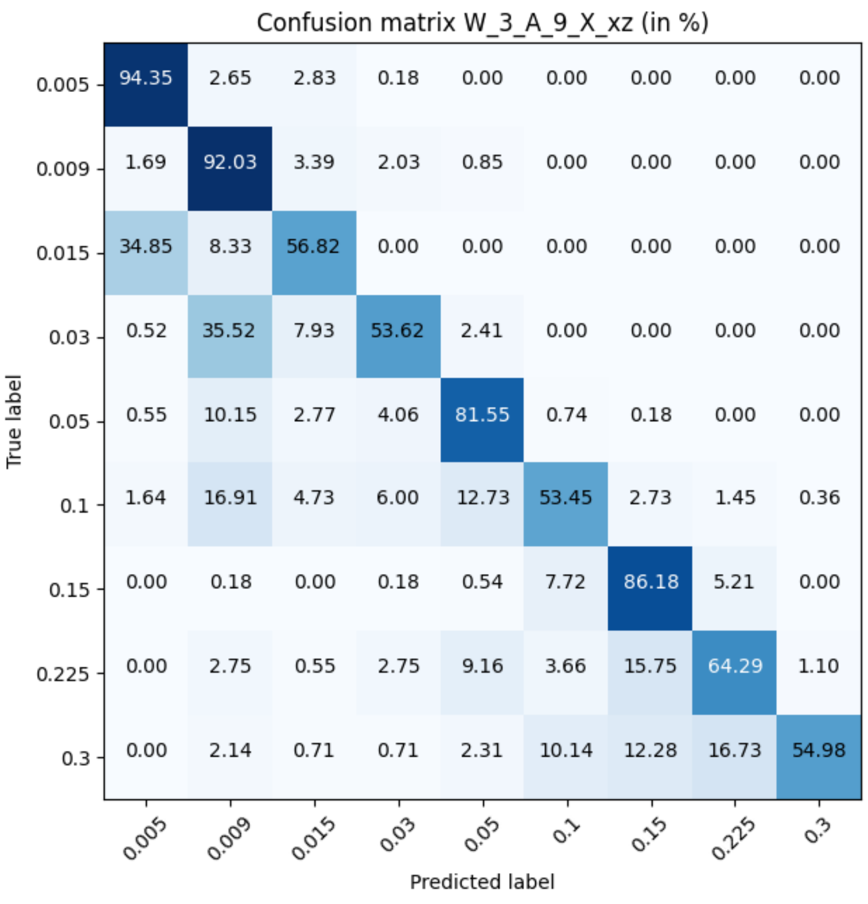
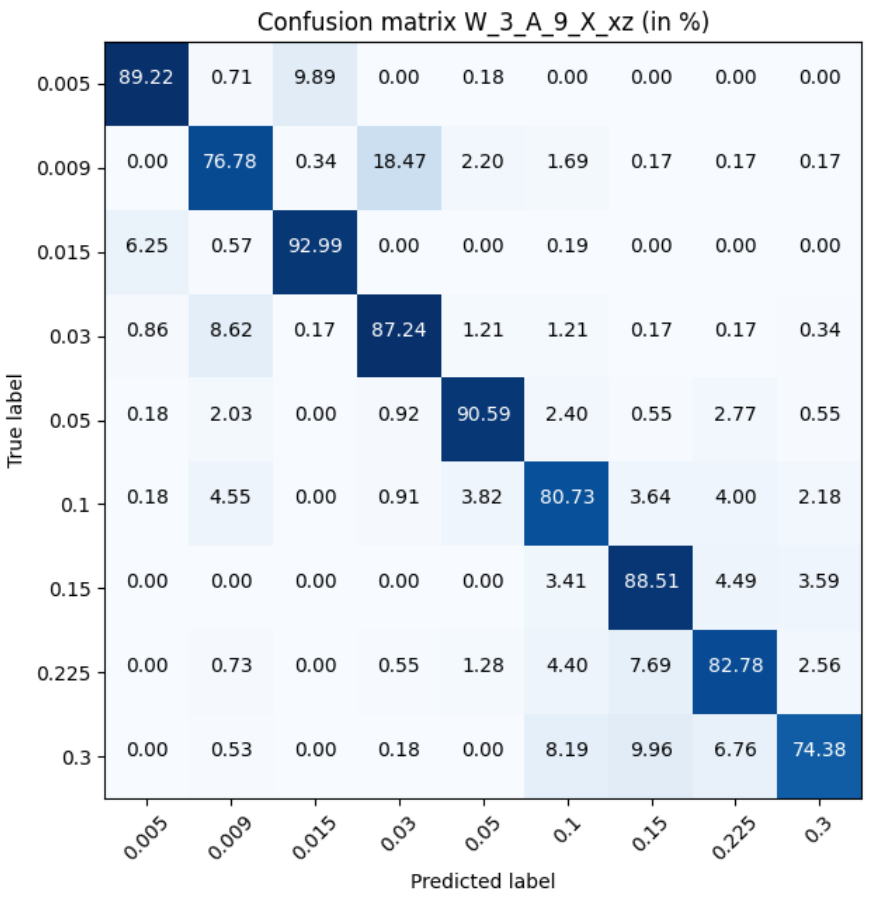


Figure 6.8: Distribution of predicted k_s values for different data subsets and elasticity values.

This figure shows the distribution of predicted stiffness coefficient (k_s) for the CNN-LSTM Conv2D architecture. Subplots (A) and (B) depict data for RBCs with elasticity 0.03 and 0.1, respectively, using the xy_xz data subset with window size $w = 5$. Subplots (C) and (D) show the distribution for the same elasticity values using the xyz data subset with window size $w = 3$.

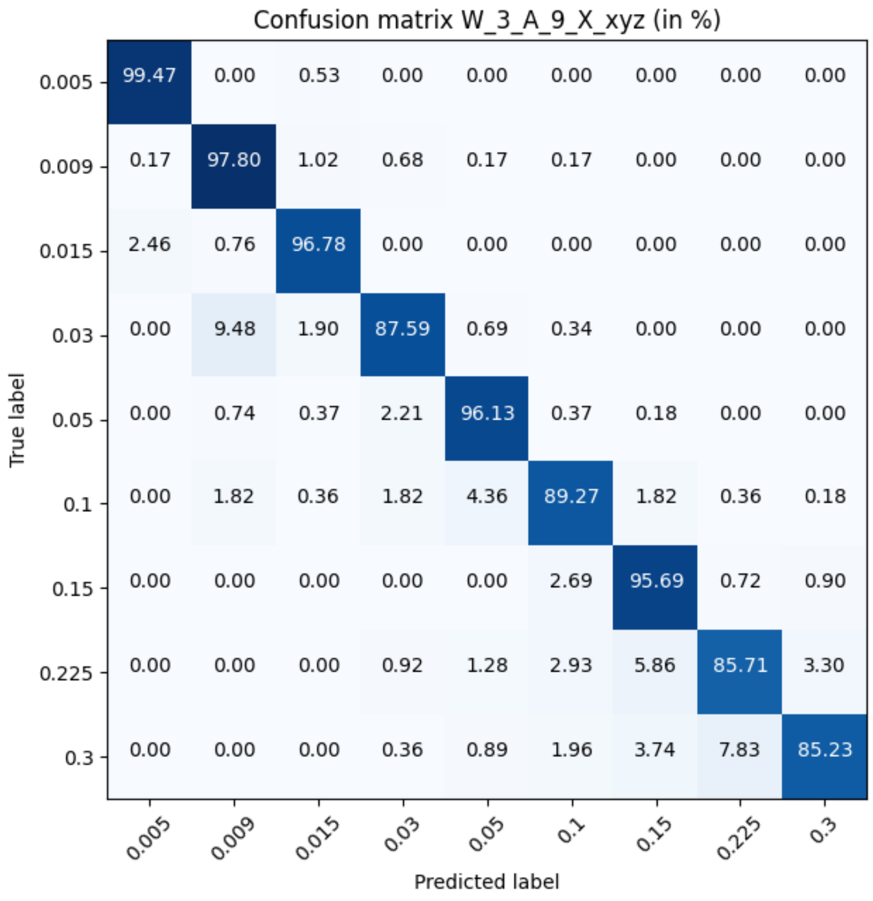


True label \ Predicted label	0.005	0.009	0.015	0.03	0.05	0.1	0.15	0.225	0.3
0.005	94.35	2.65	2.83	0.18	0.00	0.00	0.00	0.00	0.00
0.009	1.69	92.03	3.39	2.03	0.85	0.00	0.00	0.00	0.00
0.015	34.85	8.33	56.82	0.00	0.00	0.00	0.00	0.00	0.00
0.03	0.52	35.52	7.93	53.62	2.41	0.00	0.00	0.00	0.00
0.05	0.55	10.15	2.77	4.06	81.55	0.74	0.18	0.00	0.00
0.1	1.64	16.91	4.73	6.00	12.73	53.45	2.73	1.45	0.36
0.15	0.00	0.18	0.00	0.18	0.54	7.72	86.18	5.21	0.00
0.225	0.00	2.75	0.55	2.75	9.16	3.66	15.75	64.29	1.10
0.3	0.00	2.14	0.71	0.71	2.31	10.14	12.28	16.73	54.98

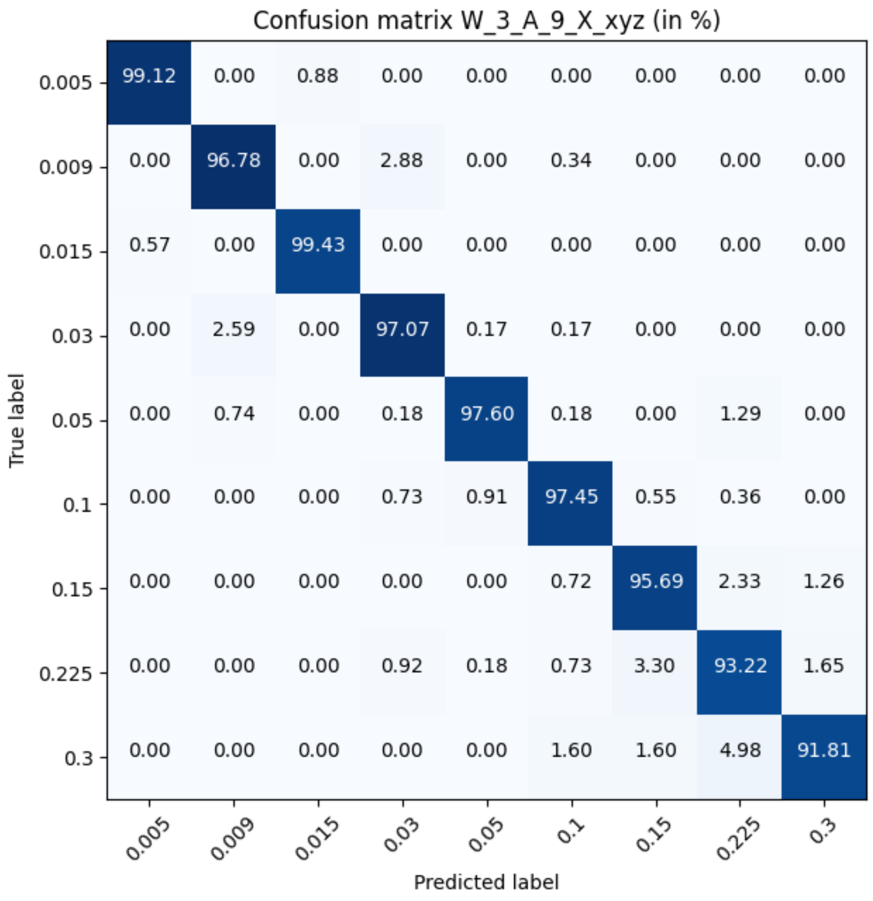


True label \ Predicted label	0.005	0.009	0.015	0.03	0.05	0.1	0.15	0.225	0.3
0.005	89.22	0.71	9.89	0.00	0.18	0.00	0.00	0.00	0.00
0.009	0.00	76.78	0.34	18.47	2.20	1.69	0.17	0.17	0.17
0.015	6.25	0.57	92.99	0.00	0.00	0.19	0.00	0.00	0.00
0.03	0.86	8.62	0.17	87.24	1.21	1.21	0.17	0.17	0.34
0.05	0.18	2.03	0.00	0.92	90.59	2.40	0.55	2.77	0.55
0.1	0.18	4.55	0.00	0.91	3.82	80.73	3.64	4.00	2.18
0.15	0.00	0.00	0.00	0.00	0.00	3.41	88.51	4.49	3.59
0.225	0.00	0.73	0.00	0.55	1.28	4.40	7.69	82.78	2.56
0.3	0.00	0.53	0.00	0.18	0.00	8.19	9.96	6.76	74.38

Figure 6.9: RBC classification confusion matrices for RegToClass (up) and Classification (down) xz neural networks.



True label \ Predicted label	0.005	0.009	0.015	0.03	0.05	0.1	0.15	0.225	0.3
0.005	99.47	0.00	0.53	0.00	0.00	0.00	0.00	0.00	0.00
0.009	0.17	97.80	1.02	0.68	0.17	0.17	0.00	0.00	0.00
0.015	2.46	0.76	96.78	0.00	0.00	0.00	0.00	0.00	0.00
0.03	0.00	9.48	1.90	87.59	0.69	0.34	0.00	0.00	0.00
0.05	0.00	0.74	0.37	2.21	96.13	0.37	0.18	0.00	0.00
0.1	0.00	1.82	0.36	1.82	4.36	89.27	1.82	0.36	0.18
0.15	0.00	0.00	0.00	0.00	0.00	2.69	95.69	0.72	0.90
0.225	0.00	0.00	0.00	0.92	1.28	2.93	5.86	85.71	3.30
0.3	0.00	0.00	0.00	0.36	0.89	1.96	3.74	7.83	85.23



True label \ Predicted label	0.005	0.009	0.015	0.03	0.05	0.1	0.15	0.225	0.3
0.005	99.12	0.00	0.88	0.00	0.00	0.00	0.00	0.00	0.00
0.009	0.00	96.78	0.00	2.88	0.00	0.34	0.00	0.00	0.00
0.015	0.57	0.00	99.43	0.00	0.00	0.00	0.00	0.00	0.00
0.03	0.00	2.59	0.00	97.07	0.17	0.17	0.00	0.00	0.00
0.05	0.00	0.74	0.00	0.18	97.60	0.18	0.00	1.29	0.00
0.1	0.00	0.00	0.00	0.73	0.91	97.45	0.55	0.36	0.00
0.15	0.00	0.00	0.00	0.00	0.00	0.72	95.69	2.33	1.26
0.225	0.00	0.00	0.00	0.92	0.18	0.73	3.30	93.22	1.65
0.3	0.00	0.00	0.00	0.00	0.00	1.60	1.60	4.98	91.81

Figure 6.10: Confusion matrices classification RBC for RegToClass (up) and Classification (dow) xyz neural networks.

Chapter 7

Determination of elasticity in obstacle canal

Prior research has explored the determination of RBC elasticity through simulation experiments [16], same as we do in Chapter 6. However, this chapter introduces a novel methodology for distinguishing between different levels of RBC elasticity corresponding to different diseases based on the analysis of RBCs in canal with obsticles. The approach involves examining the geometric characteristics of RBCs in video recordings or numerical data. Video classification is accomplished using a CNN.

The simulation model of blood flow offers novel opportunities for the analysis of blood flow and its properties through NNs. While real blood flow experiment videos recorded by cameras offer insights into blood cell dynamics, they are limited to two dimensions. Gaining information about the third dimension necessitates additional camera recordings or alternative viewing angles. With a precise simulation model, this research explores the accuracy of identifying cell properties from a 2D image of the simulation compared to complete model data. The focus here is not solely on replicating the real experiment but on investigating the capabilities of NNs in categorizing cells of varying elasticity when provided with complete information (exact particle positions in all three dimensions) versus only two-dimensional video data. The use of CNNs to identify the elasticity of moving blood cells constitutes the second attempt to address this issue. In prior work, statistical methods and multidimensional data analysis were employed [15]. Additionally, CNNs were applied to predict the trajectory of red blood cells in blood flow analysis [31].

CNNs, inspired by biological systems, are widely employed in deep learning for image and video classification tasks. The primary advantage of CNNs lies in their capacity to analyze image content while explicitly leveraging spatial structure through features such as local filters, convolution, and max pooling. CNN architectures have demonstrated their proficiency in learning interpretable image features [137]. Consequently, these architectures effectively shift the focus from manual feature design to the design of network connectivity structures.

7.1 Simulation Experiments for RBC Health Classification

The data source for classifying the health of RBCs was obtained through multiple simulation experiments using the open-source software ESPResSo [129].

All simulations were conducted under consistent channel and fluid flow parameters. The channel had a cuboid shape with dimensions of $104 \times 60 \times 40\mu$m, and the fluid was discretized into a three-dimensional grid with a spatial step of 1 μm. The elasticity of RBCs becomes most apparent when they come into contact with other objects. Therefore, we designed a simulated canal topology where RBCs flow through a space with obstacles. The simulated canal featured five cylinders acting as obstacles, restricting the area of blood flow and inducing RBC elasticity, as depicted in Figure 7.1. This canal design aimed to replicate a realistic laboratory environment.

The kinematic viscosity of the fluid was $1.3\times10^{-6}\,\mathrm{m^2/s}$, and the density was $1.025\times10^3\,\mathrm{kg/m^3}$. To initiate fluid flow, external forces were applied, with values chosen to achieve a maximum velocity of approximately $0.03\,\mathrm{m/s}$.

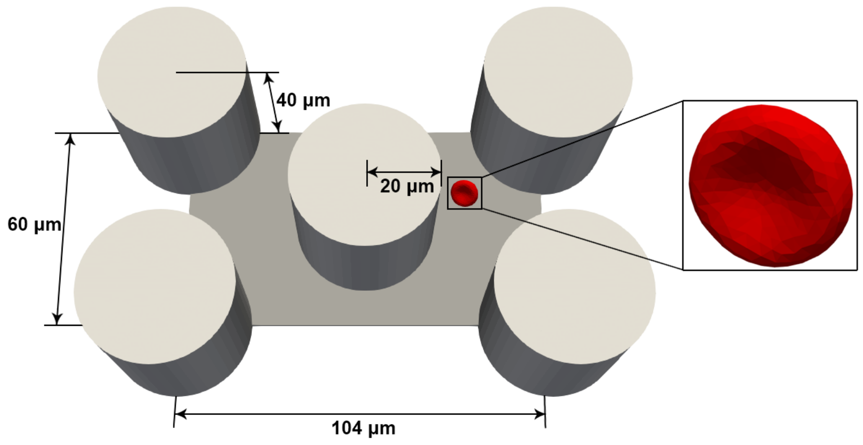


Figure 7.1: Scheme of the simulation microfluidic channel with cylindrical obstacles.

Cell-cell interactions were simulated using the *membrane_ collision* potential, while interactions between cells and the channel walls were modeled using the *soft_ sphere* potential. RBCs were represented by a surface network consisting of 374 nodes. The elastic properties of the cells were simulated using five types of elastic forces, each corresponding to a different elastic modulus in the same way as in the previous chapter.

Four levels of RBC elasticity were taken into the consideration. Healthy RBCs, with a stiffness coefficient (k_s) of 0.005, and the least elastic RBCs representing malaria-infected cells at stage 3 of the disease with a k_s value of 0.03. The remaining two levels of RBC elasticity were evenly distributed between healthy and malaria-infected RBCs, with k_s values of 0.0133 and 0.0216, respectively.

The size of the training dataset plays a crucial role in the training of machine learning (ML) models. When the dataset is small, the model may struggle to capture the intricate patterns and nuances present in the data, resulting in poor generalization performance on unseen data. In contrast, a larger dataset provides the model with more examples to learn from, leading to better generalization and reduced overfitting, where the model becomes too specialized to the training data. Therefore, having a sufficiently large training dataset is crucial for developing accurate and reliable ML models.

However, it is essential to consider the computational resources required to train models on large datasets. Larger datasets demand more computational power and longer training times. Thus, striking a balance between dataset size and available computational resources is necessary for successful model training.

Each simulation in our research involved 36 RBCs, with nine cells representing each level of elasticity. The simulated channel has a periodic structure, meaning that when an RBC exits the channel, it reappears at the beginning. We remove the initial and final passes of RBCs due to their incompleteness. As a result, each RBC completes approximately 20–21 passes through the simulated channel (Figure 7.2). Consequently, each part of the train/validation datasets contains approximately

$$4 \text{ types of RBC} \times 9 \text{ from each type} \times 21 \text{ passes} = 756 \text{ samples.}$$

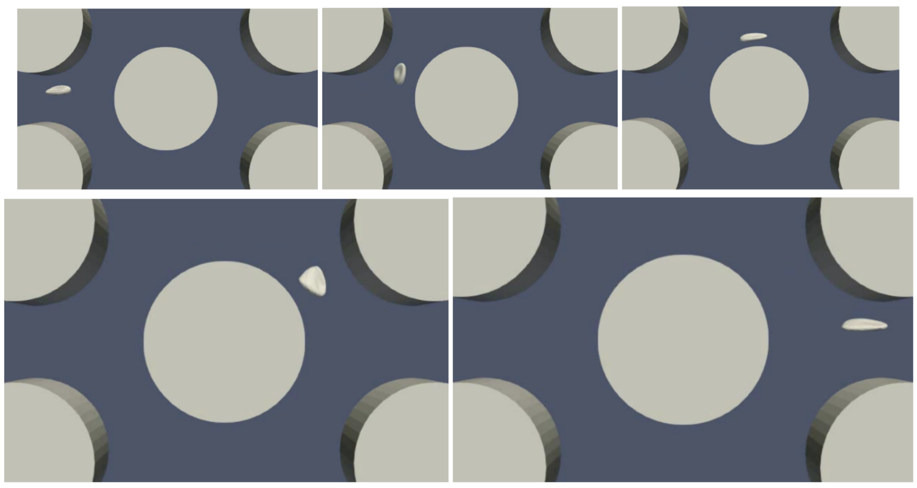


Figure 7.2: One pass of a RBC through the simulated channel.

Given the small number of training examples, we have two options to overcome this limitation.

The first option involves utilizing data augmentation techniques to augment our training set and increase its size. Data augmentation entails applying various transformations to the existing examples to generate new variations. In our case, we can perform vertical flips (to preserve the direction of blood flow) and rotations (while considering the preservation of blood flow direction). By applying these augmentations, we can create additional training examples and enhance the diversity of the dataset.

The second option is to leverage pretrained models, which do not necessitate training from scratch and thus require fewer training examples. Pretrained models are pre-trained on large-scale datasets and have acquired general features. We have chosen two pretrained models, EfficientNet v2 B0 and ResNet50, which have demonstrated successful results in various image and video tasks [81, 89, 36, 140]. These models only require a sufficient number of training examples to fine-tune the last few layers of the NN, making them well-suited to our small training dataset.

To prepare the dataset, we generated video recordings of individual RBCs from the simulation data. These videos are employed for training and validation purposes. As the simulations conducted in ESPResSo provide three-dimensional information about the flow of simulated RBCs, we projected this data onto a two-dimensional plane. We accomplished this by creating a 2D video that captures the width and length of the channel while disregarding differences in depth. This transformation enabled us to effectively analyze and classify RBC behavior in the videos using image and video classification techniques.

7.2 CNN-based Red Blood Cell Classification

Our network utilized video samples in the shape of $N \times T \times H \times W \times C$, where N represents the batch size, T is the number of frames in the video, H is the height, W is the width, and C is the number of channels, which is 3 (representing red, green, and blue). Subsequently, we rescaled the video into black and white format, reducing the number of channels from 3 to 1.

As the base for our models, we employed pretrained models, specifically EfficientNet v2 B0 and ResNet50. These models are originally designed for image classification. To adapt them for video classification, we utilized a time-distributed layer, enabling the model to classify video recordings. The pretrained model is not trainable. This was followed by 3D Average Pooling. Our network was completed with a sequence of Flatten, Dropout, and Dense layers (as illustrated in Figure 7.3), resulting in an output with the number of neurons equal to the number of classes, which, in our case, is four neurons. We used Dropout and regularization for the last dense layer to mitigate overfitting. The network's optimizer could be either Adam or SGD.

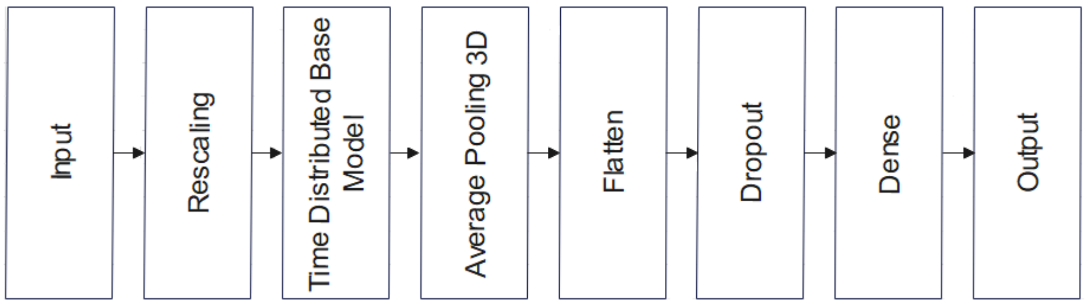


Figure 7.3: Diagram of the network architecture

Title: The base model can be either EfficientNet v2 B0 or ResNet50.

We optimized the hyperparameters of our network using the *Hyperband* class from the *keras_ tuner* module [91]. The optimized hyperparameters exhibit slight variations based on the network's optimizer, which are detailed in Table 7.1.

Table 7.1: Hyperparamters optimized for Adam and SGD.

Hyperparameter	Adam	SGD	Options
Dropout	yes	yes	0.05, 0.1, 0.2, 0.3
Regularizer	yes	yes	L1, L2, L1 + L2
Init learning rate	yes	yes	1e-3, 1e-4, 1e-5
Final learning rate	yes	no	1e-5, 1e-6, 1e-7
Decay steps	yes	no	1e+4, 1e+5, 1e+6
Momentum	no	yes	0.6, 0.8, 1.0

Neural Network Model	Optimizer	Validation Accuracy	
		4 Classes	2 Classes
EfficientNet v2 B0	Adam	55.48%	61.72%
	SGD	46.86%	56.44%
ResNet50	Adam	54.26%	55.12%
	SGD	51.45%	58.25%

Table 7.2: Validation accuracies for the models with 4 classes using optimized hyperparameters.

Hyperparameter	**Value**
Dropout	0.1
Regularizer	L1 + L2
Initial learning rate	1×10^{-5}
Final learning rate	1×10^{-5}
Decay steps	1×10^{4}

Table 7.3: The best set of hyperparameters from Adam optimizer.

The results of the hyperparameter optimization for each type of network and optimizer are presented in Table 7.1. The highest accuracy is attained with the EfficientNet v2 B0 model using the Adam optimizer, with the best hyperparameters outlined in Table 7.3. It is worth noting that the validation results for each model type are subpar, while the training accuracies are high, indicating two key findings. Firstly, the networks tend to overfit for each architecture and optimizer, despite the incorporation of regularization methods to mitigate this effect. Secondly, the presence of overfitting suggests that there is valuable information in the data that can be learned.

Upon analyzing the confusion matrix for the validation set (Figure 7.4), we identified that the primary challenge in classification lies in distinguishing between RBCs with reduced elasticity, while the differentiation between healthy and sick RBCs is more accurate. Consequently, we decided to train a binary classification NN. In this setup, the first class encompasses healthy RBCs ($k_s = 0.005$), and the second class includes all cells with reduced elasticity. We conducted hyperparameter optimization for two types of NNs with two different optimizer options.

Upon examining the confusion matrix depicted in Figure 7.5, we found that our initial hypothesis did not hold. Transitioning from a 4-class to a 2-class classification problem resulted in increased final accuracies, though not as significantly as the confusion matrix in Figure 7.4 may have implied. We also experimented with weighted classification using a combination of EfficientNet v2 B0 and the Adam optimizer, resulting in a marginal accuracy increase to 61.74%, marking only a 0.02% improvement.

Another approach was to separate healthy and sick RBCs after they were classified into 4 classes, where class 0 represented healthy RBCs, while classes 1, 2, and 3 represented sick RBCs. With this approach, we achieved a classification accuracy of 93.54%.

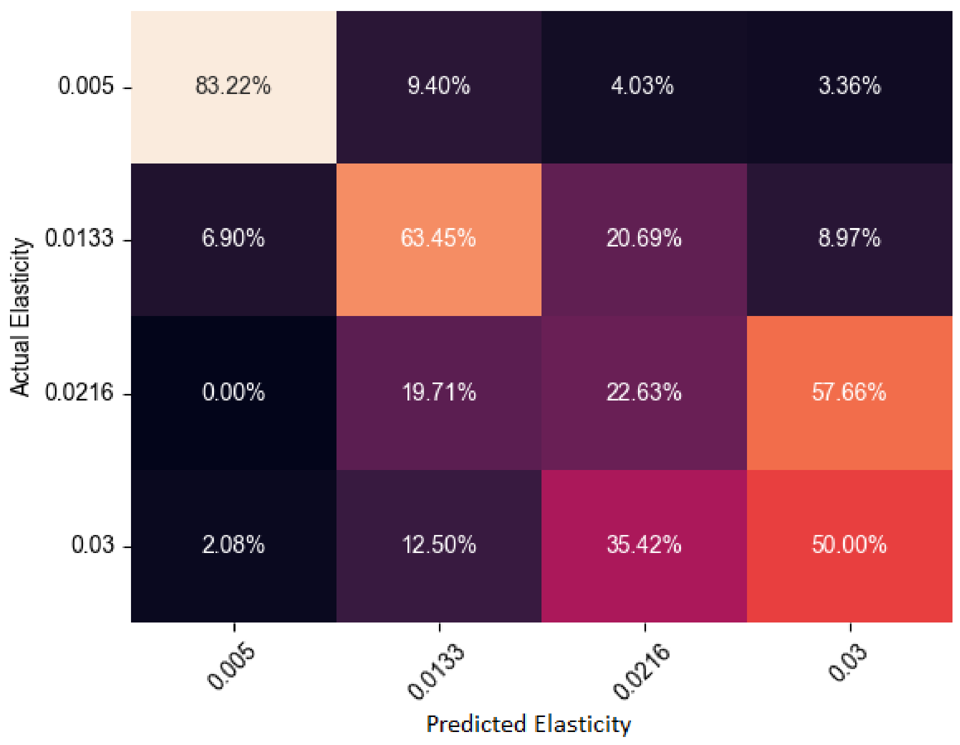


Figure 7.4: The confusion matrix of 4 classes classification.

Description: The confusion matrix represents the classification results of the validation set, which consists of samples belonging to 4 classes. The classification was performed using the EfficientNet v2 B0 model, which was optimized with the Adam optimization algorithm.

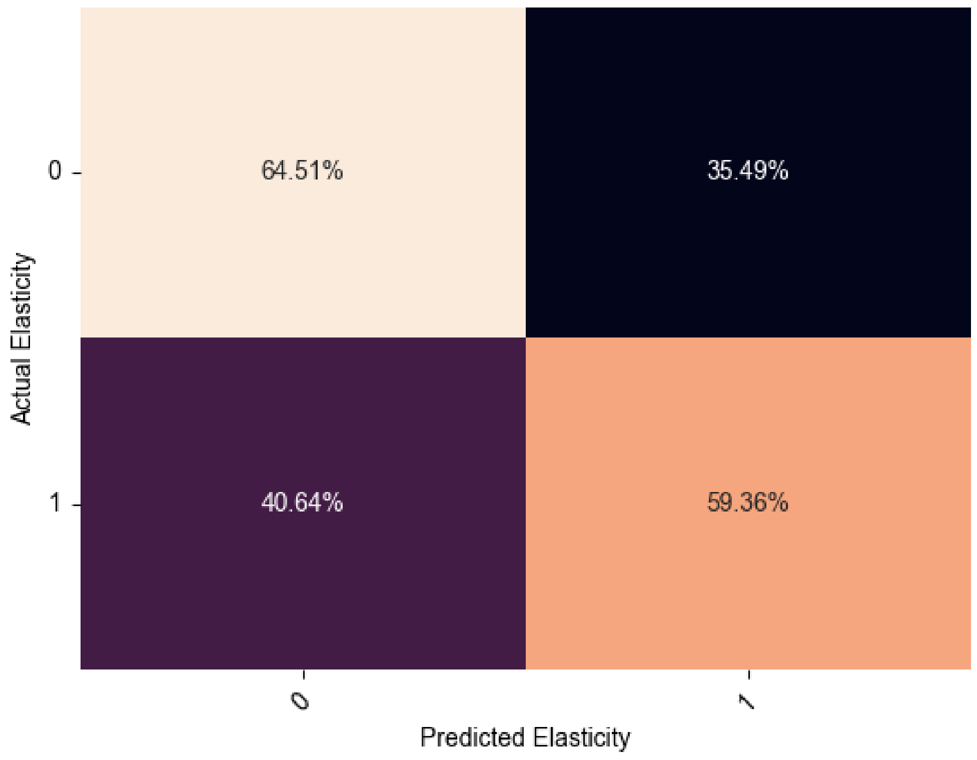


Figure 7.5: The confusion matrix of 2 classes classification.

Description: The confusion matrix represents the classification results of the validation set, which consists of samples belonging to 2 classes. The classification was performed using the EfficientNet v2 B0 model, which was optimized with the Adam optimization algorithm. The classes of healthy and sick are represented by the labels 0 and 1, respectively.

7.2.1 Adding Physical Information

In order to enhance the performance of the network, we added information about the underlying physics—specifically, about the velocity of the fluid flowing in the channel. Our hypothesis was that by adding the physical information, the NN would be able to learn to classify the RBCs better than without it.

First, using ESPReSso, we calculated the velocities of each point of the fluid (which is represented as a mesh; more information is available in [34]) in the empty channel—the channel with no RBCs in it. We obtained a 3D data of velocities, since the channel is a 3D object. Then, we took a layer $H \times W \times \lfloor D/2 \rfloor$ that corresponds to the velocities in the middle of the channel along the depth axis. This information can be used in two ways: we can either use it as it is, meaning 3D data with dimensions $W \times H \times 3$, where 3 represents the x, y, and z components of a velocity vector, or we can create a heatmap where the colors represent the velocity of the flow.

In both options, we added the physical information by creating a new branch of our NN that has the physical information about the flow as input. This information is passed through a Rescale layer, three 2D convolution layers with a number of filters 32, 16, 8, followed by a max pooling layer, a flattening layer, and a dense layer with 1024 hidden units. Then, it is concatenated with the penultimate output of the main branch of our NN, which is passed to the last dense layer.

Generally, it is not effective to add the same information to each training example for a NN. The reason for this is that NNs learn patterns and relationships in the data through the variations and differences between the examples. When all examples contain the same information, the network is unable to distinguish between them and may not learn the relevant patterns that are necessary for accurate predictions. However, there may be some cases where adding the same information to each training example can be helpful. For example, if the added information provides some contextual or background information that is relevant to all examples, it may help the network learn more effectively. In general, it is important to carefully consider the information that is added to each training example and how it may affect the network's ability to learn and generalize. We trained both versions of PINNs using the optimized hyperparameters described in Table 7.3, with the best performing model being EfficientNet v2 B0 with the Adam optimizer. The final results are presented in Table 7.4.

NN Model	**Accuracy**
4c_heatmap	87.83%
4c_values	91.48%
2c_heatmap	57.04%
2c_values	67.33%
2cw_heatmap	59.13%
2cw_values	69.78%

Table 7.4: The validation accuracies were assessed for six distinct classes of models with optimized hyperparameters.

Description: The names of these neural network (NN) models are composed of two parts. The first part indicates the number of classes used during training, and the second part specifies the type of physical information incorporated into the physics-informed neural network (PINN). The abbreviations "4c," "2c," and "2cw" represent models trained with four classes, two classes, and two weighted classes, respectively. Additionally, "Heatmap" and "Values" indicate the use of heatmap information and velocity vectors from the middle flow layer, respectively.

The obtained results reveal several key observations concerning the performance of different classification approaches. Firstly, when comparing the two-class classification (whether weighted or unweighted) to the conversion of four-class classification into a two-class classification, it is evident that the latter achieves significantly superior results. This implies that the additional information present in the four-class classification contributes to enhancing the overall model accuracy.

Moreover, the incorporation of physics-related information in the form of a heatmap seems to be less effective compared to using velocity vectors as input. This suggests that velocity vectors carry more meaningful and discriminative information for precise classification. Utilizing velocity vectors as input likely enables the model to capture dynamic patterns and better comprehend the motion characteristics of the analyzed data.

However, it is interesting to note that even with the inclusion of velocity vector information, there is no substantial improvement in the final accuracy of the network. Specifically, the NN relying solely on the provided input without any additional physical information achieves an accuracy of 93.56%. In contrast, the network using velocity vectors attains a slightly lower accuracy of 91.48%. This indicates that while the inclusion of velocity vectors may offer valuable information, it does not necessarily translate into a significant enhancement in classification performance.

7.2.2 Up-Scaling of the Healthy Examples

Our simulations encompass four types of RBCs, of which three types exhibit reduced elasticity and are categorized as sick. Although the ratio of healthy to sick RBCs is 1:3 in our dataset, this does not align with the observed ratio in reality [130]. To address this class imbalance, we employed data augmentation to augment the minority class and balance the class sizes. We applied horizontal flips and rotations to create two new training examples for each original example. This augmentation, in conjunction with the original healthy examples, increased the size of the healthy class threefold.

We trained the best-performing model from our previous experiments, a four-class classification NN without additional information, which was then converted into a two-class classification. We augmented both the training and validation datasets to maintain class distribution consistency between the datasets. After training and validating the model with the same class distribution, we cross-validated it on a dataset with a different class ratio.

Table 7.5 presents a comparison of three models: the original model (O_4x1) and the best-performing model from previous experiments, as well as two models trained on datasets with equal ratios of healthy and unhealthy RBCs. The first model employs unmodified class weights (A_3_1_1_1), resulting in equal weights for each class. The second model (AW_3_1_1_1), on the other hand, utilizes class weights proportional to the sizes of the classes. The results reveal that the second model outperforms the first model in terms of classification accuracy.

The findings indicate that the original dataset achieved the highest accuracy on the original dataset for all three models. The O_4x1 model attained an accuracy of 93.54%, while the A_3_1_1_1 and AW_3_1_1_1 models achieved lower accuracies of 88.88% and 81.55%, respectively.

When comparing the accuracies on the augmented dataset, it can be observed that the A_3_1_1_1 model performed slightly better, with an accuracy of 93.91%, compared to the 91.01% accuracy of the O_4x1 model. However, the AW_3_1_1_1 model achieved an accuracy of 92.21%, which was slightly lower than both the original dataset and the A_3_1_1_1 model.

In conclusion, augmenting the dataset through upsampling had a mixed impact on the model performance. While the A_3_1_1_1 model showed a slight improvement, the addition of class weights in the AW_3_1_1_1 model did not yield significant improvements and even resulted in a slightly decreased accuracy compared to the original dataset.

Validation	**O_4x1**	**A_3_1_1_1**	**AW_3_1_1_1**
original dataset	93.56%	88.88%	81.55%
augmented dataset	91.01%	93.91%	92.21%

Table 7.5: The validation accuracies of three models with identical architecture.

Description: The first model, referred to as O_4x1, was trained on the original dataset, which consisted of a roughly equal number of examples per class. The second model, denoted as A_3_1_1_1, was trained on an upsampled dataset, while the third model, labeled as AW_3_1_1_1, utilized the upsampled dataset with additional class weights proportional to their respective proportions.

7.2.3 Four-Class to Two-Class Classification

The obtained results unveil significant insights into the performance of different classification approaches. A notable observation lies in the comparison between two-class classification and the four-class classification converted to two-class classification. Two-class classification pertains to a classification task that distinguishes between two specific classes (e.g., healthy and sick), while four-class classification converted to two-class classification involves merging multiple classes into two broader categories.

The results demonstrate that the four-class classification converted to two-class classification yields significantly superior performance compared to two-class classification. This indicates that including additional classes in the training process imparts valuable information that enhances the overall model accuracy. Training the model on a dataset comprising multiple classes allows it to capture a broader spectrum of patterns, variances, and data characteristics. This expanded comprehension and increased model complexity contribute to improved classification accuracy when discerning between the two broader categories in the four-class classification converted to two-class classification scenario.

The enhanced performance achieved with the four-class classification converted to two-class classification underscores the importance of considering a more comprehensive representation of data during model training. By incorporating additional classes, the model can learn more nuanced and discriminative features, resulting in better differentiation between the target categories. This finding underscores the value of harnessing the full range of available classes and their associated information when constructing classification models.

The results highlight the importance of thoughtful classification task design and the selection of an appropriate class representation to achieve optimal performance. By utilizing the additional information provided by the four-class classification, the model gains a deeper understanding of the underlying data, leading to improved accuracy when distinguishing between broader categories. This underscores the significance of considering the relevance and inclusion of additional information in classification tasks.

Furthermore, the findings highlight the significance of choosing suitable input features in classification models. In this instance, the incorporation of velocity vectors as input features proves more effective compared to the use of a heatmap representation. This suggests that the choice of input features plays a main role in capturing relevant patterns and characteristics for accurate classification.

Moreover, the research underscores that the network's architecture and optimization techniques are main factors influencing the final performance. It is essential to consider the interplay between the model's architecture, training algorithms, and the specific classification task at hand. Fine-tuning these components and exploring alternative approaches may further enhance the accuracy and performance of the classification model.

The results stress the importance of carefully considering various factors, including classification task design, class representation, input feature selection, network architecture, and optimization techniques, to achieve optimal classification accuracy. Further analysis, experimentation, and method refinement are warranted to advance our understanding and improve the accuracy of classification tasks.

The code is available at [83].

7.3 Applying Random Forest Model

In case we utilize video pictures straightforwardly for classification, it is characteristic to select profound neural systems as a show. Be that as it may, within the occasion that we have accessible as information particular properties of the filtered RBC portraying its shape, speed and changes of these properties over time, it makes sense to moreover apply models that are appropriate for working with unthinkable information. The recreation from which we created the recordings can be additionally well utilized to create such information. Some time recently contributing in experiments with costly sensors, able to in this way confirm the classification exactness that can theoretically be anticipated with such a method. At the same time, we are going be able to compare the classification capacity of person models additionally assess which indicators have a noteworthy affect on classification precision. This will be vital when planning genuine tests.

7.3.1 Utilized Indicators - Featurization

We utilize RBC triangulation to calculate indicators from the recreation. The output of the simulation is for each cell the position of each of the 374 triangulation nodes, determined by three coordinates in three-dimensional space. Different characteristics can be calculated from this information, which we along these lines utilize as indicators for classification. In add up to, we made 41 indicators. We partitioned them into a few sets concurring to the evaluated complexity of getting them from a genuine explore. Based on this, we made 6 tests reviewed concurring to the number of indicators we utilize: Within the to begin with test we utilize as it were the easiest predictors to get, within the final test all of them.

- The 1st set comprises of as it were two indicators: the measurements of the rectangle in which the checked RBC is located, i.e. j. dimension of the RBC in the x-direction and within the y-axis heading. These values can be easily gotten from a (inactive) depiction of the RBC.
- Within the 2nd set, there are additionally velocities and changes in the speed of the RBC within the x and y directions. These information can be calculated from a few successive pictures.
- The 3rd set moreover contains measurements and velocities within the z-axis heading. These values can be gotten from pictures taken from a diverse point.
- Within the 4th set, we included the RBC axis length and the most extreme and least breadth of the RBC equator, which can possibly be decided from multi-angle pictures.

- The 5th set also contains predictors that can be calculated from the complete triangulation of the cell: cell surface, cell volume and diameters, standard deviations and skewness coefficients calculated from the lengths of all triangulation edges, angles subtended by every two triangles of the triangulation and spatial angles at all triangulation nodes. Cell triangulation is quite difficult to obtain from a real experiment, it would require the creation of a 3D image of the cell from the scanned flow.

- The 6th set too contains the midpoints, standard deviations and skewness coefficients of the deviations and supreme deviations of the characteristics added in the 5th set. The deviations are calculated from the RBC in a loose state, so for the calculation we got to have the essential triangulation of the RBC,which in our procedure corresponds to the cell in the first step of the simulation.

The mentioned predictors compare to the current state of the cell at one checked minute (in one step of the reenactment). It can be anticipated that the classification precision will progress in the case that we track the cell for a longer period of time. To evaluate this effect, we made an experiment for each $S = 1, 10, 20, 40, 80, 160, 320, 640$, and 1280 in which we observed each cell for S recorded steps[1].

7.3.2 Used Models

We used two of the most popular techniques: Random Forest and Gradient Boosted Decision Trees. For the implementation, we used the Python language and the methods *RandomForestClassifier* from the sklearn library and *XGBClassifier* from the xgboost library.

The goal in this part was not to maximize the accuracy to the highest possible level, therefore we did not optimize the hyperparameters of individual models and were satisfied with the default values.

For each of the six sets of predictors and each of the nine S values, we trained 2 models using both methods – one for classification into four classes (four degrees of cell elasticity), the other for classification into two classes (healthy/diseased cell). When evaluating the accuracy of the classification into two classes, in addition to the second model, we also used the first model, in which we combined the three degrees of elasticity into the result "sick cell" (same is in Chapter 7.2).

[1]Commentary: One entry of the cell through the channel compares to around 100 recorded steps.) Since the values of the person indicators alter amid the section of the cell through the channel, we have up to S values for each indicator rather than one value. From these values, we calculated the cruel and standard deviation and utilized them as predictors - in total, we have up to two indicators for each esteem shown within the statement over (with the exemption of the case of S = 1.

7.3.3 Data preparation

We sampled the simulation for each recorded step of the simulation, followed by at least $S\ \ 1$ additional recorded steps (so that we could calculate predictors for a given value of S). However, we removed the first 100 recorded steps of the simulation. The total number of generated samples is thus equal to $C\ \ (N\ \ 100\ \ S + 1)$, where C is the total number of cells in the simulation and N is the total number of simulation recorded steps.

It follows from the above that many generated samples are very similar to each other. The cell changes little between two consecutive recorded steps, moreover, for $S > 1$, two consecutive samples have a large part of the data from which we generate the predictors in common (the last $S\ \ 1$ recorded steps of the sample are identical to the first $S\ \ 1$ recorded steps of the next samples). This must be kept in mind when dividing the set into a possible training and validation or testing part - it is not appropriate to use data from one simulation in several parts and a new simulation must be used each time.

Due to the tree models used and the omission of hyperparameter optimization, we did not need the validation part, so we were satisfied with two simulations. We created training samples from the simulation with values of $C = 36$ and $N = 2356$, the total number of training samples is thus in the range from 81216 (for $S = 1$) to 35172 (for $S = 1280$). The second simulation, from which we generated test samples, has parameters $C = 36$ and $N = 2289$, so the number of samples is in a similar range (from 78804 for $S = 1$ to 32760 for $S = 1280$).

7.3.4 Results of Random Forest Approach

The classification accuracy in respect to the value of S, that is, the effect of the number of tracking recorded steps of one cell, was the first phenomena we focused on. From the graphs in figures 7.6, 7.7, it can be concluded that increasing S has a positive effect up to the level around the values of $S = 160$ or 320, from which the classification accuracy ceases to continuously improve. Thus, it seems pointless to track the cell significantly longer than during one passage through the channel (approx. 100 simulation recorded steps).

The graphs in figures 7.8, 7.9 show how the classification accuracy changes depending on which set of predictors we use. Especially when classifying into 4 classes, a significant improvement in accuracy can be observed when moving from the 4th set to the 5th set. On the contrary, the effect of the third dimension (the transition from the 2nd set to the 3rd set) does not seem to be very significant.

In all figures, the dashed line shows the accuracy we achieved in the classification

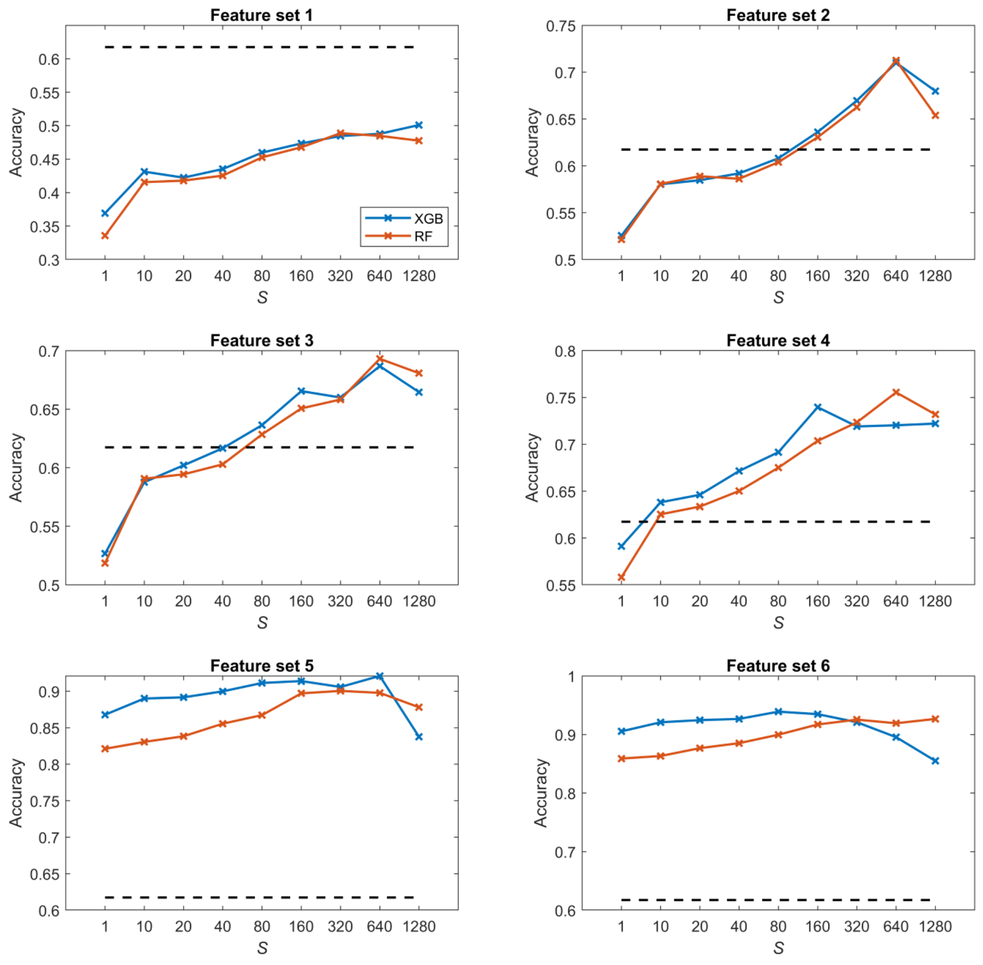


Figure 7.6: Classification to 4 classes.

using deep neural networks in 7.2[2]. In fig. 7.8, 7.9 in the middle graph it can be seen that we achieved almost identical accuracy with the 3rd set of predictors.

To see which predictors have the biggest impact on the classification, we took a closer look at the models for $S = 80$ (i.e., those where we observe approximately one cell passage through the channel) with the 6th set of predictors and the 4th set of predictors. For these models in figures 7.10 and 7.11 we present the significance of the predictors calculated by the permutation_importance method from the sklearn library. In the case of the 6th set and classification into four classes, the "edge length delta abs deviation" predictor appears to be the most significant in both models, that is, the predictor indicating the standard deviation of the absolute changes in the lengths of the edges of the triangulation against the relaxed state of the cell. When classifying

[2]We observed one passage through the channel in the video image of each cell, which corresponds approximately to a model with a value of S = 80

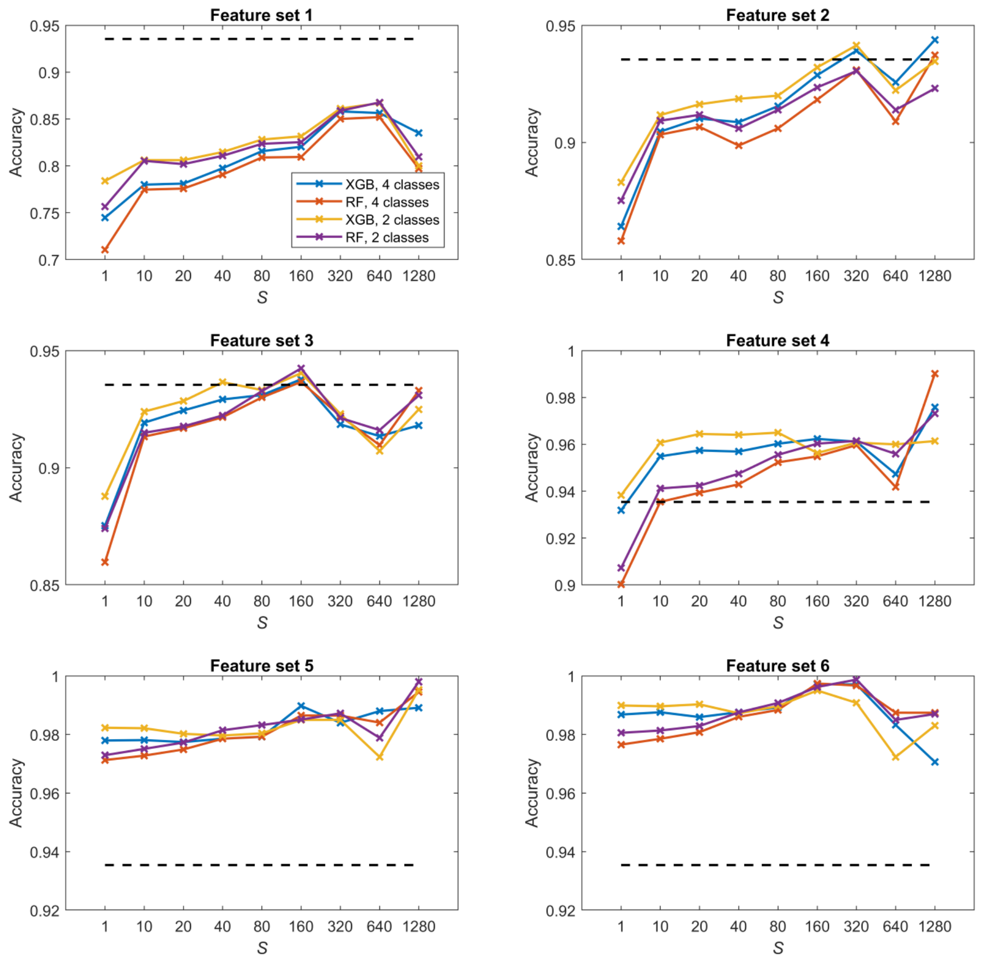


Figure 7.7: Classification to 2 classes.

into only two classes, "edge angle delta abs mean" and "edge length delta skewness" are significant for both models, that is, the average absolute change in the angles at the edges of the triangulation compared to the relaxed state and the coefficient of skewness of the changes in the lengths of the edges of the triangulation compared to the relaxed state. For the simpler 4th set of predictors, the most significant predictor for all models is the cell axis length.

7.3.5 Summary of Random Forest Approach

We investigated the effectiveness of a Random Forest model for classifying RBC elasticity using data generated from a simulation. The key findings are:

- Tracking duration impact: Increasing the number of recorded steps a cell is tracked improves classification accuracy up to a point (around 160-320 recorded

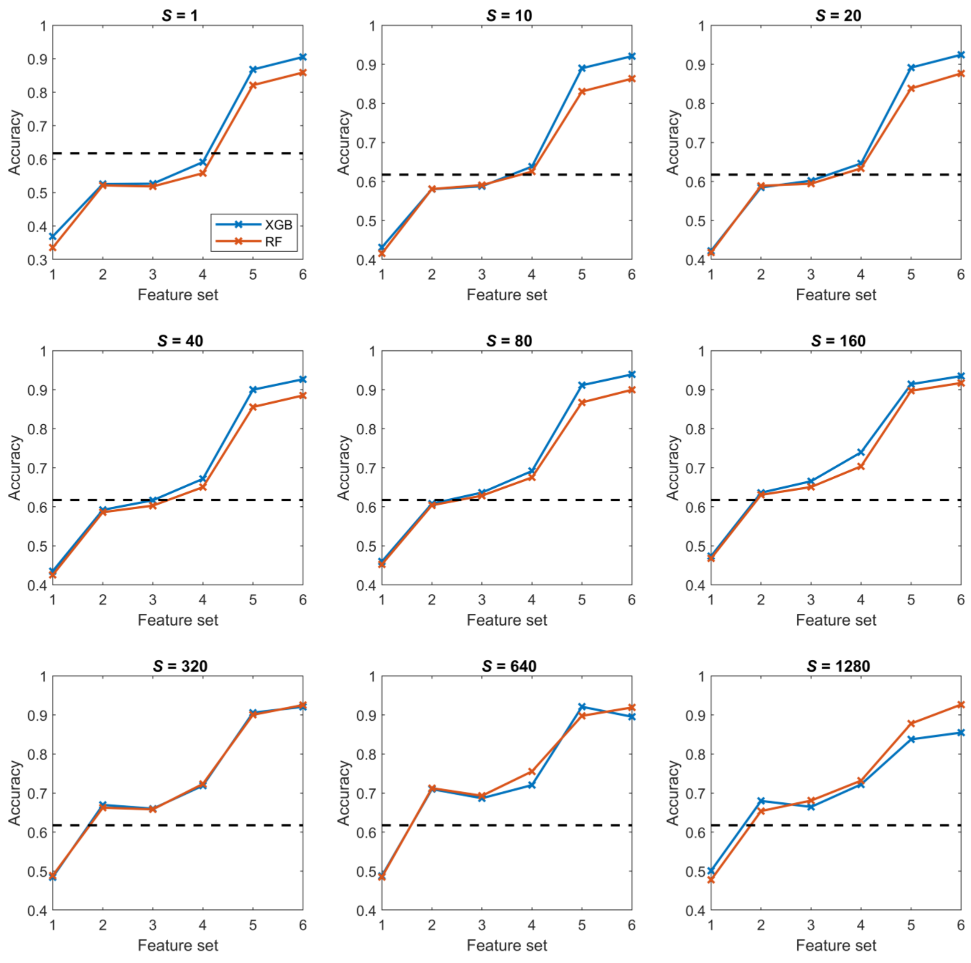


Figure 7.8: Classification to 4 classes.

steps), which corresponds roughly to a single passage through the simulated channel. Further tracking yielded minimal improvement.

- Feature set influence: The choice of features significantly affects classification accuracy. Using a more comprehensive feature set, including information from 3D triangulation, led to a substantial improvement, especially for classifying cells into four elasticity classes.

- Comparison with deep learning: The Random Forest model with a well-chosen feature set achieved comparable accuracy to a deep neural network approach (used previously) for classifying cells into four classes.

- Feature importance: The model provided insights into the most influential features for classification. For complex feature sets, standard deviation of changes in edge lengths and absolute changes in edge angles emerged as significant factors.

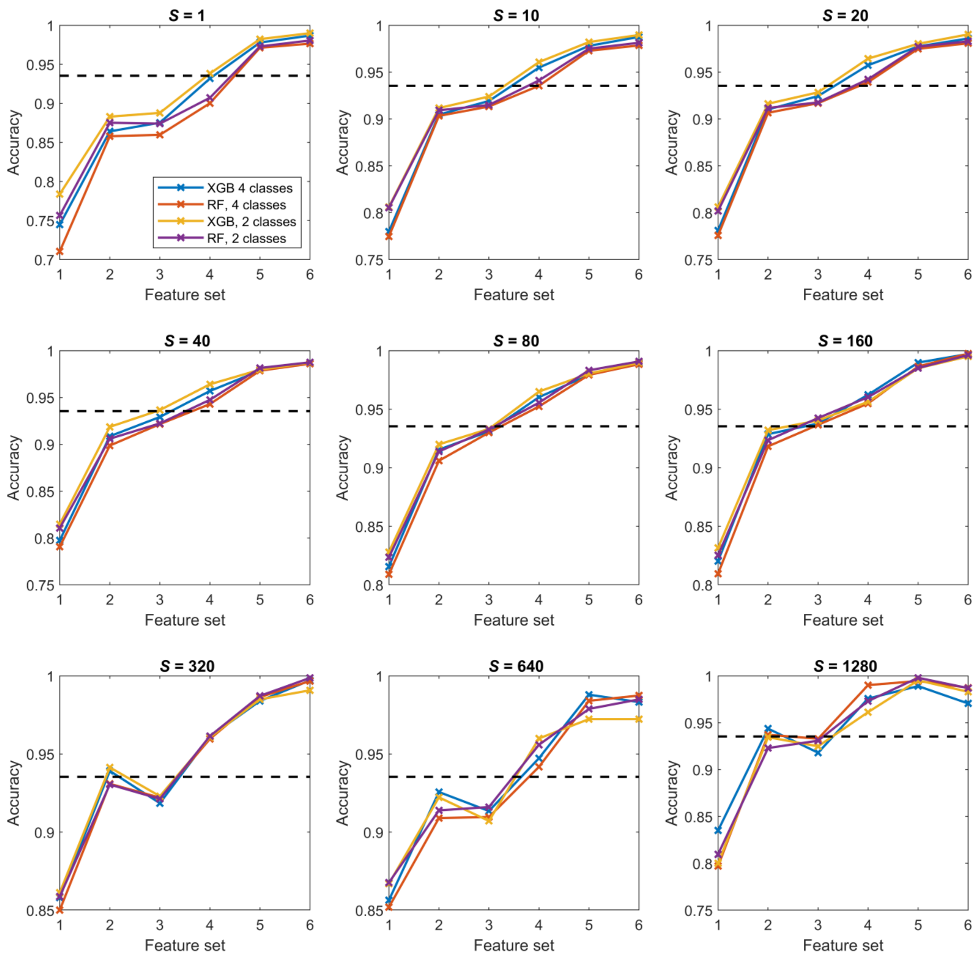


Figure 7.9: Classification to 2 classes.

For simpler sets, cell axis length played a key role.

These findings suggest that Random Forest models are a viable approach for RBC classification using simulation data. By carefully selecting features and considering the optimal tracking duration, accurate classification can be achieved. Additionally, the interpretability of Random Forest models allows for identifying the key characteristics impacting classification, which is valuable for understanding RBC behavior.

7.4 Statistical analysis of elasticity based on surface optimization

To fully utilize this data, we focus on numerical statistics to uncover deeper insights into canal dynamics. By analyzing numerical data, we aim to understand the complex

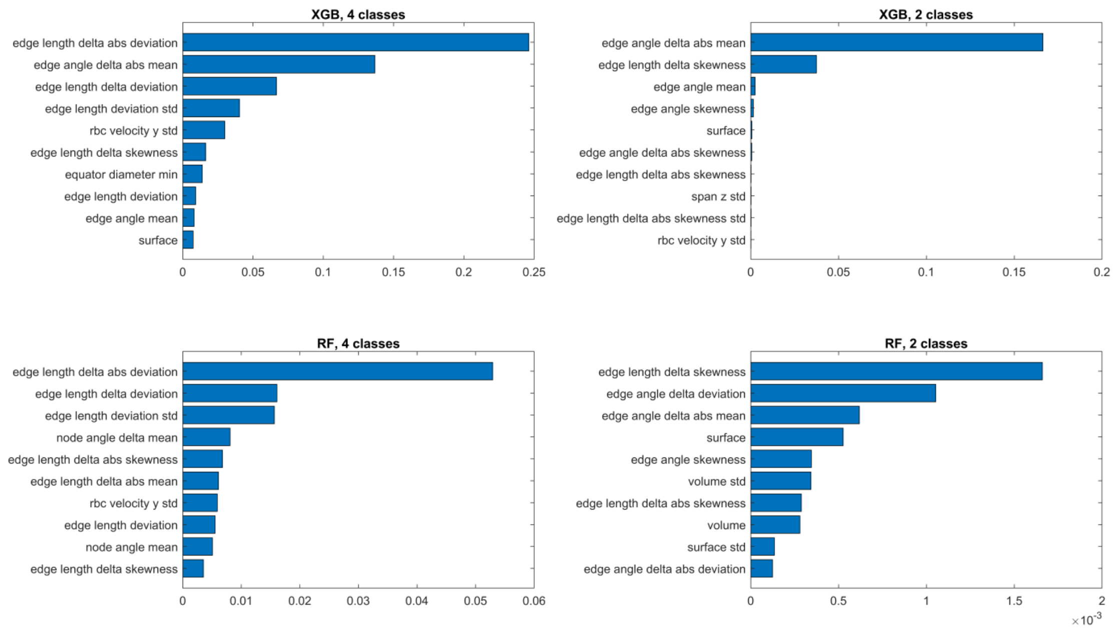


Figure 7.10: Feature importance for 6th set.

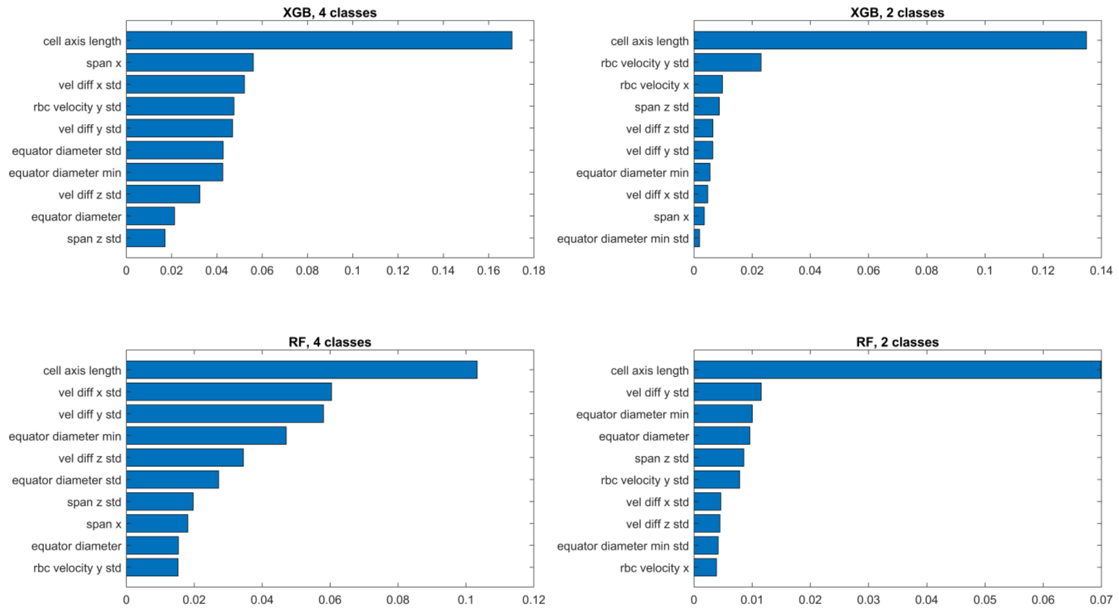


Figure 7.11: Feature importance for 4th set.

relationship between fluid flow and obstacles in canals. This approach allows us to identify patterns and trends within the data, enhancing our understanding of canal dynamics.

One of the key characteristics of RBC is their tendency to adopt a shape that maximizes surface area relative to volume, described as the SA/V ratio. While elasticity plays a significant role in various RBC properties such as membrane penetration in healthy cells and clumping in sickle-cell anemia, we were interested in how differences in elasticity affect the SA/V ratio. [88] and [101] suggest similar findings from real RBC studies.

We utilized simulation experiments to compare surface and volume values of individual RBCs. Real experiments have limitations in obtaining such data, making simulations particularly suitable. Study [88] suggests the possibility of measuring SA/V values in vitro, linking experimental and computational efforts. [94] also examines SA and volume values.

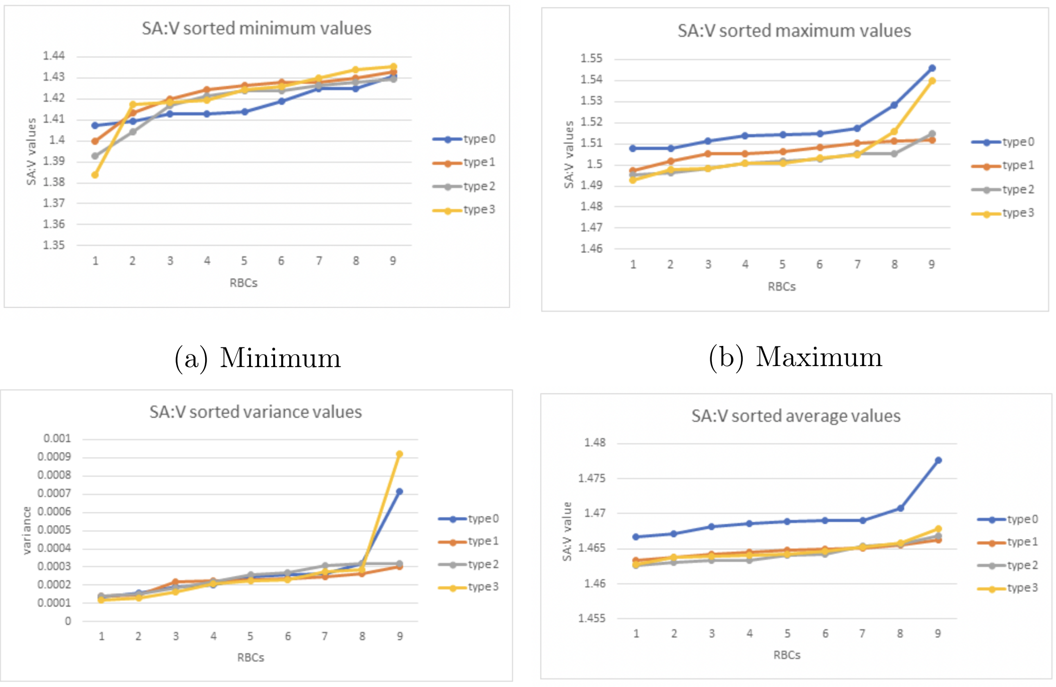
The core of RBCs typically adopts a shape that maximizes surface area in relation to volume, commonly described as the SA/V ratio. While elasticity plays a pivotal role in various RBC properties such as membrane penetration in healthy RBCs and clumping in sickle-cell anemia, we aimed to explore how differences in RBC elasticity affect the SA/V ratio. Previous studies [88] and [101] have indicated the feasibility of similar investigations with real RBCs.

Our comparison rely on surface and volume values of individual RBCs obtained from simulation experiments. We acknowledge the limitations in acquiring such data from real experiments and recognize the suitability of in-silico experiments for methodological analysis. Notably, [88] suggested the potential for measuring SA/V values in vitro, linking them to computational experiments. Additionally, [94] addressed the measurement of SA and V values. Deliberately opting for straightforward statistical methods, we seek to distinguish if this approach can yield meaningful results, contrasting with more complex machine learning methods.

Following the simulation experiment, each of the 36 cells with varying elasticity underwent analysis using a sequence of 2356 surface and volume values. For simplification, we selected four basic statistics for each RBC: minimum 7.12a, maximum 7.12b, variance 7.12c and average 7.12d of the SA/V values. The examination aimed to determine whether these statistics could distinguish between different levels of RBC elasticity.

The results showed promising differentiation particularly in the average and maximum SA/V values for normally elastic RBCs compared to others, with no significant difference observed for SA/V minimum and variance values. Verification of our assumption using the Kolmogorov-Smirnov test 7.13 supported these findings, showcasing distinctiveness in average and maximum SA/V values between cells of varying elasticity types.

By utilizing basic statistical measures such as maximum, minimum, average, and variance, along with the straightforward Kolmogorov-Smirnov test, we can observe significant differences in the average and maximum SA:V ratio values between healthy RBC type 0 and each group of RBCs with reduced elasticity. Conversely, in other



(a) Minimum (b) Maximum

(c) Variance (d) Average

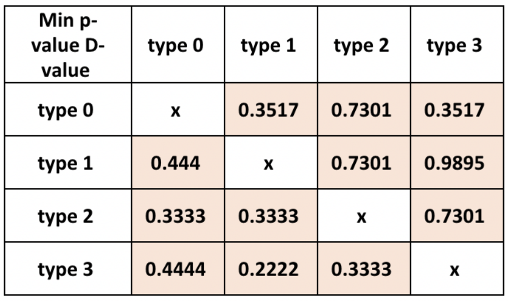
Figure 7.12: Graphs of values of SA:V for different statistics.

scenarios, such a distinction may not be achieved.

Acknowledging the limitations imposed by the small dataset resulting from the design of the simulation experiment, we recognize the simplicity of the proposed methodology as advantageous. This simplicity allows for verification of conclusions in larger in silico experiments and potentially in vitro as well.

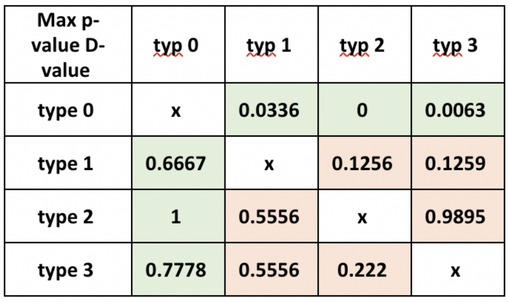
Moreover, we highlight the notable consistency between our findings and those mentioned in 7.2, despite the utilization of significantly different tools, data formats, and amounts, all originating from the same initial simulation.

Additionally, it's recognized that during the described experiment or simulation run, the volume of RBCs in the channel should remain constant, while their surface may deteriorate. Statistical verification revealed that the variance of simulated RBC surface values was approximately one hundred times greater than that of volume values. Consequently, limiting the examination solely to SA values would yield essentially the same results.



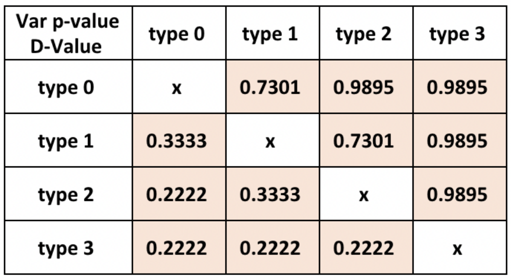
Min p-value D-value	type 0	type 1	type 2	type 3
type 0	x	0.3517	0.7301	0.3517
type 1	0.444	x	0.7301	0.9895
type 2	0.3333	0.3333	x	0.7301
type 3	0.4444	0.2222	0.3333	x

(a) Minimum



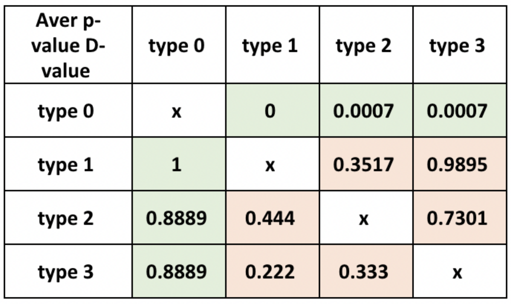
Max p-value D-value	typ 0	typ 1	typ 2	typ 3
type 0	x	0.0336	0	0.0063
type 1	0.6667	x	0.1256	0.1259
type 2	1	0.5556	x	0.9895
type 3	0.7778	0.5556	0.222	x

(b) Maximum



Var p-value D-Value	type 0	type 1	type 2	type 3
type 0	x	0.7301	0.9895	0.9895
type 1	0.3333	x	0.7301	0.9895
type 2	0.2222	0.3333	x	0.9895
type 3	0.2222	0.2222	0.2222	x

(c) Variance



Aver p-value D-value	type 0	type 1	type 2	type 3
type 0	x	0	0.0007	0.0007
type 1	1	x	0.3517	0.9895
type 2	0.8889	0.444	x	0.7301
type 3	0.8889	0.222	0.333	x

(d) Average

Figure 7.13: Graf of Kolmogorov-Smirnov test values of SA:V for different statistics.

Chapter 8

Summary

In this thesis, we looked into how computer simulations help us understand the elasticity of red blood cells, and we used neural networks to predict their elasticity. The simulations modeled how red blood cells move in blood flow, giving us important data to train neural networks. While simulations aren't exactly like real experiments, we adjusted the simulation results to match simple video recordings from experiments.

Chapter 6 takes a look into predicting red blood cell elasticity using high-performance CNN-LSTM, considering various input data dimensions. The study emphasizes the importance of using at least two devices for recording to capture blood flow in linearly independent planes. The CNN-LSTM architecture with 2D convolutional layers is identified as the most efficient, achieving acceptable accuracy, particularly with input data representing all three dimensions of the bounding boxes surrounding the RBC.

In Chapter 7, the focus shifts to the challenges of measuring RBC deformation rates and the potential of neural networks to compensate for the loss of accuracy and dimensionality when processing video recordings. The study explores RBC elasticity classification through CNN-based video analysis, employing architectures like ResNet and EfficientNet. The EfficientNet model emerges as the best classifier, and the importance of regularization techniques and cross-validation is highlighted.

Overall, the thesis contributes to the understanding of red blood cell dynamics and explores innovative diagnostic applications by combining computer simulations and CNN-guided video analysis. The findings underscore the potential of neural networks in categorizing red blood cell elasticity and emphasize the need for careful consideration of input data dimensions and model architectures for optimal classification performance. The research demonstrates the effectiveness of simulation-based training, ensuring robust neural network performance even with limited real-world video datasets.

In conclusion, our thesis highlights the potential of combining computer simulations, video analysis, and CNNs for RBC elasticity classification. Ongoing research and optimization are essential to enhance accuracy, ultimately contributing to improved healthcare outcomes.

Publications

Molčan, S.; Smiešková, M.; Bachratý, H.; Bachratá, K.: Computational Study of Methods for Determining the Elasticity of Red Blood Cells Using Machine Learning. Symmetry 2022, 14, 1732.

Molčan, S., Smiešková, M., Bachratý, H., Bachratá, K., & Novotný, P.: Classification of Red Blood Cells Using Time-Distributed Convolutional Neural Networks from Simulated Videos. Applied Sciences 2023, 13(13), 7967.

Samuel Molčan ... [et al.].: Models for monitoring progressive multiple sclerosis. Študentská vedecká konferencia FMFI UK, Bratislava 2021, Proceedings of the Student Science Conference 2021 : Zborník príspevkov.

Bibliography

[1] The annotated resnet-50. `https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758`. Accessed: 2023-06-16.

[2] Cell in fluid (cif), biomedical modeling & computation group. `https://cellinfluid.fri.uniza.sk/`. Accessed: 2022-08-01.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. `https://www.tensorflow.org/`, 2015. Software available from tensorflow.org.

[4] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11), 2018.

[5] Adoubi Vincent De Paul Adombi, Romain Chesnaux, and Marie-Amélie Boucher. Comparing numerical modelling, traditional machine learning and theory-guided machine learning in inverse modeling of groundwater dynamics: A first study case application. *Journal of Hydrology*, 615:128600, 2022.

[6] R Advani, S Sorenson, E Shinar, W Lande, E Rachmilewitz, and SL Schrier. Characterization and comparison of the red blood cell membrane damage in severe human alpha-and beta-thalassemia. 1992.

[7] Hadi Afsaneh and Rasool Mohammadi. Microfluidic platforms for the manipulation of cells and particles. *Talanta Open*, page 100092, 2022.

[8] Rupesh Agrawal, Thomas Smart, João Nobre-Cardoso, Christopher Richards, Rhythm Bhatnagar, Adnan Tufail, David Shima, Phil H. Jones, and Carlos Pavesio. Assessment of red blood cell deformability in type 2 diabetes mellitus and diabetic retinopathy by dual optical tweezers stretching technique. *Scientific reports*, 6(1):15873, 2016.

[9] Patrick Ahlrichs and Burkhard Dünweg. Lattice-boltzmann simulation of polymer-solvent systems. *International Journal of Modern Physics C*, 9(08):1429–1438, 1998.

[10] Shams Forruque Ahmed, Md Sakib Bin Alam, Maruf Hassan, Mahtabin Rodela Rozbu, Taoseef Ishtiak, Nazifa Rafa, M Mofijur, ABM Shawkat Ali, and Amir H Gandomi. Deep learning modelling techniques: current progress, applications, advantages, and challenges. *Artificial Intelligence Review*, pages 1–97, 2023.

[11] Md Zahangir Alom, Tarek M Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C Van Essen, Abdul AS Awwal, and Vijayan K Asari. A state-of-the-art survey on deep learning theory and architectures. *electronics*, 8(3):292, 2019.

[12] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

[13] Filippo Amato, Alberto López, Eladia María Peña-Méndez, Petr Vaňhara, Aleš Hampl, and Josef Havel. Artificial neural networks in medical diagnosis, 2013.

[14] Axel Arnold, Olaf Lenz, Stefan Kesselheim, Rudolf Weeber, Florian Fahrenberger, Dominic Roehm, Peter Košovan, and Christian Holm. Espresso 3.1: Molecular dynamics software for coarse-grained models. In *Meshfree methods for partial differential equations VI*, pages 1–23. Springer, 2013.

[15] K Bachratá, H Bachratỳ, and M Slavík. Statistics for comparison of simulations and experiments of flow of blood cells. In *EPJ Web of Conferences*, volume 143, page 02002. EDP Sciences, 2017.

[16] Katarína Bachratá, Katarína Buzáková, Michal Chovanec, Hynek Bachratỳ, Monika Smiešková, and Alžbeta Bohiniková. Classification of red blood cell rigidity from sequence data of blood flow simulations using neural networks. *Symmetry*, 13(6):938, 2021.

[17] Hynek Bachratý, Katarína Bachratá, Michal Chovanec, Iveta Jančigová, Monika Smiešková, and Kristína Kovalčíková. Applications of machine learning for simulations of red blood cells in microfluidic devices. *BMC bioinformatics*, 21(2):1–15, 2020.

[18] Hynek Bachratý, Katarína Bachratá, Michal Chovanec, František Kajánek, Monika Smiešková, and Martin Slavík. Simulation of blood flow in microfluidic devices for analysing of video from real experiments. In *International Conference on Bioinformatics and Biomedical Engineering*, pages 279–289. Springer, 2018.

[19] Peter Balogh and Prosenjit Bagchi. The cell-free layer in simulated microvascular networks. *Journal of Fluid Mechanics*, 864:768–806, 2019.

[20] Morteza Bayareh. Active cell capturing for organ-on-a-chip systems: a review. *Biomedical Engineering/Biomedizinische Technik*, 67(6):443–459, 2022.

[21] David Bento, Carla S Fernandes, Ana I Pereira, João Mário Miranda, and R Lima. Visualization and measurement of the cell-free layer (cfl) in a microchannel network. In *European congress on computational methods in applied sciences and engineering*, pages 930–936. Springer, 2017.

[22] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25:197–227, 2016.

[23] MM Brandao, A Fontes, ML Barjas-Castro, LC Barbosa, FF Costa, CL Cesar, and STO Saad. Optical tweezers for measuring red blood cell elasticity: application to the study of drug response in sickle cell disease. *European journal of haematology*, 70(4):207–211, 2003.

[24] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52:477–508, 2020.

[25] Martin Bušík, Iveta Jančigová, Renáta Tóthová, and Ivan Cimrák. Simulation study of rare cell trajectories and capture rate in periodic obstacle arrays. *Journal of Computational Science*, 17:370–376, 2016.

[26] Martin Bušík, Martin Slavík, and Ivan Cimrák. Dissipative coupling of fluid and immersed objects for modelling of cells in flow. *Computational and Mathematical Methods in Medicine*, 2018, 2018.

[27] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, pages 1–12, 2022.

[28] Joshua M Campbell, Joseph B Balhoff, Grant M Landwehr, Sharif M Rahman, Manibarathi Vaithiyanathan, and Adam T Melvin. Microfluidic and paper-based devices for disease detection and diagnostic research. *International journal of molecular sciences*, 19(9):2731, 2018.

[29] Juan R Cebral, Peter J Yim, Rainald Löhner, Orlando Soto, and Peter L Choyke. Blood flow modeling in carotid arteries with computational fluid dynamics and mr imaging. *Academic radiology*, 9(11):1286–1299, 2002.

[30] César Cheuque, Marvin Querales, Roberto León, Rodrigo Salas, and Romina Torres. An efficient multi-level convolutional neural network approach for white blood cells classification. *Diagnostics*, 12(2):248, 2022.

[31] Michal Chovanec, Hynek Bachratỳ, Katarína Jasenčáková, and Katarína Bachratá. Convolutional neural networks for red blood cell trajectory prediction in simulation of blood flow. In *International Work-Conference on Bioinformatics and Biomedical Engineering*, pages 284–296. Springer, 2019.

[32] T J Chung. *Computational fluid dynamics*. Cambridge university press, 2002.

[33] I Cimrák. Collision rates for rare cell capture in periodic obstacle arrays strongly depend on density of cell suspension. *Computer methods in biomechanics and biomedical engineering*, 19(14):1525–1530, 2016.

[34] Ivan Cimrak, Markus Gusenbauer, and I Jančigová. An espresso implementation of elastic objects immersed in a fluid. *Computer Physics Communications*, 185(3):900–907, 2014.

[35] Ivan Cimrak and Iveta Jancigova. *Computational Blood Cell Mechanics: Road Towards Models and Biomedical Applications*. CRC Press, 2018.

[36] Davide Alessandro Coccomini, Nicola Messina, Claudio Gennaro, and Fabrizio Falchi. Combining efficientnet and vision transformers for video deepfake detection. In *International conference on image analysis and processing*, pages 219–229. Springer, 2022.

[37] Louis F Diehl and Lloyd H Ketchum. Autoimmune disease and chronic lymphocytic leukemia: autoimmune hemolytic anemia, pure red cell aplasia, and autoimmune thrombocytopenia. In *Seminars in oncology*, volume 25, pages 80–97, 1998.

[38] Katherine S Elvira, Xavier Casadevall i Solvas, Robert CR Wootton, and Andrew J Demello. The past, present and potential for microfluidic reactor technology in chemical synthesis. *Nature chemistry*, 5(11):905–915, 2013.

[39] David Erickson. Towards numerical prototyping of labs-on-chip: modeling for integrated microfluidic devices. *Microfluidics and Nanofluidics*, 1(4):301–318, 2005.

[40] Magalie Faivre, Céline Renoux, Amel Bessaa, Lydie Da Costa, Philippe Joly, Alexandra Gauthier, and Philippe Connes. Mechanical signature of red blood cells flowing out of a microfluidic constriction is impacted by membrane elasticity, cell surface-to-volume ratio and diseases. *Frontiers in Physiology*, 11:576, 2020.

[41] Azin Shokraei Fard, David C Reutens, and Viktor Vegh. From cnns to gans for cross-modality medical image estimation. *Computers in Biology and Medicine*, 146:105556, 2022.

[42] Dmitry A Fedosov, Bruce Caswell, and George Em Karniadakis. A multiscale red blood cell model with accurate mechanics, rheology, and dynamics. *Biophysical journal*, 98(10):2215–2225, 2010.

[43] Maria Figurová, Dusan Pudis, Peter Gaso, and Ivan Cimrak. Pdms microfluidic structures for loc applications. In *2016 ELEKTRO*, pages 608–611. IEEE, 2016.

[44] Jörg Franke, Antti Hellsten, Heinke Schlünzen, and Bertrand Carissimo. *Best practice guideline for the CFD simulation of flows in the urban environment*. PhD thesis, COST European Cooperation in Science and Technology, 2007.

[45] Jonathan B Freund, Jonathan F MacArt, and Justin Sirignano. Dpm: A deep learning pde augmentation method (with application to large-eddy simulation). *arXiv preprint arXiv:1911.09145*, 2019.

[46] Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Guido L Chiarotti, Matteo Cococcioni, Ismaila Dabo, et al. Quantum espresso: a modular and open-source software project for quantum simulations of materials. *Journal of physics: Condensed matter*, 21(39):395502, 2009.

[47] Fang Hao, Xinyu Li, Ming Li, Yongfei Wu, and Wen Zheng. An accurate urine red blood cell detection method based on multi-focus video fusion and deep learning with application to diabetic nephropathy diagnosis. *Electronics*, 11(24):4176, 2022.

[48] H Hassoun and J Palek. Hereditary spherocytosis: a review of the clinical and molecular aspects of the disease. *Blood Reviews*, 10(3):129–147, 1996.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[50] L Hervé, DCA Kraemer, O Cioni, O Mandula, M Menneteau, S Morales, and C Allier. Alternation of inverse problem approach and deep learning for lens-free microscopy image reconstruction. *Scientific reports*, 10(1):1–12, 2020.

[51] Gene Hou, Jin Wang, and Anita Layton. Numerical methods for fluid-structure interaction—a review. *Communications in Computational Physics*, 12(2):337–377, 2012.

[52] Lotien Richard Huang, Edward C Cox, Robert H Austin, and James C Sturm. Continuous particle separation through deterministic lateral displacement. *Science*, 304(5673):987–990, 2004.

[53] Rick Huisjes, Anna Bogdanova, Wouter W Van Solinge, Raymond M Schiffelers, Lars Kaestner, and Richard Van Wijk. Squeezing for life–properties of red blood cell deformability. *Frontiers in physiology*, 9:656, 2018.

[54] Mehmet Tahir Huyut and Andrei Velichko. Diagnosis and prognosis of covid-19 disease using routine blood values and lognnet neural network. *Sensors*, 22(13):4820, 2022.

[55] Md Milon Islam, Md Zabirul Islam, Amanullah Asraf, Mabrook S Al-Rakhami, Weiping Ding, and Ali Hassan Sodhro. Diagnosis of covid-19 from x-rays using combined cnn-rnn architecture with transfer learning. *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, 2(4):100088, 2022.

[56] Iveta Jančigová, Kristína Kovalčíková, Alžbeta Bohiniková, and Ivan Cimrák. Spring-network model of red blood cell: From membrane mechanics to validation. *International Journal for Numerical Methods in Fluids*, 92(10):1368–1393, 2020.

[57] Iveta Jancigová, Kristína Kovalcíková, Rudolf Weeber, and Ivan Cimrák. Pyoif: Computational tool for modelling of multi-cell flows in complex geometries. *PLoS Computational Biology*, 16(10), 2020.

[58] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[59] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural

networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[60] Douglas B Kell and Etheresia Pretorius. No effects without causes: the iron dysregulation and dormant microbes hypothesis for chronic, inflammatory diseases. *Biological Reviews*, 93(3):1518–1557, 2018.

[61] Byungsoo Kim, Vinicius C Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. In *Computer graphics forum*, volume 38, pages 59–70. Wiley Online Library, 2019.

[62] Donghyuk Kim, Xiaojie Wu, Ashlyn T Young, and Christy L Haynes. Microfluidics-based in vivo mimetic systems for the study of cellular biology. *Accounts of chemical research*, 47(4):1165–1173, 2014.

[63] Tae-Young Kim and Sung-Bae Cho. Predicting residential energy consumption using cnn-lstm neural networks. *Energy*, 182:72–81, 2019.

[64] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[65] S Peter Klinken. Red blood cells. *The international journal of biochemistry & cell biology*, 34(12):1513–1518, 2002.

[66] Claudia Koch, Simon A Joosse, Svenja Schneegans, Okka JW Wilken, Melanie Janning, Desiree Loreth, Volkmar Müller, Katharina Prieske, Malgorzata Banys-Paluchowski, Ludwig J Horst, et al. Pre-analytical and analytical variables of label-independent enrichment and automated detection of circulating tumor cells in cancer patients. *Cancers*, 12(2):442, 2020.

[67] Christos Kotsalos, Jonas Latt, and Bastien Chopard. Bridging the computational gap between mesoscopic and continuum modeling of red blood cells for fully resolved blood flow. *Journal of Computational Physics*, 398:108905, 2019.

[68] Alexander Kovacs, Lukas Exl, Alexander Kornell, Johann Fischbacher, Markus Hovorka, Markus Gusenbauer, Leoni Breth, Harald Oezelt, Dirk Praetorius, Dieter Suess, et al. Magnetostatics and micromagnetics with physics informed neural networks. *Journal of Magnetism and Magnetic Materials*, 548:168951, 2022.

[69] Alexander Kovacs, Lukas Exl, Alexander Kornell, Johann Fischbacher, Markus Hovorka, Markus Gusenbauer, Leoni Breth, Harald Oezelt, Masao Yano, Noritsugu Sakuma, et al. Conditional physics informed neural networks. *Communications in Nonlinear Science and Numerical Simulation*, 104:106041, 2022.

[70] Kristina Kovalcikova, Ivan Cimrak, Katarina Bachrata, and Hynek Bachraty. Comparison of numerical and laboratory experiment examining deformation of red blood cell. In *International Work-Conference on Bioinformatics and Biomedical Engineering*, pages 75–86. Springer, 2019.

[71] Erik S Lamoureux, Emel Islamzada, Matthew VJ Wiens, Kerryn Matthews, Simon P Duffy, and Hongshen Ma. Assessing red blood cell deformability from microscopy images using deep learning. *Lab on a Chip*, 22(1):26–39, 2022.

[72] Can Li, Wei He, Nan Wang, Zhipeng Xi, Rongrong Deng, Xiyu Liu, Ran Kang, Lin Xie, and Xin Liu. Application of microfluidics in detection of circulating tumor cells. *Frontiers in Bioengineering and Biotechnology*, 10, 2022.

[73] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018.

[74] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 2021.

[75] Zhengjie Lin, Guanyi Luo, Weixiang Du, Tiantian Kong, Changkun Liu, and Zhou Liu. Recent advances in microfluidic platforms applied in cancer metastasis: Circulating tumor cells'(ctcs) isolation and tumor-on-a-chip. *Small*, 16(9):1903899, 2020.

[76] Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas B Schön. *Machine Learning: A First Course for Engineers and Scientists*. Cambridge University Press, 2022.

[77] Yi Liu, Sijing Li, and Yaling Liu. Machine learning-driven multiobjective optimization: An opportunity of microfluidic platforms applied in cancer research. *Cells*, 11(5):905, 2022.

[78] Ziqing Liu, Olivia Chen, J Wall, Michael Zheng, Yang Zhou, Li Wang, Haley Ruth Vaseghi, Li Qian, and Jiandong Liu. Systematic comparison of 2a peptides for cloning multi-genes in a polycistronic vector. *Scientific reports*, 7(1):1–9, 2017.

[79] KA Lukyanenko, IA Denisov, AS Yakimov, EN Esimbekova, KI Belousov, AS Bukatin, IV Kukhtevich, VV Sorokin, AA Evstrapov, and PI Belobrov. Analytical enzymatic reactions in microfluidic chips. *Applied biochemistry and microbiology*, 53:775–780, 2017.

[80] Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

[81] Gonçalo Marques, Deevyankar Agarwal, and Isabel De la Torre Díez. Automated medical diagnosis of covid-19 through efficientnet convolutional neural network. *Applied soft computing*, 96:106691, 2020.

[82] JP Mills, L Qie, M Dao, CT Lim, and S Suresh. Nonlinear elastic and viscoelastic deformation of the human red blood cell with optical tweezers. *Molecular & Cellular Biomechanics*, 1(3):169, 2004.

[83] Samuel Molčan. Source code for rbc classification model using video data. `https://github.com/molcan23/rbc_video_classification`.

[84] Samuel Molčan. Source code for rbc regression model using numerical data. `https://github.com/molcan23/RBC_NN`.

[85] Grégoire Montavon, Geneviève Orr, and Klaus-Robert Müller. *Neural networks: tricks of the trade*, volume 7700. springer, 2012.

[86] J Krishna Murthy, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations*, 2020.

[87] Sunitha Nagrath, Lecia V Sequist, Shyamala Maheswaran, Daphne W Bell, Daniel Irimia, Lindsey Ulkus, Matthew R Smith, Eunice L Kwak, Subba Digumarthy, Alona Muzikansky, et al. Isolation of rare circulating tumour cells in cancer patients by microchip technology. *Nature*, 450(7173):1235–1239, 2007.

[88] Arman Namvar, Adam Blanch, Matthew Dixon, Olivia Carmo, Boyin Liu, Snigdha Tiash, Oliver Looker, Dean Andrew, Li-Jin Chan, Wai-Hong Tham, et al. Surface area-to-volume ratio, not cellular viscoelasticity is the major determinant of red blood cell traversal through small channels. *Biophysical Journal*, 120(3):170a, 2021.

[89] Dillip Ranjan Nayak, Neelamadhab Padhy, Pradeep Kumar Mallick, Mikhail Zymbler, and Sachin Kumar. Brain tumor classification using dense efficient-net. *Axioms*, 11(1):34, 2022.

[90] Johan Nilsson, Mikael Evander, Björn Hammarström, and Thomas Laurell. Review of cell and particle trapping in microfluidic systems. *Analytica chimica acta*, 649(2):141–157, 2009.

[91] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Kerastuner. `https://github.com/keras-team/keras-tuner`, 2019.

[92] Milan Ondrašovič and Peter Tarábek. Siamese visual object tracking: A survey. *IEEE Access*, 9:110149–110172, 2021.

[93] Kacper Ostalowski and Jifu Tan. Direct simulation of blood flow with heterogeneous cell suspensions in a patient-specific capillary network. *Physics of Fluids*, 34(4), 2022.

[94] HyunJoo Park, SangYun Lee, Misuk Ji, Kyoohyun Kim, YongHak Son, Seongsoo Jang, and YongKeun Park. Measuring cell surface area and deformability of individual human red blood cells over blood storage using quantitative phase imaging. *Scientific reports*, 6(1):34257, 2016.

[95] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[96] Roberto Pol, Francisco Céspedes, David Gabriel, and Mireia Baeza. Microfluidic lab-on-a-chip platforms for environmental monitoring. *TrAC Trends in Analytical Chemistry*, 95:62–68, 2017.

[97] Yanjun Qi. Random forest for bioinformatics. *Ensemble machine learning: Methods and applications*, pages 307–323, 2012.

[98] Ning Qin, Pei Zhao, Emmanuel A Ho, Gongming Xin, and Carolyn L Ren. Microfluidic technology for antibacterial resistance study and antibiotic susceptibility testing: Review and perspective. *ACS sensors*, 6(1):3–21, 2020.

[99] John A Quinn, Rose Nakasi, Pius KB Mugagga, Patrick Byanyima, William Lubega, and Alfred Andama. Deep convolutional neural networks for microscopy-based point of care diagnostics. In *Machine learning for healthcare conference*, pages 271–281. PMLR, 2016.

[100] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[101] Céline Renoux, Magalie Faivre, Amel Bessaa, Lydie Da Costa, Philippe Joly, Alexandra Gauthier, and Philippe Connes. Impact of surface-area-to-volume ratio, internal viscosity and membrane viscoelasticity on red blood cell deformability measured in isotonic condition. *Scientific reports*, 9(1):6771, 2019.

[102] Céline Renoux, Magalie Faivre, Amel Bessaa, Lydie Da Costa, Philippe Joly, Alexandra Gauthier, and Philippe Connes. Impact of surface-area-to-volume ratio, internal viscosity and membrane viscoelasticity on red blood cell deformability measured in isotonic condition. *Scientific reports*, 9(1):6771, 2019.

[103] Gianluca Rigatelli, Marco Zuin, Sarthak Agarwal, Vivian Nguyen, Cardy Nguyen, Sanyaa Agarwal, and Thach Nguyen. Applications of computational fluid dynamics in cardiovascular disease. *TTU J Biomed Sci*, 2022.

[104] Jason Riordon, Dušan Sovilj, Scott Sanner, David Sinton, and Edmond WK Young. Deep learning with microfluidics for biotechnology. *Trends in biotechnology*, 37(3):310–324, 2019.

[105] Valeria Rizzuto, Arianna Mencattini, Begoña Álvarez-González, Davide Di Giuseppe, Eugenio Martinelli, David Beneitez-Pastor, Maria del Mar Mañú-Pereira, Maria José Lopez-Martinez, and Josep Samitier. Combining microfluidics with machine learning algorithms for rbc classification in rare hereditary hemolytic anemia. *Scientific Reports*, 11(1):1–12, 2021.

[106] Valeria Rizzuto, Arianna Mencattini, Begoña Álvarez-González, Davide Di Giuseppe, Eugenio Martinelli, David Beneitez-Pastor, Maria del Mar Mañú-Pereira, Maria José Lopez-Martinez, and Josep Samitier. Combining microfluidics with machine learning algorithms for rbc classification in rare hereditary hemolytic anemia. *Scientific Reports*, 11(1):13553, 2021.

[107] J Paul Robinson. Flow cytometry: past and future. *BioTechniques*, 72(4):159–169, 2022.

[108] Saurabhi Samant, Jules Joel Bakhos, Wei Wu, Shijia Zhao, Ghassan S Kassab, Behram Khan, Anastasios Panagopoulos, Janaki Makadia, Usama M Oguz, Akshat Banga, et al. Artificial intelligence, computational simulations, and extended reality in cardiovascular interventions. *Cardiovascular Interventions*, 16(20):2479–2497, 2023.

[109] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[110] Alessia Sarica, Antonio Cerasa, and Aldo Quattrone. Random forest algorithm for the classification of neuroimaging data in alzheimer's disease: a systematic review. *Frontiers in aging neuroscience*, 9:329, 2017.

[111] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[112] Matthias Scholz, Martin Fraunholz, and Joachim Selbig. Nonlinear principal component analysis: neural network models and applications. In *Principal manifolds for data visualization and dimension reduction*, pages 44–67. Springer, 2008.

[113] Erica L Schwarz, Luca Pegolotti, Martin R Pfaller, and Alison L Marsden. Beyond cfd: Emerging methodologies for predictive simulation in cardiovascular health and disease. *Biophysics Reviews*, 4(1), 2023.

[114] Christoph Schwarzmeier and Ulrich Rüde. Comparison of refilling schemes in the free-surface lattice boltzmann method. *AIP Advances*, 12(11), 2022.

[115] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[116] Monika Smiešková and Katarína Bachratá. Validation of bulk properties of red blood cells in simulations. In *2019 International Conference on Information and Digital Technologies (IDT)*, pages 417–423. IEEE, 2019.

[117] Subra Suresh. Mechanical response of human red blood cells in health and disease: Some structure-property-function relationships. *Journal of materials research*, 21(8):1871–1877, 2006.

[118] Subra Suresh, J Spatz, John P Mills, Alexandre Micoulet, Ming Dao, CT Lim, M Beil, and Thomas Seufferlein. Connections between single-cell biomechanics and human disease states: gastrointestinal cancer and malaria. *Acta biomaterialia*, 1(1):15–30, 2005.

[119] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[120] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021.

[121] Giovanna Tomaiuolo. Biomechanical properties of red blood cells in health and disease towards microfluidics. *Biomicrofluidics*, 8(5), 2014.

[122] Chia-Hung Dylan Tsai, Junichi Tanaka, Makoto Kaneko, Mitsuhiro Horade, Hiroaki Ito, Tatsunori Taniguchi, Tomohito Ohtani, and Yasushi Sakata. An on-chip rbc deformability checker significantly improves velocity-deformation correlation. *Micromachines*, 7(10):176, 2016.

[123] Aliaksei S Vasilevich, Aurélie Carlier, Jan de Boer, and Shantanu Singh. How not to drown in data: a guide for biomaterial engineers. *Trends in biotechnology*, 35(8):743–755, 2017.

[124] Ricardo Vinuesa and Steven L Brunton. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366, 2022.

[125] Chi-Hwa Wang and Aleksander S Popel. Effect of red blood cell shape on oxygen transport in capillaries. *Mathematical biosciences*, 116(1):89–110, 1993.

[126] Junchao Wang, Victor GJ Rodgers, Philip Brisk, and William H Grover. Instantaneous simulation of fluids and particles in complex microfluidic devices. *PloS one*, 12(12):e0189429, 2017.

[127] Sun-Chong Wang and Sun-Chong Wang. Artificial neural network. *Interdisciplinary computing in java programming*, pages 81–100, 2003.

[128] Serena AJ Watkin, Rachel Z Bennie, Jenna M Gilkes, Volker M Nock, F Grant Pearce, and Renwick CJ Dobson. On the utility of microfluidic systems to study protein interactions: advantages, challenges, and applications. *European Biophysics Journal*, 52(4-5):459–471, 2023.

[129] Florian Weik, Rudolf Weeber, Kai Szuttor, Konrad Breitsprecher, Joost de Graaf, Michael Kuron, Jonas Landsgesell, Henri Menke, David Sean, and Christian Holm. Espresso 4.0–an extensible software package for simulating soft matter systems. *The European Physical Journal Special Topics*, 227:1789–1816, 2019.

[130] Ibert C Wells and Harvey A Itano. Ratio of sickle-cell anemia hemoglobin to normal hemoglobin in sicklemics. *Journal of Biological Chemistry*, 188(1):65–74, 1951.

[131] John F Wendt. *Computational fluid dynamics: an introduction.* Springer Science & Business Media, 2008.

[132] Rui Yan, Jiaqiang Liao, Jie Yang, Wei Sun, Mingyue Nong, and Feipeng Li. Multi-hour and multi-site air quality index forecasting in beijing using cnn, lstm, cnn-lstm, and spatiotemporal clustering. *Expert Systems with Applications*, 169:114513, 2021.

[133] Yihuan Yan and Jiyuan Tu. Computational fluid dynamics. In *Bioaerosol Characterisation, Transportation and Transmission: Fundamental, Modelling and Application*, pages 65–83. Springer, 2023.

[134] DongHun Yeo and DongHun Yeo. *A summary of industrial verification, validation, and uncertainty quantification procedures in computational fluid dynamics.* US Department of Commerce, National Institute of Standards and Technology, 2020.

[135] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[136] Mohd Hafiz Zawawi, A Saleha, A Salwa, NH Hassan, Nazirul Mubin Zahari, Mohd Zakwan Ramli, and Zakaria Che Muda. A review: Fundamentals of computational fluid dynamics (cfd). In *AIP conference proceedings*, volume 2030. AIP Publishing, 2018.

[137] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

[138] Xiao Zhang, Christina Caruso, Wilbur A Lam, and Michael D Graham. Flow-induced segregation and dynamics of red blood cells in sickle cell disease. *Physical review fluids*, 5(5):053101, 2020.

[139] Yonghao Zhang and Pinar Ozdemir. Microfluidic dna amplification—a review. *Analytica chimica acta*, 638(2):115–125, 2009.

[140] Linchao Zhu and Yi Yang. Compound memory networks for few-shot video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 751–766, 2018.

Appendix - LSTM Principles

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \quad \text{Input gate} \tag{8.1}$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \quad \text{Forget gate} \tag{8.2}$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \quad \text{Output gate} \tag{8.3}$$

where:

i_t is the input gate

f_t is the forget gate

o_t is the output gate

h_{t-1} is the hidden state from the previous time step

x_t is the input at the current time step

σ represents the sigmoid activation function

w_i, w_f, w_o are the weight matrices for the input, forget, and output gates

b_i, b_f, b_o are the bias terms for the input, forget, and output gates

$$c'_t = \tanh(w_c[h_{t-1}, x_t] + b_c) \qquad \text{Input gate} \tag{8.4}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t \qquad \text{Memory cell update} \tag{8.5}$$

$$h_t = o_t \cdot \tanh(c_t) \qquad \text{Hidden state update} \tag{8.6}$$

where:

c'_t is the input gate

c_t is the memory cell

h_t is the hidden state

w_c is the weight matrix for the memory cell

b_c is the bias term for the memory cell

tanh represents the hyperbolic tangent activation function

To calculate the memory vector for the current timestamp (c_t), a candidate is computed. The cell state, at any given timestamp, decides what to forget from the previous state (i.e., $f_t * c_{t-1}$) and what to consider from the current timestamp (i.e., $i_t * c'_t$). The symbol * represents element-wise multiplication of vectors.

Finally, the cell state is filtered and passed through an activation function that predicts which portion should appear as the output of the current LSTM unit at timestamp t. This output (h_t) from the current LSTM block can be further processed through a softmax layer to obtain the predicted output (y_t) from the block.