symmetry

MDPI

Article

Computational Study of Methods for Determining the Elasticity of Red Blood Cells Using Machine Learning

Samuel Molčan *, Monika Smiešková *, Hynek Bachratý and Katarína Bachratá

Department of Software Technology, Faculty of Management Science and Informatics, University of Žilina, 010 26 Žilina, Slovakia

* Correspondence: samuel.molcan@fri.uniza.sk (S.M.); monika.smieskova@fri.uniza.sk (M.S.)

Abstract: RBC (Red Blood Cell) membrane is a highly elastic structure, and proper modelling of this elasticity is essential for biomedical applications that involve computational experiments with blood flow. In this work, we present a new method for estimating one of the key parameters of red blood cell elasticity, which uses a neural network trained on the simulation outputs. We test classic LSTM (Long-Short Term Memory) architecture for the time series regression task, and we also experiment with novel CNN-LSTM (Convolutional Neural Network) architecture. We paid special attention to investigating the impact of the way the three-dimensional training data are reduced to their two-dimensional projections. Such a comparison is possible thanks to working with simulation outputs that are equivalently defined for all dimensions and their combinations. The obtained results can be used as recommendations for an appropriate way to record real experiments for which the reduced dimension of the acquired data is essential.

Keywords: regression neural networks; red blood cells elasticity; elastic object in flow; simulations of blood flow; sequential data

Citation: 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. https://doi.org/10.3390/sym14081732

Academic Editor: Rahmat Ellahi

Received: 14 July 2022
Accepted: 13 August 2022
Published: 19 August 2022

1. Introduction

Microfluidics is a field that has seen a rapid growth in recent decades. It has a wide variety of biological and biomedical applications, for example in manipulation of cells and particles, development of new medicines, study of hemodynamics, simulations of organ systems, or development of various diagnostic techniques [1,2]. The combination of microfluidics and machine learning (ML), mostly referred to as intelligent microfluidics, is an innovative approach which uses the advantages of both fields [3–5]. On one hand, microfluidics focus on a precise control and manipulation of sub-millimeter structures and the flow of fluids at the micrometer scale. On the other hand, ML is an effective tool for processing and analysing large datasets. In particular, it can be used to find causality between biochemical factors, various physical factors and physical properties of used materials, and the structure of microfluidic devices. Some of the most important applications of intelligent microfluidics include isolation and detection of CTCs (Circulating Tumor Cells) in blood samples [6], or separation and analysis of RBCs with low elasticity related to several diseases such as anemia [7] or malaria [8]. Another extensive area of application of ML in connection with microfluidic techniques is the processing and extraction of data from video sequences of biological experiments [9–11].

Although recent progress in microfabrication allows relatively fast prototyping of multifunctional microfluidic devices, this approach still has very significant time, financial, and technological constraints. One of the possible alternatives is computer simulation, which can be used for much more cost-effective design of microfluidic devices with required structures and properties. Furthermore, numerical models provide a more detailed insight into the behaviour of the examined blood flow. For example, while it is practically impossible to extract the full information about the movement, deformation, and rotation of each

cell from a laboratory experiment, this can be easily obtained from a simulation experiment. Hence, computer simulations can be very helpful in investigation of hemodynamics and development of new diagnostic techniques. Our implementation of the simulation model is realized as an extension of ESPResSo [12], and it can be used to model blood cells as elastic objects in the fluid flow. An important part of our approach is a continuous validation of our model by comparison of simulation and laboratory experiments published in [13].

Increases in the complexity of simulation experiments naturally makes them more time and computationally intensive. Hence, as the next step, we can use data from the simulation experiments and ML to develop more efficient methods to model the flow of blood. The main advantage of using the simulation data is that they are virtually unlimited, which allows us to have a much better insight into simulation experiments. When training our models, mostly neural networks (NN), we aim to use only the type of data that are also obtainable from the video analysis of in vitro experiments, see in [14,15]. However, it is still difficult to acquire them from the medical environment.

Since RBCs are the dominant particles present in the blood plasma, their proper modelling is the key part when modelling blood flow. An accurate description of the elasticity of a RBC is necessary for such models, and it also has a significant application in diagnostic medicine. Membrane, which forms the surface of a RBC, is deformable, allowing the cells to easily flow through blood vessels. Elasticity of this membrane is affected by different natural factors (such as the age of the cell), but the most important ones are various diseases, such as sickle cell disease, leukemia, malaria, or diabetes. Deformability-based sorting RBCs using the microfluidic ratchet device was introduced in [16]. We try to solve the same problem without using a special microfluidic device. Our results are based only on geometrical properties of cells which can be recorded on video.

The problem of determining a RBC's elasticity using simulation experiments was investigated in [17], where the main goal was to classify a RBC's elasticity into predefined categories. In this paper we extended the solution to the problem of categorisation of values of elastic parameters to a regression task, where we determine for each cell the value of the regression function determining the elasticity of the given cell. An ultimate but distant goal should be the possibility of determining elastic properties (or, alternatively, elastic coefficients) of RBCs from video recordings of laboratory experiments. In our numerical model the elasticity of a RBC is controlled by five parameters, described in detail in Section 2.1. For several reasons, including a good physical interpretation and the most substantial impact on the overall elastic properties of RBCs, we consider the stretching parameter of the edges of a triangulation (given by the stretching coefficient k_s) to be the key parameter. Values of the coefficients are shown in Table 1. In this paper, we propose machine learning methods for determination of elastic parameter of the cell membrane. The novel approach is in the use of data. The data come from computer simulations that are based on laboratory experiments. The information in these data is accurate so it can be used for supervised teaching. Another advantage is that these data do not need to be marked manually, but the computer will do it. With the help of simulations, we can obtain a sufficient amount of data for neural network input. We may choose to use only data obtained from video analysis of such experiments. However, for methodological reasons, we also use complete simulation data for comparison. We compare the quality of the estimation of the elastic coefficients obtained from the limited data with those obtained using the full simulation data.

Table 1. Overview of used simulation parameters.

Coefficient	Value
stretching coefficient (k_s):	5×10^{-6} N/m
bending coefficient (k_b):	3×10^{-19} Nm
coefficient of local area conservation (k_{al}):	2×10^{-5} N/m
coefficient of global area conservation (k_{ag}):	7×10^{-4} N/m
coefficient of volume conservation (k_v):	900 N/m^2

2. Materials and Methods

2.1. Simulation Model and Input Settings

A physics-informed neural network is the subject of the current research [18,19]. In our work, we exploit the implicit information of the ML model by using data that correctly capture the physics.

The source data we use in the machine learning-based estimation of k_s coefficients were obtained from several simulation experiments. Their settings and parameters were based on real experiments and some were already used in the study [20]. All simulation experiments were performed using the freely available open-source software ESPREesSo [21] and its modules Lattice-Boltzmann and Object-in-fluid [22]. Blood flow simulations typically involve two parts of the model—the fluid and the cell membrane. These are coupled and interact with each other in the form of forces. Our model uses the Lattice-Boltzmann method [23] for the fluid, a spring network model for the cell membrane, and a dissipative version of the IBM (Immersed Boundary Method) for their interconnection [24].

For all simulations, the same channel and fluid flow parameters were used. The simulation channel had a cuboid shape with four walls with dimensions of $60 \times 40 \times 40$ µm (see Figure 1). Periodic fluid properties were ensured in the fluid flow direction (in the x-axis direction). The fluid was discretised into a three-dimensional grid with a spatial step of 1 µm. The kinematic viscosity of the fluid was 1.3×10^{-6} m^2/s and the density was 1.025×10^3 kg/m^3. The coefficient of friction providing the fluid–object interaction was equal to 1.414. External forces were used to set the flow in motion with values that provided a maximum velocity of approximately equal to 0.03 m/s.

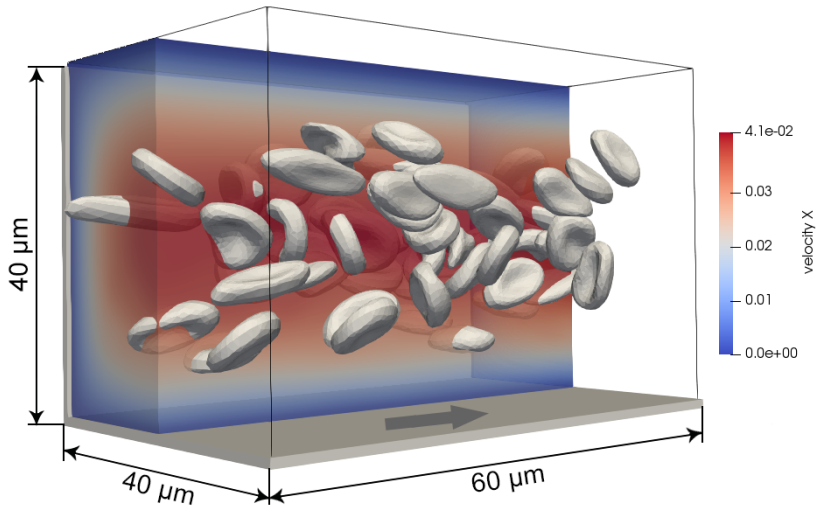


Figure 1. An illustration of the channel with cells (from simulation *Sim9a*). The colour represents the fluid velocity (blue for slower and red for fast).

The interaction between cells defined as cell-cell interaction was modeled using the membrane_collision potential with parameters $mc_a = 0.01$, $mc_n = 1.0$, $mc_{cut} = 0.4$. Interactions between cells and walls of the channel were modelled using the soft_sphere potential with parameters $soft_a = 0.00035$, $soft_n = 1.0$, $soft_{cut} = 0.5$.

The red blood cells are modelled by a surface network with 374 nodes, which in the relaxed state assumed a typical biconcave shape with dimensions of $7.82 \times 7.82 \times 2.58$ µm and a volume of 90.75 µm^3. The cells were filled with the same fluid as was in their surroundings. We use five types of elastic forces to model the elastic properties of the cell membrane. Each corresponds to one elastic modulus and its corresponding parameter: elastic modulus (preservation of edge length), bending modulus (preservation of angles between adjacent triangles), local area preservation modulus, global area preservation modulus, and volume preservation modulus. A schematic representation of the RBC

model and the individual elastic forces is shown in Figure 2. The calibration of the elastic coefficients for a healthy well deformable RBCs and their subsequent validation is discussed in more detail in [25]. The values of the coefficients used are shown in Table 1.

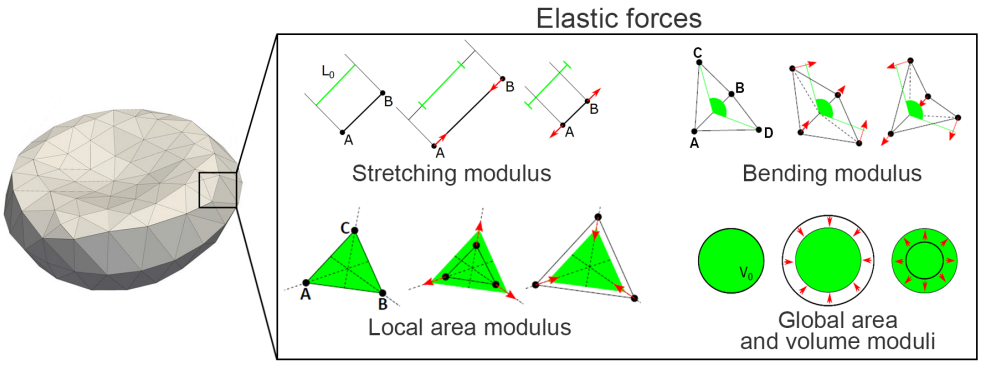


Figure 2. A schematic illustration of the cell membrane. Each individual cell is modeled by a spring network of boundary points bound by elastic interaction.

In straight microfluidic channels, a parabolic velocity profile is typical, which ensures a symmetric distribution of cells in the flow around the axial axis of the channel. As elastic RBCs, due to the velocity gradient, migrate toward the center of the microchannel, hydrodynamic collisions between cells lead to the movement of stiffer RBCs toward the vessel wall. Thus, hemodynamic phenomena such as marginalisation of stiffer RBCs [26,27] or cell-free layers [28,29] can be observed in dense cell suspensions. Several factors influence the migration of particles across streams in confined-space flow suspensions, with reduced elasticity of a certain fraction of cells generally considered to be a key factor. In a previous study by [17], we worked with only two levels of elasticity of RBCs, healthy and damaged (stiffer). In the present work, RBCs with three and nine elasticity levels, respectively, are included in the individual simulations, which both represents a shift towards a more realistic model of blood flow and creates the need to address the issue from a more holistic perspective. Thus, the individual simulations differ by varying the value of the elasticity coefficient k_s, while the other elasticity parameters remain set at the same values. Overall, we worked with nine cell types with the k_s values shown in Table 2. A value of 0.005 corresponds to healthy cells with good elasticity. In contrast, elasticity of cells with a set value of $k_s = 0.03$ corresponds approximately to the reduced elasticity of malaria-infected cells at stage 3 of the disease (according to the optical tweezers stretching experiment [30]).

Table 2. Overview of k_s values in each simulation.

Simulation Name	Values of Stretching Coefficients (k_s)
Sim3a	0.3, 0.005, 0.03
Sim3b	0.15, 0.015, 0.009
Sim3c	0.225, 0.1, 0.05
Sim9a	0.005, 0.009, 0.015, 0.03, 0.05, 0.1, 0.15, 0.225, 0.3
Sim9b	0.005, 0.009, 0.015, 0.03, 0.05, 0.1, 0.15, 0.225, 0.3

2.2. Description of Obtained Simulation Data

The initial arrangement of RBCs was random in the simulations and the number of recorded simulation steps was 3400 for *Sim3a*, *Sim3b* and *Sim3c* and 1240 steps for *Sim9a* and *Sim9b*, which corresponded to RBC movement in the channel of approximately 5.5 µm. In doing so, the recorded steps corresponded to every 2000th internal step of the simulation, which is sufficiently detailed for the purposes of training the methods. Because of the need to steady the flow in the run-up part of the experiment, we did not use the first 300 recordings for the simulation outputs. To create a balanced dataset, each

red blood cell type (by elasticity) was represented by an equal amount of data. For the *Sim*3 experiments, we simulated nine elasticity types, six red blood cells of each type with 3100 records for each. For *Sim*9, there were also nine different types with six cells of each type with 940 records. In total, we simulated a total of 54 red blood cells in both cases, *Sim*3 and *Sim*9. For example, we used a similar procedure in the study [17]. For each internal simulation step of the ESPreSso module, the current position of each non-stationary point in the blood flow, and hence all red blood cell triangulation points, is calculated. Since the scale of these data is huge, usually the basic position and velocity data of the significant points of each cell are stored, such as: 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 x, y, z-axes, the x, y, z-coordinates of the extreme points of the cell triangulation (according to the minimum and maximum coordinates in the direction of each axis, as shown in Figure 3), the velocities of the extreme points of the cell determined by the components in the x, y, z-axes direction, the cell volume or cell surface.

Figure 3. 3D cover cuboid of RBC from simulation and two 2D cover rectangles of RBC from the video.

When selecting and editing the data, we considered what information we could extract from the actual video footage. From the information contained in the simulation outputs, we can best determine the extreme points of a red blood cell in real experiment, while the center of the cell, the speed of its movement and the movement of the extreme points, the volume, or the overall surface of the cell are much more difficult to extract, if at all. Therefore, in our work, we focused on the use of data representing the projection of information from 3D to two-dimensional space according to the xy-axes and xz-axes. By subtracting the positions of the extreme points in the direction of each axis, we can also extract from the data information about the size of the "bounding box" rectangle (or cuboid) that bounds the blood cell, for each time step (see Figure 3). In individual computational experiments designed to the train machine learning methods, we generally used the following types of simulation data modifications for input to the neural network:

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

An overview of datasets from different projections is in Table 3.

Table 3. Overview of used datasets.

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	

We use the data in the x-axis direction in each option because the fluid in the channel flows along this axis, and thus the cell elasticity effect is most observable along this axis. We used data using coordinates along all three axes in order to verify the correctness of the neural network models we used and also to see to what extent we can extract information about the elasticity of RBC from the full simulation data.

2.3. Preprocessing and Data Augmentation

The dataset used, obtained from the simulation experiments, had to be subsequently transformed to be suitable for training the machine learning model.

Since the simulation data are meant to substitute the information from the real video recordings, the cell velocity information is implicitly included according to the change of the x-coordinate values in the downstream recordings. Therefore, we divided the resulting dataset into time windows consisting of a sequence of consecutive simulation records. In video processing, they would correspond to the sequence of subsequent frames. The length of the sequence may also depend on the quality of the tracking/tracking of the individual RBCs. Thus, the window size w corresponding to the length of the sequence will represent an extremely important hyperparameter of the model.

In computational experiments consisting in learning individual neural networks, it took values from the set $w = \{5, 10, 20, 30, 40, 50\}$. Once w was chosen, the whole dataset was divided into time sequences (time series) of size w, producing training data (x_{wi}, y_i), where x_{wi} is the time sequence and y_i is the actual value of the stretching coefficient k_s for the given sequence.

We applied standardisation and normalisation techniques described in [31] to the simulation data in order to speed up the training process of the model. We applied the standardisation to the values of the extreme points according to the y and z axes and the size of the sides of the bounding rectangle (or cuboid), so that the data were transformed to have a mean of 0 and a standard deviation of 1. Then, for each training example separately, the attributes containing information about the coordinates of the extreme points according to the x axis were adjusted. The minimum value of the attribute x_i in a given window was subtracted from the value of the attribute x_i for each record within the time window:

$$x_{ij\_transformed} = x_{ij} - \min(x_i) \tag{1}$$

In this way, we normalize the training examples while implicitly preserving information about the rate of red blood cell movement.

The generated data were then divided into three parts:

- *training data*—for the training of the model
- *validation data 1*—for the validation of the model after each epoch
- *validation data 2*—data used to compare different models

The partitioning was done in such a way as to avoid data leakage. The data leakage occurs when the information that was used to train the machine learning model is also used to validate or final test it.

Such a situation would occur if the model received only training data as an input, which it automatically divided into training and validation parts, since the next step in creating the dataset was to increase the number of training data for the machine learning by augmenting the training data by noise and transforming the original positions.

The noise n generated from a random normal distribution with mean 0 and standard deviation 1 was used, which was multiplied by a constant of 0.1 for each component of the training example. One offset s was randomly generated from a uniform distribution from the interval $(-0.25, 0.25)$ for each training example. The number of augmentations a was determined as:

$$a = \frac{10,000}{\frac{3100-50}{w}} - 1 \tag{2}$$

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

2.4. Types of Tested Neural Network Architectures

2.4.1. Long-Short Term Memory

The LSTM (Long-Short Term Memory) architecture [32] is typically used for input data in the form of a time sequence. Our network, using this architecture, was composed of four layers, sequentially with 512, 64, 32 and 10 hidden neurons with hyperbolic tangent activation function and recurrent sigmoid activation function. Each LSTM layer was followed by a dropout in which 10% of the neurons were fired. The result is flattened (we use *keras.layers.Flatten()* from [33]) and travels to a pair of fully connected layers with 1024, 512 neurons and ReLU activation function and also a fully connected layer with 1 neuron and linear activation function, which is also the output layer.

2.4.2. CNN-LSTM

The CNN-LSTM (Convolutional Neural Networks—Long-Short Term Memory) architecture [32] involves the use of convolutional layers CNN to extract features from the input data combined with LSTM to support prediction from the sequence. One reason for combining these layers is motivated by the analysis in [34] and papers, such as [35], which suggests that the performance of LSTM can be improved. This architecture is suitable for problems that:

- input with spatial structure, such as the 2D structure of pixels in an image or the 1D structure of words in a sentence, paragraph, or document
- have temporal structure in their input, such as the order of images in a video or words in a text, or require the generation of output with temporal structure, such as words in a textual description

We have created two network versions of such a CNN-LSTM network. The first one, *CNN-LSTM Conv1D*, used 1D convolution in traversing the temporal sequence and the second one, *CNN-LSTM Conv2D*, used 2D convolution. First, we reduced the variance in the input data by passing the input through a pair of convolutional layers with 256 filters, a step size of 1, and alignment to the same size with ReLU activation. For *CNN-LSTM Conv1D*, the size of the filters is for both layers the triple of the time window width. For *CNN-LSTM Conv2D*, the size of the filters are the triple of the time window widths for the first layer and then 4×3 for the second layer. After the CNN pair, the result is adjusted by layer pooling with a filter size of 2×2. The multidimensional intermediate result is smoothed ($keras.layers.Flatten()$) and is passed to the LSTM layer with 256 hidden neurons and the ReLU activation function. Finally, we passed the output from the LSTM to a fully connected layer with 512 neurons and ReLU activation function. The network terminated with a linear output layer with one neuron.

The architectures of the neural networks used for our experiments are shown in Figure 4.

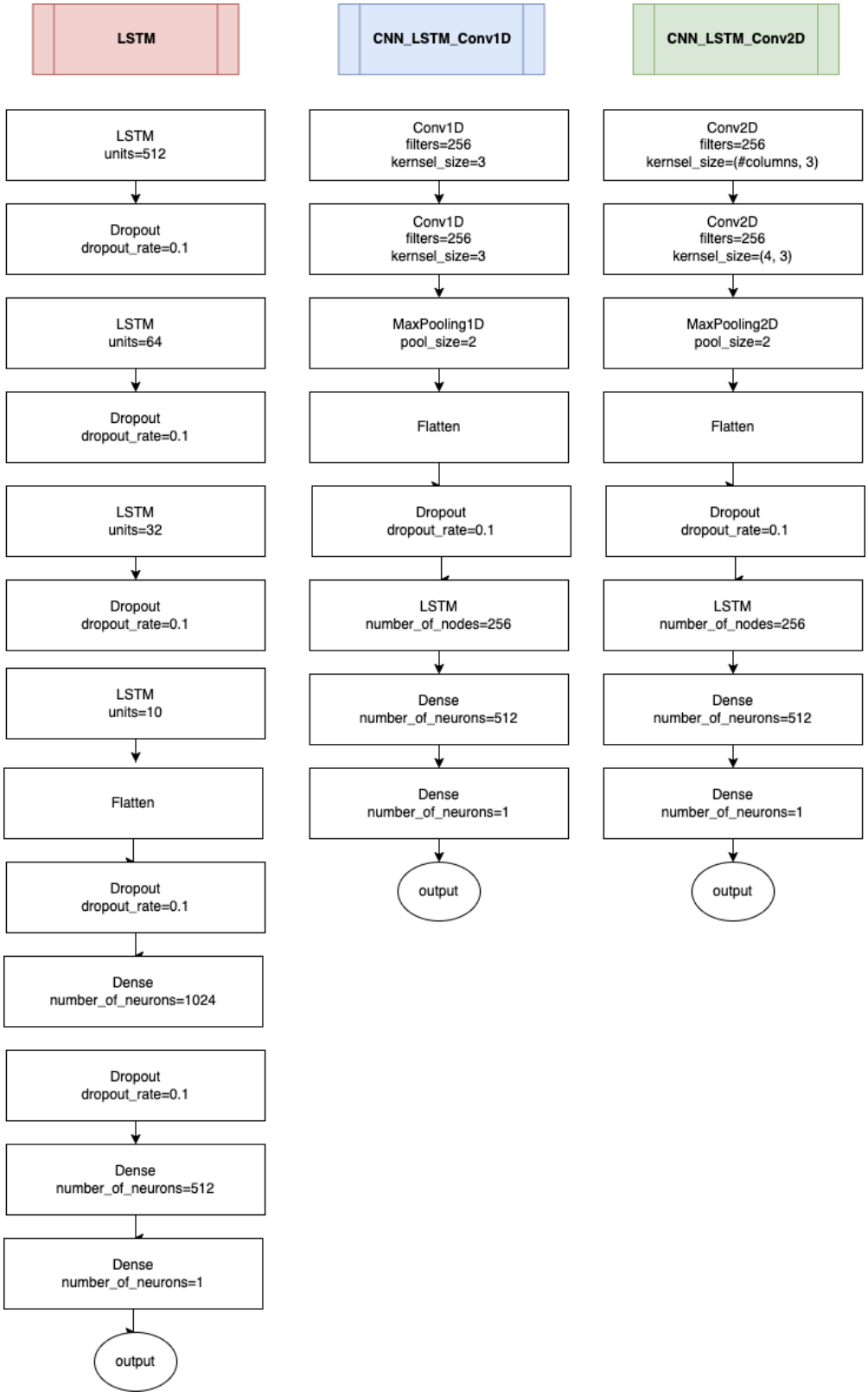


Figure 4. Diagrams of the used neural network architectures.

All experiments were performed using Python 3.8 and the Tensorflow library (Keras) was used to create the neural networks. The training was performed on a machine with an ADM Ryzen 5 5600H with Radeon Graphics, 16 GB RAM and a NVIDIA GeForce RTX 3060 Laptop GPU graphics card.

3. Results and Discussion

3.1. Results of Using CNN-LSTM Networks to Simulate Triplets of RBC Types

We trained a total of 84 different neural network models, for each combination of architecture type (3 types), data (4 used datasets), and window size w (7 possibilities) with the *Sim*3 data. We used the mean absolute percentage error as the loss function defined in Equation (3). The error MAPE (Mean Absolute Percentage Error) of the prediction and the actual value is calculated as:

$$MAPE(y_{true}, y_{pred}) = 100 \times |\frac{y_{true} - y_{pred}}{y_{true}}| \quad (3)$$

Figure 5 shows the MAPE for each possible combination of the above options. The plot shows that as the size of the time window for training increases, the MAPE also increases. This trend was likely due to the addition of noise and bias in the augmentation.

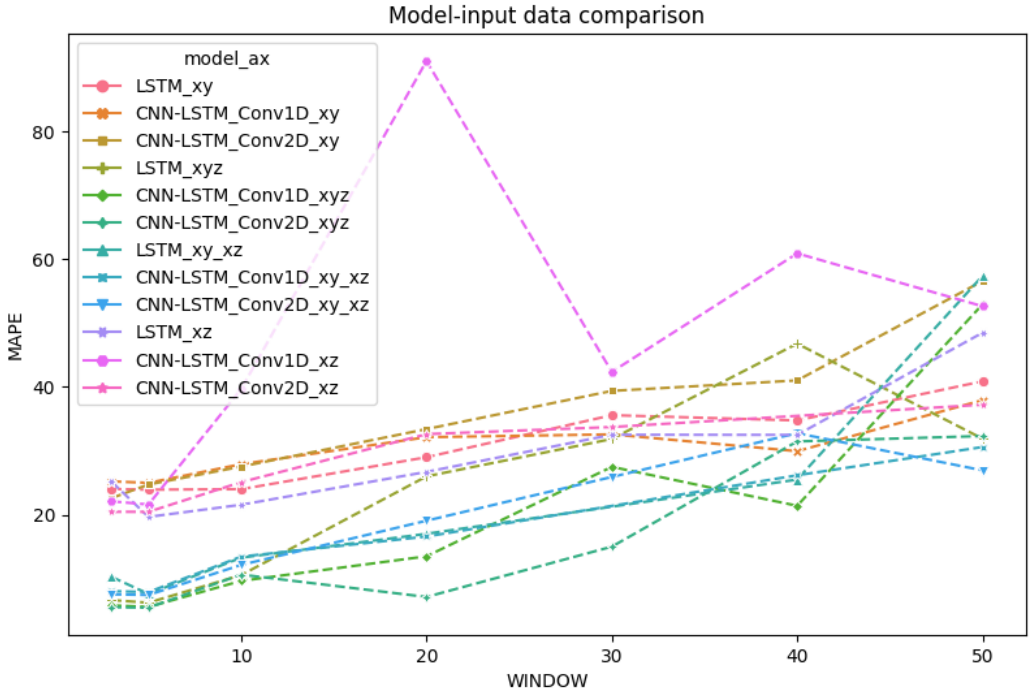


Figure 5. Comparison of the training MAPE for each combination of w, subset of data and type of NN model.

The MAPE is in the 20% range for the data that simulate the data obtained from the video recordings, while the combination of these data can reduce the error to half. The data using information according to all three xyz-axes reach the lowest level, namely 5%, which confirms our hypothesis that information about the elasticity of the observed cell can be obtained from the data used. Table 4 shows the MAPE values by architecture and the subset of data used for the value of the hyperparameter w for which the resulting model had the smallest error. The results show that the most accurate neural network model was the *CNN-LSTM Conv2D* for data along the xy-axes, xyz-axes, xy_xz-axes with the size of the time window w used being successively 3, 3, 5, and along the xz-axes, the best model was the LSTM with $w = 5$. *CNN-LSTM Conv2D* with $w = 3$ had the lowest MAPE value of all the trained models, namely 4.58%. For the 2D projection, LSTM with $w = 5$ was the best for the z data according to the xz-axes, with a MAPE value of 20.46%. For the concatenation of the two 2D projections, the MAPE was 7.97% for *CNN-LSTM Conv2D* with $w = 5$.

Table 4. MAPE values according architecture, the subset of data used for training and the value of the hyperparameter w. The lowest (best) value for each subset of data 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-LSTMConv1D	xy	3	22.353682
CNN-LSTMConv1D	xz	5	22.477804
CNN-LSTMConv1D	xyz	3	5.173491
CNN-LSTMConv1D	xy_xz	3	9.300526
CNN-LSTMConv2D	xy	3	20.930489
CNN-LSTMConv2D	xz	3	20.559206
CNN-LSTMConv2D	xyz	3	4.578221
CNN-LSTMConv2D	xy_xz	5	7.974189

The MAPE distribution of all the elastic blood cell types we simulated can be seen in Figure 6 for *CNN-LSTM Conv2D xyz* with $w = 3$ and in Figure 7 for *CNN-LSTM Conv2D xy_xz* with $w = 5$. The green triangles represent the average MAPE for each value of the elastic coefficient k_s, the yellow line indicates the median of the values. The plots on both sides show the same phenomenon, plots (b) and (d) additionally show outliers.

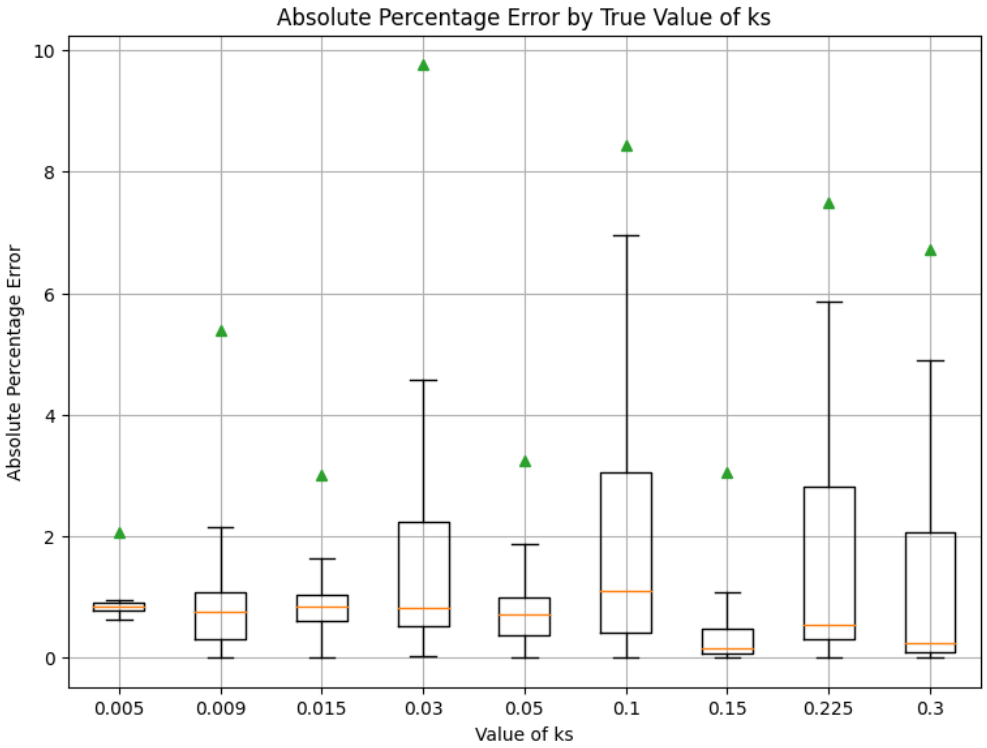


Figure 6. *Cont.*

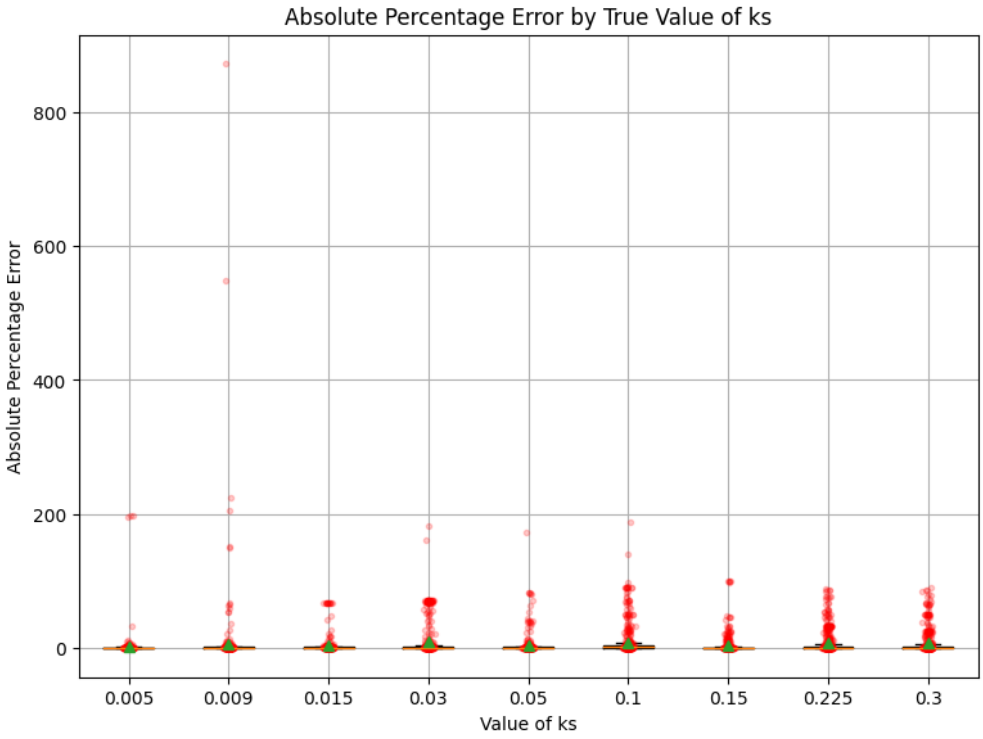


Figure 6. Boxplots of MAPE for each RBC elasticity type for the architecture *CNN-LSTM Conv2D*, the subset of data used *xyz* with $w = 3$.

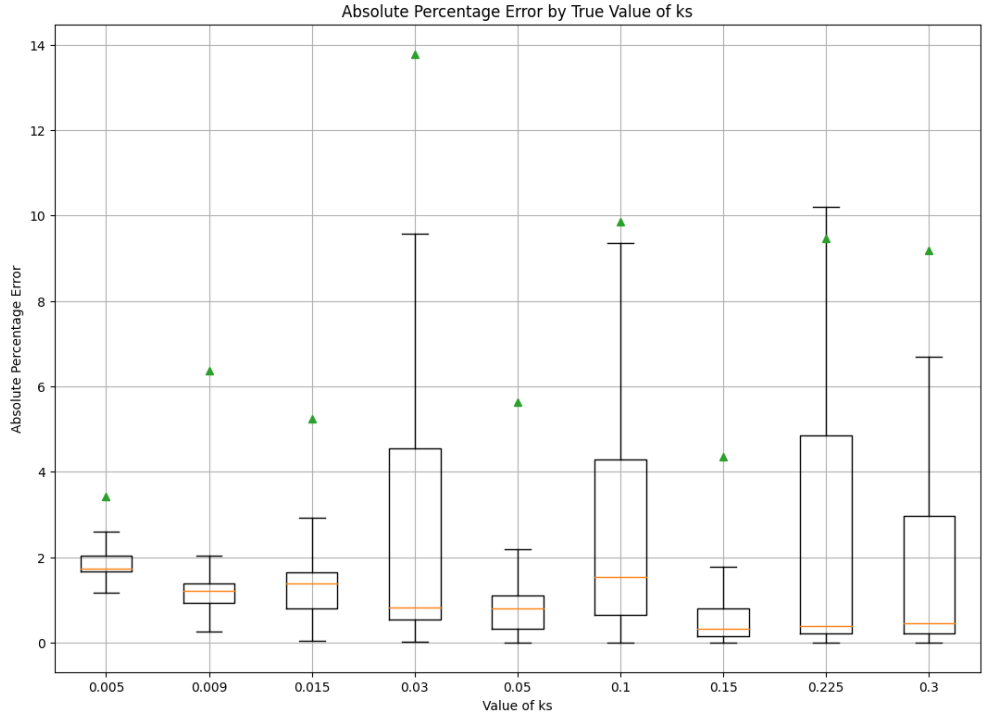


Figure 7. *Cont.*

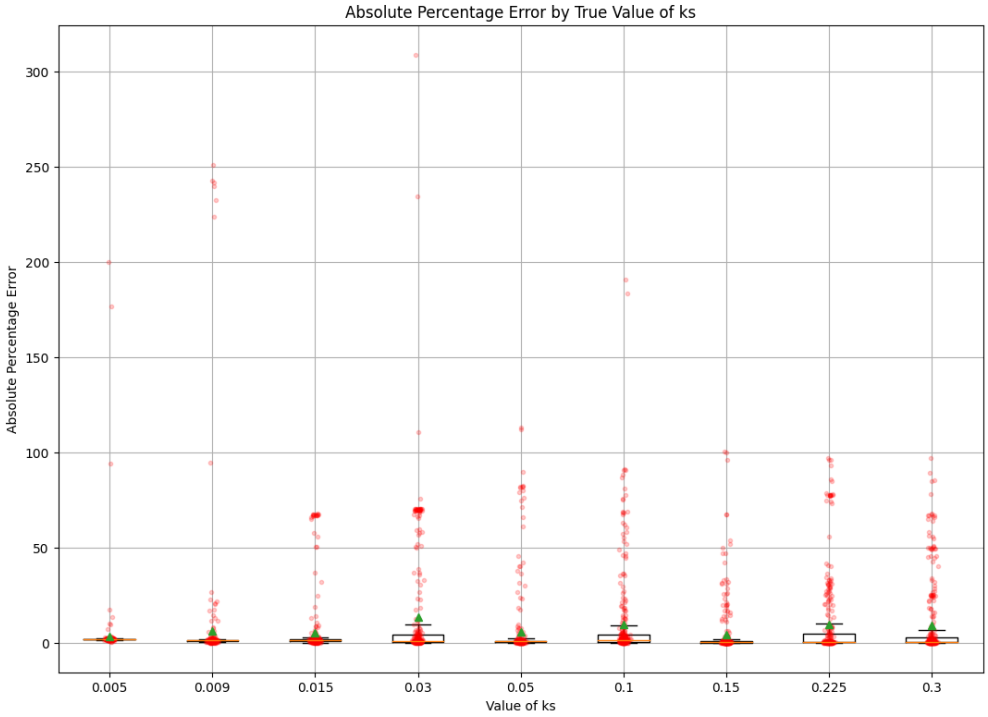


Figure 7. Boxplots of MAPE for each RBC elasticity type for the architecture *CNN-LSTM Conv2D*, the subset of data used xy_xz with $w = 5$.

The largest average MAPEs are seen at elasticity values of 0.03 and 0.1 for the *CNN-LSTM Conv2D* models xyz with $w = 3$ and xy_xz with $w = 5$ in Figures 8 and 9, respectively. For both models, we can notice that for the part of the corpuscles with elasticity of 0.03, the predicted value was at 0.009, and similarly for the elasticity value of 0.1, where the more significant part of the predictions is from the interval $(0.005, 0.03)$.

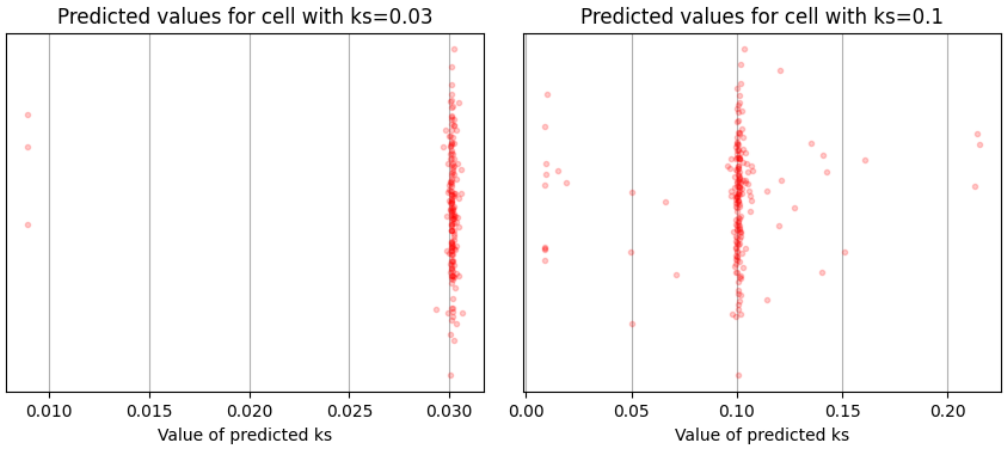


Figure 8. The largest average MAPE for elasticity 0.03 and 0.1 for the architecture *CNN-LSTM Conv2D*, the subset of data used xyz with $w = 3$.

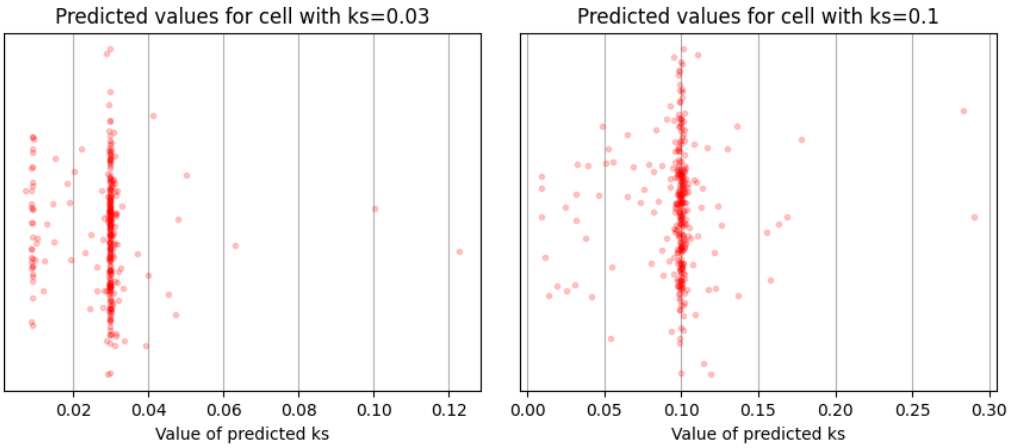


Figure 9. The largest average MAPE for elasticity 0.03 and 0.1 for the architecture CNN-LSTM_Conv2D, the subset of data used xy_xz with $w = 5$.

Table 4 shows the MAPE values by architecture, the subset of data used and the value of the hyperparameter w for which the resulting model had the smallest error. The results show that the most accurate neural network model was *CNN-LSTM_Conv2D* for data according to xy-axes, xyz-axes, xy_xz-axes with the used time window size w successively 3, 3, 5 and according to xz-axes the best model was *LSTM* with $w = 5$. The *CNN-LSTM_Conv2D* with $w = 3$ had the lowest MAPE value of all the trained models, namely 4.58%. For 2D projection, *LSTM* with $w = 5$ for the data from the along-axis xz, with a MAPE value of 20.46%. For merging the two 2D projections, the MAPE was 7.97% for *CNN-LSTM_Conv2D* with $w = 5$.

The results of the experiment described in the Section 3.1 suggest that it might be possible to determine the elasticity of a 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 the blood flow, e.g., along the xy-axes, the resulting prediction will contain a relatively large error, on average over 20%. If a video from two sides, xy and xz, is used, the prediction will achieve an average error of less than 8% of the true value. This result would be difficult to obtain from data measured in a real blood flow experiment. In the simulation experiment we have all the values of x, y, z coordinates for all points of RBC surface discretisation. Thanks to this we can estimate how big is the deviation of the real blood cell from its recording on the video. By fitting an estimate of the elasticity parameter k_s, we can estimate the error we have made compared to a situation in which we would have complete information. Our results show that when solving the k_s parameter estimation problem, we obtain better results for video recordings from multiple axes of symmetry of the monitored channel.

3.2. Use of Regression Neural Network for Red Blood Cell Elasticity Classification

As we mentioned in the introduction, the problem of determining the elasticity of RBCs as a classification problem has been studied before, and we assume that it is a more frequently-studied and an easier-to-solve problem. In order to be able to compare the results, we simply transformed the results of the regression method into a simple classifier. For this we used the same simulation data as in the Section 3.1, *Sim3a-c*.

We decided to investigate the ability of the neural network to predict cell elasticity approximately, and thus the ability to assign blood cells to categories. An intuitive way was to train a classification neural network. However, along with it, we developed yet another method of classifying the corpuscles that used the output of the regression neural network directly. Converting the output of the regression model into a classification consisted of de-averaging the elastic coefficient and then assigning it to the appropriate category. The categories were created by identifying eight *boundaries* among the nine elasticity values

used, with the boundary located midway between two adjacent coefficients (ordered in ascending order). This neural network is referred as *RegToClass*.

The neural network classification model had the same architecture as the regression neural networks and was created by alternating the last layer. The only change was in the last, output layer, which replaced one output neuron with nine, which was the number of categories, and the activation function was changed from linear to *softmax*. At the same time, we also changed the loss function to categorical cross-entropy and transformed the target variables to one-hot encoding.

The classification success rate for the best models according to the comparison in Table 4 can be consulted in Table 5. In all cases, we see improved classification performance for versions of the models directly optimized for the classification task. The largest difference in accuracy is almost 11.75% for the model that learned on data representing the projection onto 2D by the xz-axes (Figure 10), with a minimal improvement of 5.84% for the model with xyz data (Figure 11). Again, the bias is evident for the blood cells with an elastic coefficient value of 0.03, where the predicted class was 0.009, to a greater extent for the neural network model trained on the xz data. We also note the lower classification success for coefficient values of 0.225 and 0.3, for which the corpuscle is very stiff and the differences in elasticity are small, and this makes a correct prediction difficult. (It can be said that such stiff RBCs are rarely seen in reality).

Table 5. Comparison of classification accuracies of RegToClass and Classification neural networks.

Window Size w w	Subset of Data	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%

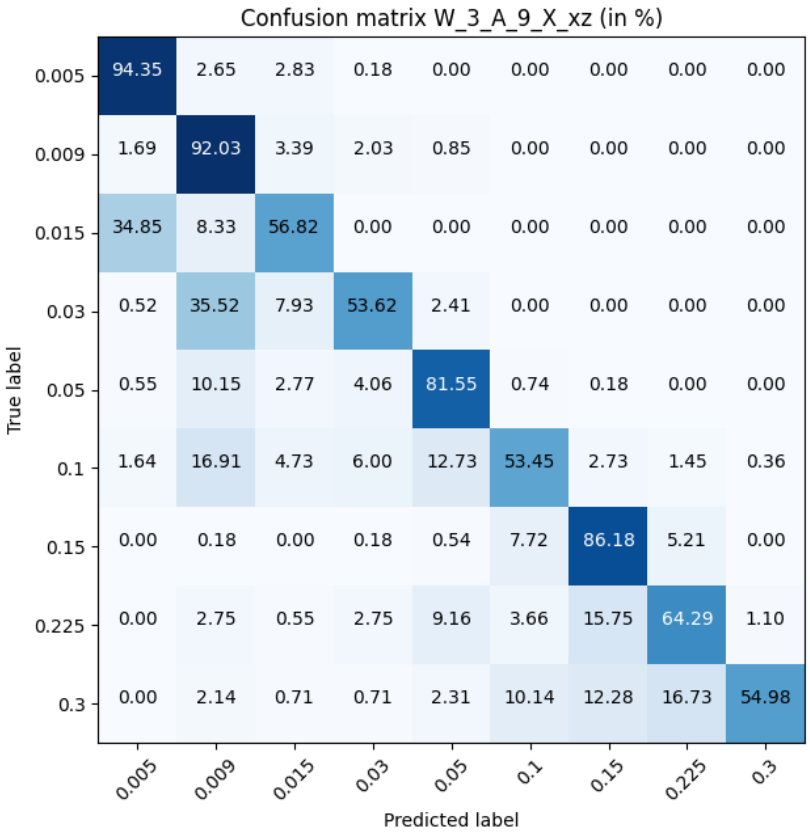


Figure 10. *Cont.*

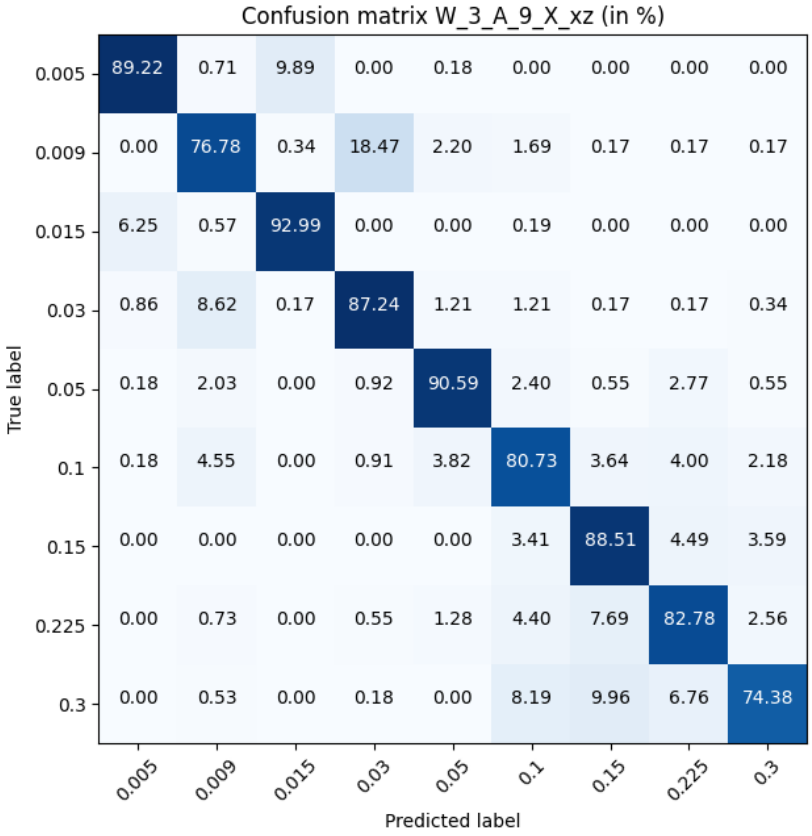


Figure 10. Confusion matrices of RBC classification for RegToClass (**up**) and Classification (**down**) neural networks *xz*.

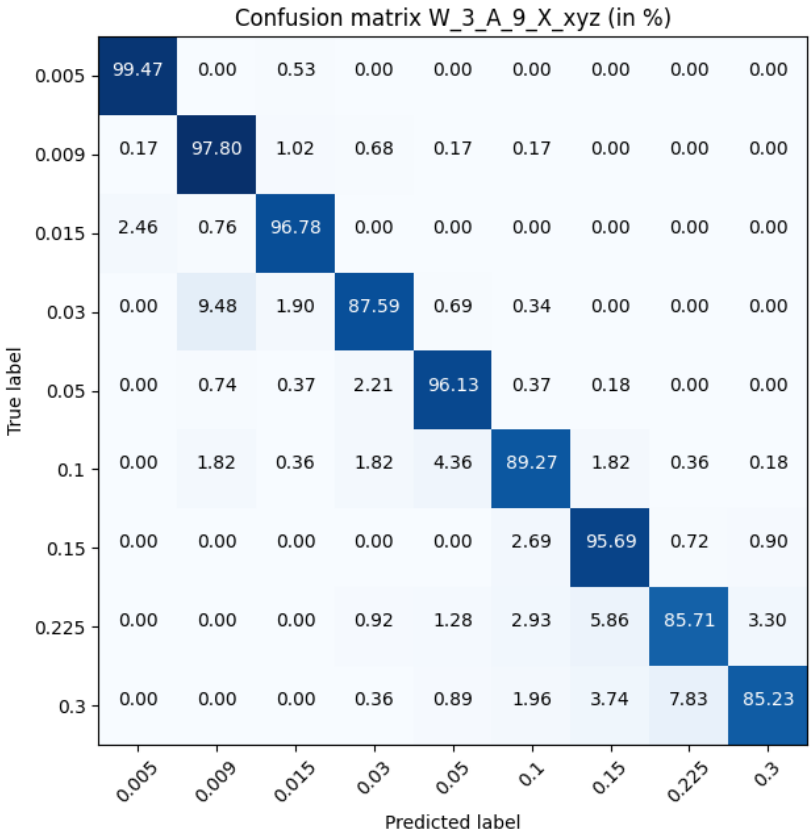


Figure 11. *Cont.*

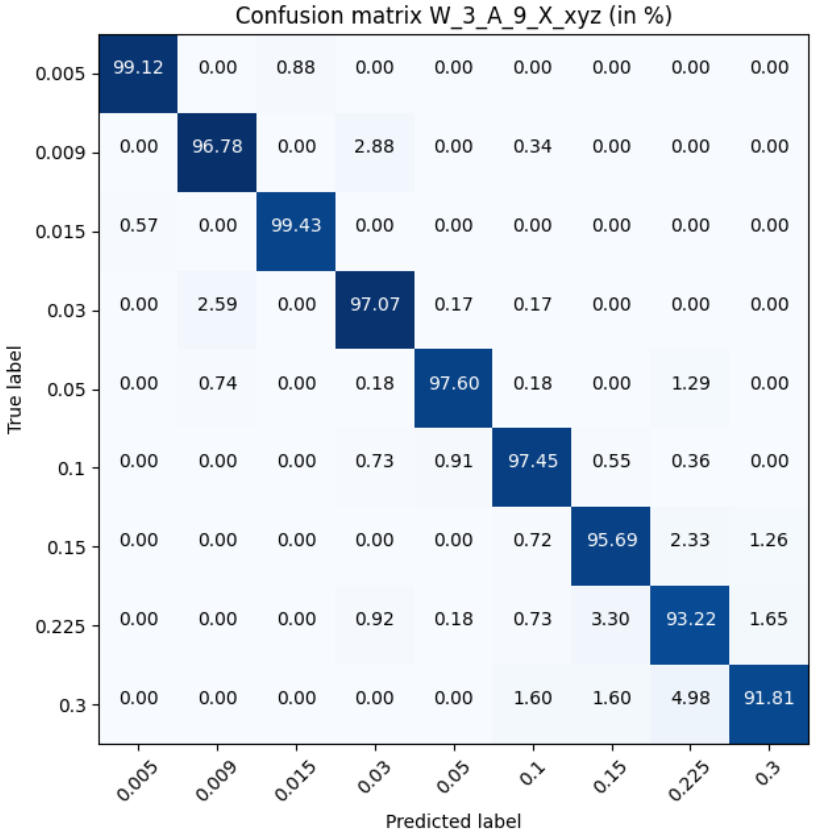


Figure 11. Confusion matrices of RBC classification for RegToClass (**up**) and Classification (**down**) neural networks *xyz*.

3.3. Validating Models on the Different Simulations

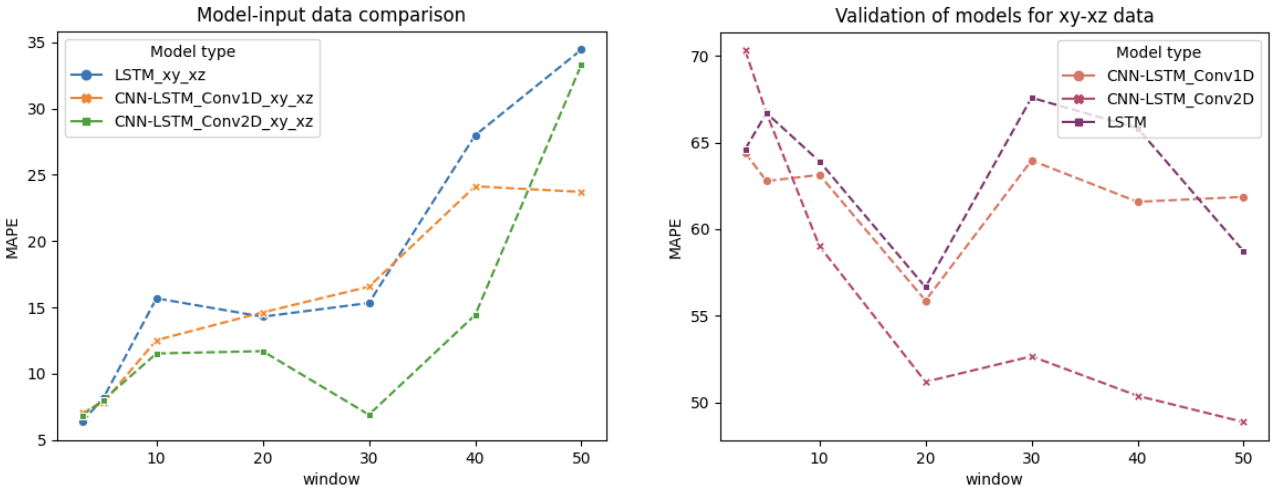
The previous experiments worked with a triplet of simulations, where each was divided into training, validation, and test parts. The next experiment focused on detecting model errors and possible overfitting. The dataset was constructed in a different way in this case:

- Training: one simulation with nine cell types (according to the elastic coefficient value $ks = 0.005, 0.009, 0.015, 0.03, 0.05, 0.1, 0.15, 0.225, 0.3$)
- Validation: one simulation with nine cell types (same as for training the model, but with different initial seeding) and the same simulation parameters

In the previous experiment, subsets of the data for the performance model were shown to be the best data along the xy and xz axes simultaneously (the subset of data labeled as xy_xz), not counting the subset of data xyz, which we cannot obtain in practice. For this combination of model and data subset we trained the models for different window lengths w. As in the previous experiment, we pre-processed the data and then expanded the data by a sufficient amount of data for a neural network. The resulting MAPE values are visible in Figure 12.

The validation showed a significantly degraded performance of the model despite our efforts to limit the possibility of overfitting by adding augmented data by noise and dropout. This deterioration was likely due to the data, or rather the components of the data, containing information about the y and z axes. The neural network overfitted, which in this case means that it over-focused on this subsection of the data by predicting the value of the elastic coefficient based on the position of the cell according to the channel. In the previous experiment, when the entire red blood cell trajectory was divided into parts with the required number of records equal to the parameter w and then divided into subparts for training and validation, which was random, this information was present in both datasets,

which explains the better performance of the model. This implies that despite splitting the dataset into four parts to prevent data leakage, such leakage did occur. The best model in such validation was again the *CNN_LSTM_Conv2D* model with window size $w = 20$, but again the MAPE was remarkably high.



(**a**) (**b**)

Figure 12. Comparison of (**a**) training MAPE and (**b**) validation MAPE of the different combinations of model architectures and the window size w.

An important observation, despite the significantly degraded results, was the observably better performance of the *CNN_LSTM_Conv2D* model in all experiments. Unlike the original *CNN_LSTM_Conv1D* architecture from the paper [36], this model contained a convolution filter of size (*4, 3*) (as opposed to number of features, *3*). This finding is further evidence that CNNs are one of the most powerful architectures of the present day.

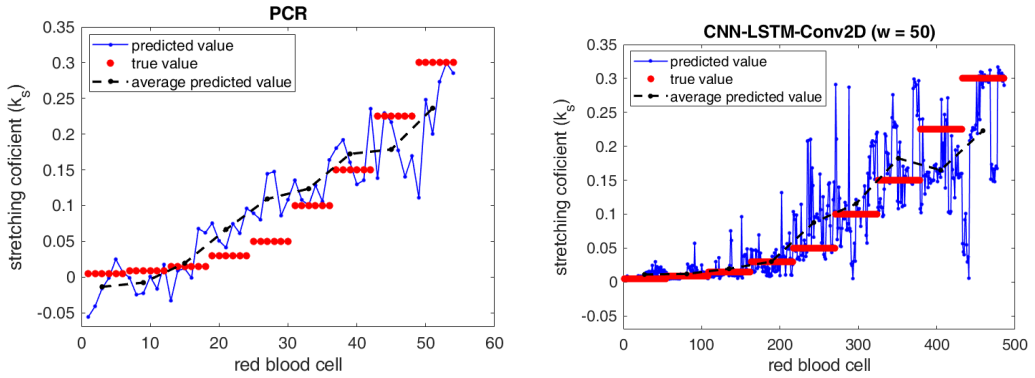
Comparison with Multiple Linear Regression

To better understand the quality of the results obtained using NN as described above, we solved a similar problem using classical regression tools. We used a modified MLR (Multivariate Linear Regression) and the same dataset (*dataset_xy_xz*). Regression coefficients were again calculated based solely on *Sim9a* data and then were used to predict k_s values in *Sim9b*. The obtained results were compared with the values obtained for the CNN-LSTM-Conv2D architecture and the parameter $w = 50$, for which we obtained the smallest MAPE value.

The input data matrix was created by arranging the data from the *dataset_xy_xz* for each simulation step, resulting in a matrix with dimensions 54 $\times$ 15,997. Its data were centered and scaled. Since the number of regression parameters is larger than the number of observations, the standard MLR method cannot be implemented due to the singular matrix formation when estimating covariances between independent variables. For this reason, it is necessary to first reduce the space of observations, for which we used the PCA (Principal Component Analysis) method. After performing the space projection using the PCA method, using the first seven principal components preserves approximately 97% of the information from the original data. This effective reduction of the 15,997-dimensional space is due to the significant "similarity" of the processed data. This combination of methods is called the PCR (Principal Component Regression) method, and we subsequently used it to predict k_s values in this reduced space.

The obtained results along with the actual values for each cell are plotted in Figure 13 part (a). Recall that the multivariate linear regression used the simulation data to estimate k_s for each cell just once; therefore, a total of 54 k_s values were predicted. For the estimation of k_s values using CNN-LSTM_Conv2D, k_s values were predicted (given a window size

of $w = 50$) for sequences of 50 consecutive positions of individual RBCs, so there were 486 estimated k_s values in total. In the Figure 13, we see that the MLR method was able to capture the increasing trend of k_s values, but the MAPE for this method reached as high as 140%. Compared to the MAPE of 48.86% using the CNN-LSTM-Conv2D model with a window size of $w = 50$, this is an expected deterioration. The PCR method can only capture linear dependencies in the data structure. In contrast, NNs are generally able to find even non-linear constraints in the data.



(**a**) (**b**)

Figure 13. Comparison of predicted (blue curve) and actual (red points) k_s values for individual cells using (**a**) the PCR method using the first seven principal components and (**b**) the CNN-LSTM-Conv2D model ($w = 50$). The black curve represents the average percentage error for cells with the same elasticity.

4. Conclusions

The results of our computational experiments and the performance of the obtained NNs in predicting the elasticity of RBCs have been described in detail in the previous sections. Recall that in our research we use the outputs from simulation experiments of RBCs motion in blood flow as input data for training NNs. Although simulations generally model and repeat real experiments, we are aware that this is only a computational model and not a record of biological data. However, we can modify the computational outputs to fit even very simple image recordings of in vitro experiments. This approach gives us, among other things, the opportunity to investigate theoretically how the predictions obtained from the experimental record differ from those obtained from the full computational data.

Our recent research goal is therefore not to compare the consistency of real and simulation experiments, but to focus on the differences in the capabilities of 3D models and their, video-recording-based, 2D descriptions. In [17], we showed that different results can be yielded by whether the video footage was acquired at the top or at the side of a channel with a flowing fluid.

In this study, with similar intent, we addressed the ability to predict the elasticity of RBCs using high-performance LSTM and CNN-LSTM networks. As expected, the networks that used full 3D information were the most successful in this task. In our case, this involved all three dimensions x, y, z of bounding boxes surrounding the RBCs (*dataset_xyz*). Second in order of success were networks where two 2D data describing the bounding box dimensions when viewed from the side and from above were used simultaneously as input (dataset_xy_xz). Networks for which only separate projections onto the xy-plane or onto the xz-plane were inputs showed the largest errors. For the design of biological experiments, we can therefore recommend that at least two devices capturing their progress in linearly independent planes should be used for video recording.

Among the NNs trained and investigated, the CNN-LSTM architecture using 2D convolutional layers was found to be the best performing. Acceptable accuracy was achieved in its standard use for elasticity prediction, described in Section 3.1. The hyperparameter w, describing the length of the sequence of data records used for training the network,

proved to be optimal in this case for low values of $w = 3$ or $w = 5$. For the processing of data from real experiments, this is a sufficient result, as it will not require long tracing of individual RBCs.

The weakest results were obtained when the model was experimentally validated on parallel simulations to test the transferability of the prediction between similar experiments. A way to improve these results is offered as a further possibility of research. However, we have shown that NN results are also significantly better in this area than using more conventional linear methods.

Author Contributions: Conceptualisation, K.B., H.B. and M.S.; methodology, S.M. and M.S.; software, S.M. and M.S.; validation, S.M. and M.S.; formal analysis, S.M.; investigation, S.M. and M.S.; resources, M.S. and H.B.; data curation, M.S.; writing—original draft preparation, S.M., H.B. and M.S.; writing—review and editing, H.B., K.B., S.M. and M.S.; visualisation, S.M. and M.S.; supervision, K.B.; project administration, K.B.; funding acquisition, K.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Operational Program "Integrated Infrastructure" of the project "Integrated strategy in the development of personalized medicine of selected malignant tumor diseases and its impact on life quality", ITMS code: 313011V446, co-financed by resources of European Regional Development Fund, and by the Ministry of Education, Science, Research and Sport of the Slovak Republic under the contract No. VEGA 1/0643/17 and by the Slovak Research and Development Agency under the contract No. APVV-15-0751.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in this study are freely available in the GIT repository: https://github.com/molcan23/RBC_NN, (accessed on 13 July 2022).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CIF	Cell in Fluid, Biomedical Modeling & Computation Group
CNN	Convolutional neural network
CTC	Circulating Tumor Cells
ESPResSo	Extensible Simulation Package for Research on Soft Matter
IBM	Immersed Boundary Method
LSTM	Long-short Term Memory
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MLR	Multiple Linear Regression
NN	Neural Networks
PCA	Principal Component Analysis
PCR	Principal Component Regression
RBC	Red Blood Cell
ReLU	Rectified Linear Unit

References

1. Afsaneh, H.; Mohammadi, R. Microfluidic platforms for the manipulation of cells and particles. *Talanta Open* **2022**, *5*, 100092. [CrossRef]
2. Li, C.; He, W.; Wang, N.; Xi, Z.; Deng, R.; Liu, X.; Kang, R.; Xie, L.; Liu, X. Application of Microfluidics in Detection of Circulating Tumor Cells. *Front. Bioeng. Biotechnol.* **2022**, *10*, 907232. [CrossRef] [PubMed]
3. Riordon, J.; Sovilj, D.; Sanner, S.; Sinton, D.; Young, E.W. Deep learning with microfluidics for biotechnology. *Trends Biotechnol.* **2019**, *37*, 310–324. [CrossRef] [PubMed]
4. Vasilevich, A.S.; Carlier, A.; de Boer, J.; Singh, S. How not to drown in data: A guide for biomaterial engineers. *Trends Biotechnol.* **2017**, *35*, 743–755. [CrossRef] [PubMed]

5. Raji, H.; Tayyab, M.; Sui, J.; Mahmoodi, S.R.; Javanmard, M. Biosensors and machine learning for enhanced detection, stratification, and classification of cells: A review. *Biomed. Microdev.* **2022**, *24*, 1–20. [CrossRef]
6. Liu, Y.; Li, S.; Liu, Y. Machine Learning-Driven Multiobjective Optimization: An Opportunity of Microfluidic Platforms Applied in Cancer Research. *Cells* **2022**, *11*, 905. [CrossRef]
7. Mantegazza, A.; Clavica, F.; Obrist, D. In vitro investigations of red blood cell phase separation in a complex microchannel network. *Biomicrofluidics* **2019**, *14*, 014101. [CrossRef]
8. Rizzuto, V.; Mencattini, A.; Álvarez-González, B.; Di Giuseppe, D.; Martinelli, E.; Beneitez-Pastor, D.; Samitier, J. Combining microfluidics with machine learning algorithms for RBC classification in rare hereditary hemolytic anemia. *Sci. Rep.* **2021**, *11*, 13553. [CrossRef]
9. Kajánek, F.; Cimrák, I. Advancements in Red Blood Cell Detection using Convolutional Neural Networks. In Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies—BIOINFORMATICS, Valletta, Malta, 24–26 February 2020; pp. 206–211.
10. Hervé, L.; Kraemer, D.C.A.; Cioni, O.; Mandula, O.; Menneteau, M.; Morales, S.; Allier, C. Alternation of inverse problem approach and deep learning for lens-free microscopy image reconstruction. *Sci. Rep.* **2020**, *10*, 20207. [CrossRef]
11. Ondrašovič, M.; Tarábek, P. Siamese visual object tracking: A survey. *IEEE Access* **2021**, *9*, 110149–110172. [CrossRef]
12. Cimrak, I.; Gusenbauer, M.; Jančigová, I. An ESPResSo implementation of elastic objects immersed in a fluid. *Comput. Phys. Commun.* **2014**, *185*, 900–907. [CrossRef]
13. Kovalčíková, K.; Cimrák, I.; Bachratá, K.; Bachratý, H. Comparison of Numerical and Laboratory Experiment Examining Deformation of Red Blood Cell. In Proceedings of the 7th International Work-Conference, IWBBIO 2019, Granada, Spain, 8–10 May 2019; pp. 75–86.
14. Bachratý, H.; Bachratá, K.; Chovanec, M.; Kajánek, F.; Smiešková, M.; Slavík, M. Simulation of blood flow in microfluidic devices for analysing of video from real experiments. In Proceedings of the 6th International Work-Conference, IWBBIO 2018, Granada, Spain, 25–27 April 2018.
15. Bachratý, H.; Bachratá, K.; Chovanec, M.; Jančigová, I.; Smiešková, M.; Kovalčíková, K. Applications of machine learning for simulations of red blood cells in microfluidic devices. *BMC Bioinform.* **2020**, *21*, 90. [CrossRef] [PubMed]
16. Lamoureux, E.; Islamzada, E.; Wiens, M.; Matthews, K.; Duffy, S.; Ma, H. Assessing Red Blood Cell Deformability using Deep Learning. *Lab Chip* **2021**, *22*, 26–39. [CrossRef]
17. Bachratá, K.; Buzáková, K.; Chovanec, M.; Bachratý, H.; Smiešková, M.; Bohiniková, A. Classification of Red Blood Cell Rigidity from Sequence Data of Blood Flow Simulations Using Neural Networks. *Symmetry* **2021**, *13*, 938. [CrossRef]
18. Kovacs, A.; Exl, L.; Kornell, A.; Fischbacher, J.; Hovorka, M.; Gusenbauer, M.; Breth, L.; Oezelt, H.; Yano, M.; Sakuma, N. Conditional physics informed neural networks. *Commun. Nonlinear Sci. Numer. Simul.* **2022**, *104*, 106041. [CrossRef]
19. Kovacs, A.; Exl, L.; Kornell, A.; Fischbacher, J.; Hovorka, M.; Gusenbauer, M.; Breth, L.; Oezelt, H.; Praetorius, D.; Suess, D. Magnetostatics and micromagnetics with physics informed neural networks. *J. Magn. Magn. Mater.* **2022**, *548*, 168951. [CrossRef]
20. Smiešková, M.; Bachratá, K. Validation of Bulk Properties of Red Blood Cells in Simulations. In Proceedings of the 2019 International Conference on Information and Digital Technologies (IDT), Žilina, Slovakia, 25–27 June 2019; pp. 417–423.
21. Arnold, A.; Lenz, O.; Kesselheim, S.; Weeber, R.; Fahrenberger, F.; Roehm, D.; Holm, C. Espresso 3.1: Molecular dynamics software for coarse-grained models. In *Meshfree Methods for Partial Differential Equations VI*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 1–23.
22. Cell in Fluid (CIF). Biomedical Modeling & Computation Group. Available online: https://cellinfluid.fri.uniza.sk/ (accessed on 21 April 2021).
23. Ahlrichs, P.; Dünweg, B. Lattice-Boltzmann simulation of polymer-solvent systems. *Int. J. Mod. Phys. C* **1998**, *9*, 1429–1438. [CrossRef]
24. Bušík, M.; Slavík, M.; Cimrák, I. Dissipative coupling of fluid and immersed objects for modelling of cells in flow. *Comput. Math. Methods Med.* **2018**, *2018*, 7842857. [CrossRef]
25. Jančigová, I.; Kovalčíková, K.; Bohiniková, A.; Cimrák, I. Spring-network model of red blood cell: From membrane mechanics to validation. *Int. J. Numer. Methods Fluids* **2020**, *92*, 1368–1393. [CrossRef]
26. Chen, Y.; Li, D.; Li, Y.; Wan, J.; Li, J.; Chen, H. Margination of stiffened red blood cells regulated by vessel geometry. *Sci. Rep.* **2017**, *7*, 15253. [CrossRef]
27. Zhang, X.; Caruso, C.; Lam, W.A.; Graham, M.D. Flow-induced segregation and dynamics of red blood cells in sickle cell disease. *Phys. Rev. Fluids* **2020**, *5*, 053101. [CrossRef] [PubMed]
28. Bento, D.; Fernandes, C.S.; Pereira, A.I.; Miranda, J.M.; Lima, R. Visualization and measurement of the Cell-Free Layer (CFL) in a microchannel network. In Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering, Porto, Portugal, 18–20 October 2017.
29. Balogh, P.; Bagchi, P. The cell-free layer in simulated microvascular networks. *J. Fluid Mech.* **2019**, *864*, 768–806. [CrossRef]
30. Suresh, S.; Spatz, J.; Mills, J.P.; Micoulet, A.; Dao, M.; Lim, C.T.; Beil, M.; Seufferlein, T. Connections between single-cell biomechanics and human disease states: Gastrointestinal cancer and malaria. *Acta Biomater.* **2005**, *1*, 15–30. [CrossRef] [PubMed]
31. Montavon, G. *Neural Networks: Tricks of the Trade*; Orr, G., Müller, K.-R., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7700.

32. Wahlström, N.; Lindholm, A.; Lindsten, F.; Schön, T.B. *Machine Learning: A First Course for Engineers and Scientists*; Cambridge University Press: Cambridge, UK, 2022.
33. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Available online: https://www.tensorflow.org/ (accessed on 12 August 2022).
34. Pascanu, R.; Gulcehre, C.; Cho, K.; Bengio, Y. How to construct deep recurrent neural networks. *arXiv* **2013**, arXiv:1312.6026.
35. Kim, T.-Y.; Cho, S.-B. Predicting residential energy consumption using CNN-LSTM neural networks. *Energy* **2019**, *182*, 72–81. [CrossRef]
36. Yan, R.; Liao, J.; Yang, J.; Sun, W.; Nong, M.; Li, F. Multi-hour and multi-site air quality index forecasting in Beijing using CNN, LSTM, CNN-LSTM, and spatiotemporal clustering. *Expert Syst. Appl.* **2021**, *169*, 114513. [CrossRef]