

Article

# Velocity Estimations in Blood Microflows via Machine Learning Symmetries

Gerardo Alfonso Perez \*  and Jaime Virgilio Colchero Paetz

Physics Department, Campus Universitario de Espinardo, Universidad de Murcia, Murcia 30100, Spain

\* Correspondence: ga284@cantab.net or gerardo.alfonsop@um.es

**Abstract:** Improving velocity forecasts of blood microflows could be useful in biomedical applications. We focus on estimating the velocity of the blood in capillaries. Modeling blood microflow in capillaries is a complex process. In this paper, we use artificial intelligence techniques for this modeling: more precisely, artificial neural networks (ANNs). The selected model is able to accurately forecast the velocity, with an  $R^2$  of 0.8992 comparing the forecast with the actual velocity. A key part of ANN model creation is selecting the appropriate parameters for the ANN, such as the number of neurons, the number of layers and the type of training algorithm used. A grid approach with 327,600 simulations was used. It is shown that there are substantial, statistically significant differences when different types of ANN structures are used. It is also shown that the proposed model is robust regarding the initial random initialization of weights in the ANN. Additionally, the sensitivity of the selected models to additional noise was also tested.

**Keywords:** microflow; blood; forecasting

## 1. Introduction



**Citation:** Alfonso Perez, G.; Colchero Paetz, J.V. Velocity Estimations in Blood Microflows via Machine Learning Symmetries. *Symmetry* **2024**, *16*, 428. <https://doi.org/10.3390/sym16040428>

Academic Editors: Marcin Michalak and Sergei D. Odintsov

Received: 26 January 2024

Revised: 22 March 2024

Accepted: 1 April 2024

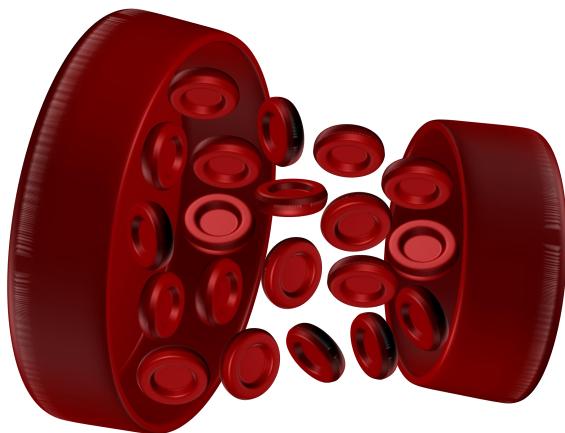
Published: 4 April 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

There has been an increased focus on blood microflow research [1]. There are some related interesting biomedical papers researching control of microflows [2] and drug absorption [3,4]. Technical advances [5–8] have made possible accurate measurements at small scale [9–11]. Milieva et al. [12] analyzed skin blood microflows using laser Doppler flowmetry (LDF) and compared healthy control cases to individuals with rheumatic diseases [13]. This LDF approach has been followed in some other interesting articles, such as Zhrebtssov et al. [14]. In Zhrebtssov et al., the authors used a combination of LDF with skin contact thermometry applied for the diagnostics of intradermal finger vessels. There is some research linking microflow (particularly microflow changes) and some illnesses. Wang et al. [15] mentioned that there appears to be an association between some type of microflows in the hepatic sinusoids and liver fibrosis and cirrhosis [16]. Microflow cytometry has been extensively used in medical applications, such as Lewis et al. [17] for the detection of prostate cancer [18]. There is even some research on blood microflow in the context of microchips. Pitts et al. [19] studied the contact angle of blood dilutions with common microchip materials.

Microflow measurement techniques have substantially advanced in recent years when compared to some of the first attempts, such as Stosseck et al. [20] in 1974. In this paper the authors measured blood microflows using electrochemically generated hydrogen. This technique requires 15 s per measurement. Another seminal paper is the one by Leniger-Follert [21] back in 1984, which analyzed microflows during bicuculline-induced seizures [22–24] in anesthetized cats. In recent years, technical developments have made the measurement task more accurate and less cumbersome. In the figure below (Figure 1), a graphical representation of red blood cell flow as a blood vessel constricts is shown.



**Figure 1.** Graphical representation of blood cell flow as the blood vessel constricts.

There are some very interesting modeling articles, such as Meongkeun et al. [25]. In this article, the authors numerically analyzed the impact of cell deformability (red blood cells) in blood microflow dynamics and concluded that the radial dispersion of red blood cells can have a significant impact on microflows. Mojarrab and Kamali [26] carried out another interesting modeling work doing a numerical simulation of a microflow sensor in a realistic model of a human aorta. This type of analysis has particular relevance in the context of atherosclerosis and aneurysm, which are conditions associated with an elevated mortality rate [27–29]. Having a more complete understanding of blood microflows might help with understanding these conditions better and potentially help with finding ways to avoid or treat them. In fact, there is some research, such as Jung et al. [30], mentioning that there is a correlation between microflow changes in cardiogenic shock (CS) [31–33] and outcomes in critically ill patients.

Given the complexity of the modeling of blood microflows, there has been an increased interest in using artificial intelligence techniques to estimate some important properties of blood microflows, such as the velocity of the blood flow. These artificial intelligence techniques typically require a set of inputs, which are reasonably related to an output. If everything else is equal, the more the inputs are related to the output, the more accurate the predictions are likely to be. Understanding the underlying issue might allow researchers to select the appropriate variables as inputs for the forecasting process. Some machine learning approaches intrinsically use some of the symmetries in the existing data. One well-known artificial intelligence technique is an artificial neural network (ANN) [34–37].

This is a biologically inspired approach that has proven useful when modeling several different types of processes [38,39]. Paproski et al. [40] used a machine learning approach in the context of extracellular vesicle microscale flow cytometry to generate predictive models in different types of tumors. Selecting which type of ANN architecture [41] to use is important in order to obtain accurate forecasts. Some of the parameters to be selected include the number of neurons ( $n$ ), the number of layers ( $m$ ) and the type of training algorithm used ( $a$ ). There are several ways to select these parameters. In this article, we propose an ANN model to estimate blood microflow velocities using a grid approach [42,43]. It should be mentioned that the use of artificial intelligence techniques is not without a variety of issues [44–46]. Lugnan et al. [47] mentioned that one issue is the high computational cost of some of these applications, as discussed also in [48–50]. An artificial intelligence approach can be useful in practical applications to generate accurate forecasts [51–53]. The main objective of this type of model is not to develop an underlying model of microflows that is easy to interpret, but to develop a tool that can generate accurate forecasts.

### 1.1. Objective 1

Creation of ANN model to estimate blood microflow velocities.

### 1.2. Objective 2

Estimate robustness of ANN with respect to noise in inputs.

## 2. Materials and Methods

The data analyzed in this article were extracted from Kloosterman et al. [54] (publicly available data). As described in Kloosterman et al., fertilized chicken (White Leghorn) embryos were incubated sitting horizontally. The authors generated optical access to the embryos by creating a viewing window in the egg shell. This window provided a microscope with optical access to the embryos. The technique used by the authors for the measurements was microscopic particle image velocimetry (micro-PIV). The authors used a Leica FluoCombi III with a magnification of 5 $\times$ . They also used a PCO Sensicam QE camera with a pixel size of  $12.9 \times 12.9 \mu\text{m}^2$  and covering an area of  $1.8 \times 1.4 \text{ mm}^2$  (full details available in [54]). We use the data from Kloosterman et al. as the experimental data on which our analysis is based.

Our proposed approach generates estimates of the velocity of the blood in a chicken embryo, which will be compared to the empirical results obtained in Kloosterman et al. This approach uses as input the positions (coordinates of the branch points  $x(t), y(t)$ ) as well as the diameter ( $d$ ) of the blood vessel, which ranges from 25 to 500  $\mu\text{m}$ , as well as the velocity ( $s$ ) in the blood vessels in the previous instant. The sampling rate for  $(x, y, d, s)$  was 9.7 Hz. The velocity was normalized to the interval [0, 1] using the expression in Equation (1). The velocity was also smoothed by using the average value of the previous ten measurements. After the initial filtering (including normalization), the data consisted of 379 data points for each variable. A table summarizing the descriptive statistics for each variable can be seen as Table 1.

$$s_{max} = \max\{s_1, \dots, s_q\}; s'_i = \frac{s_i}{s_{max}} \quad (1)$$

**Table 1.** Descriptive statistics of the data.

	$x$	$y$	$d$	$s$
Mean	1434.56	2904.10	27,155.15	0.35
$\sigma$	877.28	1488.75	32,054.03	0.15

An artificial neural network only requires a set of inputs  $x = x_1, x_2, \dots, x_n$  and an output  $y$ . These inputs and outputs are typically vectors. So  $x_1 = x_1^t = \{x_1^1, x_1^2, \dots, x_1^k\}$  and  $y = y^t = \{y^1, y^2, \dots, y^k\}$ , with the last time instant  $k$  included. The selection of which inputs to use is a critical step in the modeling process, but this might be restricted by the availability of the data. The objective of the model is to accurately estimate the empirically measured blood microflow velocity  $s(t)$ .

In this notation,  $t$  denotes time. The data were divided into two different datasets: a testing ( $T_1$ ) and a training ( $T_2$ ) dataset. See Equations (2) and (3).

$$T_1 = \begin{pmatrix} x_1(t) & x_2(t) & d(t) & s(t-1) & s(t) \\ x_1(t-1) & x_2(t-1) & d(t-1) & s(t-2) & s(t-1) \\ x_1(t-2) & x_2(t-2) & d(t-2) & s(t-3) & s(t-2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1(t-j) & x_2(t-j) & d(t-j) & s(t-j+1) & s(t-j) \end{pmatrix} \quad (2)$$

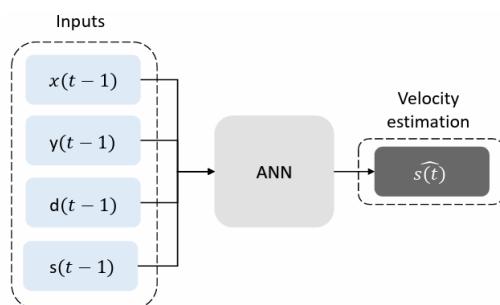
$$T_2 = \begin{pmatrix} x_1(t-j-1) & x_2(t-j-1) & d(t-j-1) & s(t-j-2) & s(t-j-1) \\ x_1(t-j-2) & x_2(t-j-2) & d(t-j-2) & s(t-j-3) & s(t-j-2) \\ x_1(t-j-3) & x_2(t-j-3) & d(t-j-3) & s(t-j-4) & s(t-j-3) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1(t-j-q) & x_2(t-j-q) & d(t-j-q) & s(t-j-q-1) & s(t-j-q) \end{pmatrix} \quad (3)$$

where  $j$  and  $q$  are predetermined parameters dividing the training and testing datasets. The dataset contained 379 data points per variable; 85% of the data were allocated to the training dataset, and 15% were allocated to the testing dataset. The training dataset was further divided into the strictly training subset (65%) and the validation subset (20%). The neural network was trained with the training dataset ( $T_2$ ).

For clarity purposes, we have added an example of the inputs and outputs of the neural network, see Table 2, and a graphical representation in Figure 2.

**Table 2.** Example of the inputs and outputs of the neural network.

Inputs				Output
$x(t - 1)$	$y(t - 1)$	$d(t - 1)$	$s(t - 1)$	$s(t)$
2125.9	20.64	103	0.53	0.53
495.36	299.28	47,967	0.53	0.59
464.4	30.96	64	0.59	0.61
2786.4	402.48	37,119	0.61	0.62
2972.2	474.72	52,861	0.62	0.61
2414.9	516	20,909	0.61	0.60
2538.7	30.96	40,355	0.60	0.55
1465.4	350.88	16,547	0.55	0.52
1052.6	381.84	19,264	0.52	0.57
:	:	:	:	:



**Figure 2.** Graphical description of inputs and outputs in the ANN, where  $x(t - 1)$ ,  $y(t - 1)$ ,  $d(t - 1)$  and  $s(t - 1)$  are the inputs and  $s(t)$  is the output.

### 2.1. Data Mappings

It is important to have a sense of the mapping of the inputs and outputs. In this section, we analyze the mapping of each input ( $x(t - 1)$ ,  $y(t - 1)$ ,  $d(t - 1)$ ) with the output ( $s(t - 1)$ ). Hence, we can consider four mappings as shown in Table 3.

**Table 3.** Mappings.

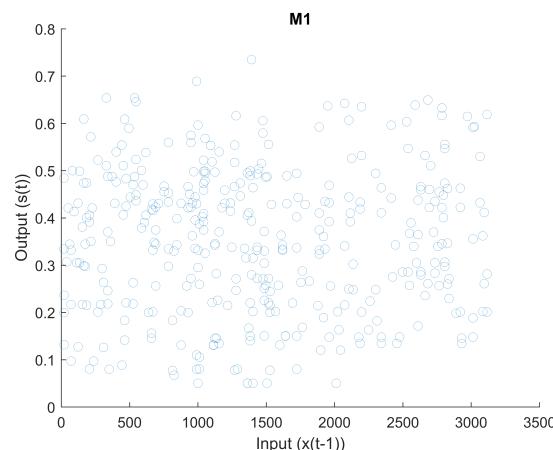
Mapping	Input	Output
Mapping 1 (M1)	$x(t - 1)$	$s(t)$
Mapping 2 (M2)	$y(t - 1)$	$s(t)$
Mapping 3 (M3)	$d(t - 1)$	$s(t)$
Mapping 4 (M4)	$s(t - 1)$	$s(t)$

#### 2.1.1. Mapping 1

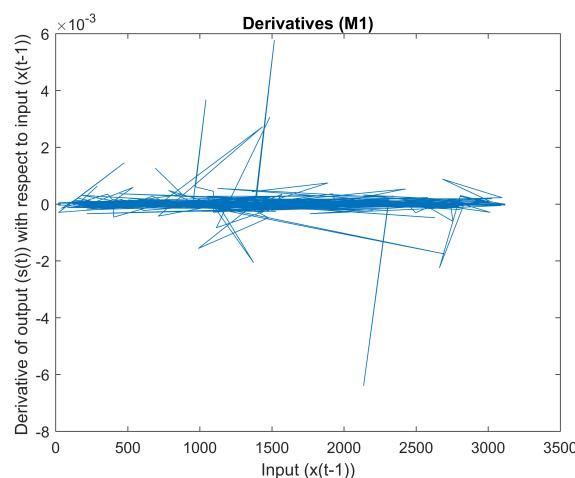
Figure 3 is a scatter plot of input  $x(t - 1)$  vs. output  $s(t)$ . The derivative of input  $x(t - 1)$  with regard to output  $s(t)$  can be seen in Figure 4. There appear to be sudden changes in the derivative. The curvature of a mapping of an input “in” and an output “o” can be estimated using Equation (4), where  $ss$  is the arch of the length of the analyzed curve. In this notation,  $d$  is the first derivative, and  $d^2$  is the second derivative. Also, as an example, the input for M1 is  $x(t - 1)$ , and the output is  $s(t)$ . This is done for each mapping

shown in Table 3. The curvature of this mapping can be seen in Figure 5. The curvature does not appear to be smooth.

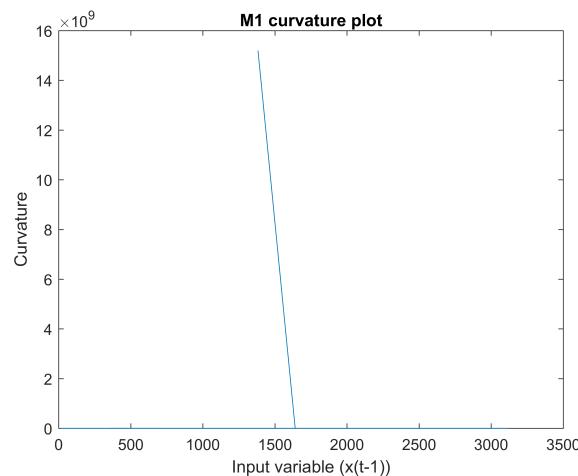
$$\text{Curvature} = \frac{\left| \left( \frac{d\ln}{ds} \right) \left( \frac{d^2 o}{ds^2} \right) - \left( \frac{do}{ds} \right) \left( \frac{d^2 \ln}{ds^2} \right) \right|}{\left( \left( \frac{d\ln}{ds} \right)^2 + \left( \frac{do}{ds} \right)^2 \right)^{3/2}} \quad (4)$$



**Figure 3.** Scatter plot (M1) of input  $s(t - 1)$  vs. output  $s(t)$ .



**Figure 4.** Derivative of input  $x(t - 1)$  with regard to output  $s(t)$ .



**Figure 5.** Curvature of M1.

A lineal model, as described in Equation (5), is another technique to analyze this mapping. The obtained adjusted  $R^2$  in this model is  $-0.002522$ ; see Table 4.

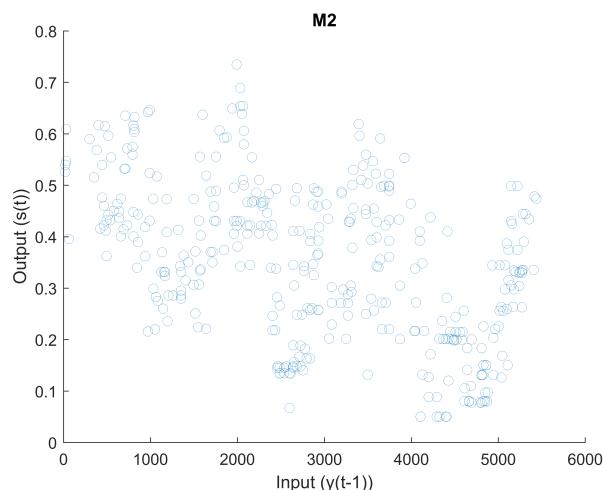
$$s(t) = \beta_0 + \beta_1 x(t-1) + \epsilon \quad (5)$$

**Table 4.** Linear regression. Equation (5).

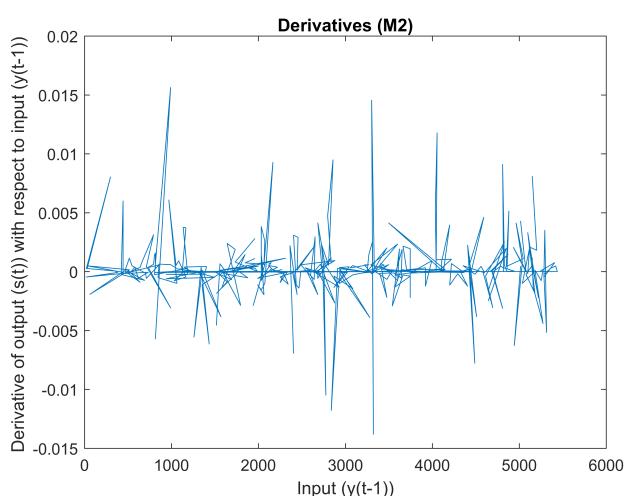
Variable	Estimate	$\sigma$	$t$	$p$
Intercept	$3.486 \times 10^{-1}$	$1.497 \times 10^{-2}$	23.279	$<2 \times 10^{-16}$
$x(t-1)$	$2.123 \times 10^{-6}$	$8.930 \times 10^{-6}$	0.238	0.812
Multiple $R^2$	0.0001512			
Adjusted $R^2$	-0.002522			
RSE	0.1514			
$p$	0.8122			

### 2.1.2. Mapping 2

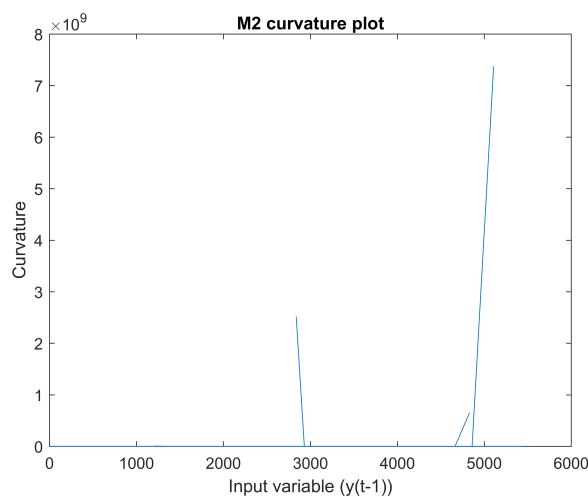
Figure 6 is a scatter plot of input  $y(t-1)$  vs. output  $s(t)$ . The derivative of input  $y(t-1)$  with regard to output  $s(t)$  can be seen in Figure 7. There appear to be sudden changes in the derivative. The curvature of this mapping can be seen in Figure 8. The curvature does not appear to be smooth.



**Figure 6.** Scatter plot (M2) of input  $y(t-1)$  vs. output  $s(t)$ .



**Figure 7.** Derivative of input  $y(t-1)$  with regard to output  $s(t)$ .



**Figure 8.** Curvature of M2.

A lineal model, as described in Equation (6), is another technique to analyze this mapping. The obtained adjusted  $R^2$  in this model is 0.2277; see Table 5. Please notice that this time series, as will be shown in the next section, is not stationary. Hence, a direct implementation of a linear model is not appropriate.

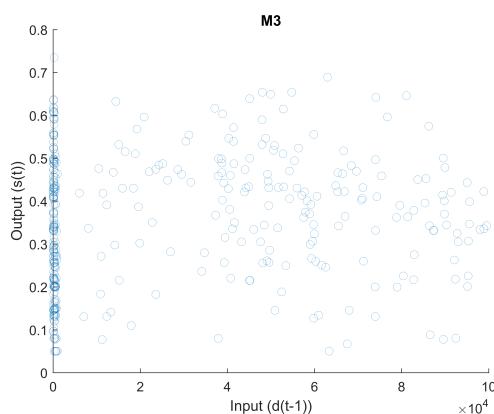
$$s(t) = \beta_0 + \beta_1 y(t-1) + \epsilon \quad (6)$$

**Table 5.** Linear regression. Equation (8).

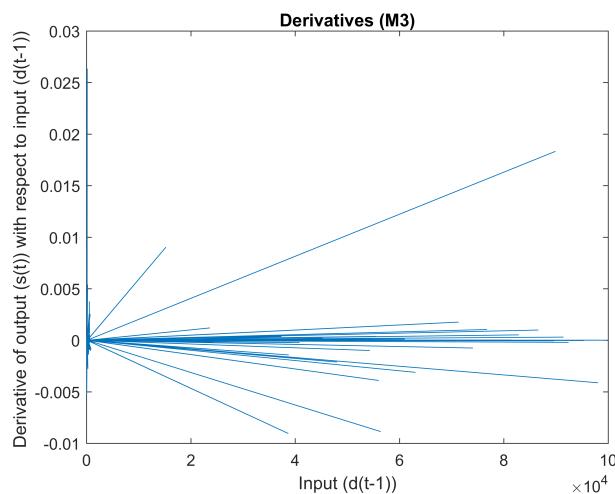
Variable	Estimate	$\sigma$	$t$	$p$
Intercept	$4.926 \times 10^{-1}$	$1.500 \times 10^{-2}$	32.84	$<2 \times 10^{-16}$
$s(t-1)$	$-4.865 \times 10^{-5}$	$4.606 \times 10^{-6}$	-10.56	$<2 \times 10^{-16}$
Multiple $R^2$	0.2298			
Adjusted $R^2$	0.2277			
RSE	0.1329			
$p$	$<2.2 \times 10^{-16}$			

### 2.1.3. Mapping 3

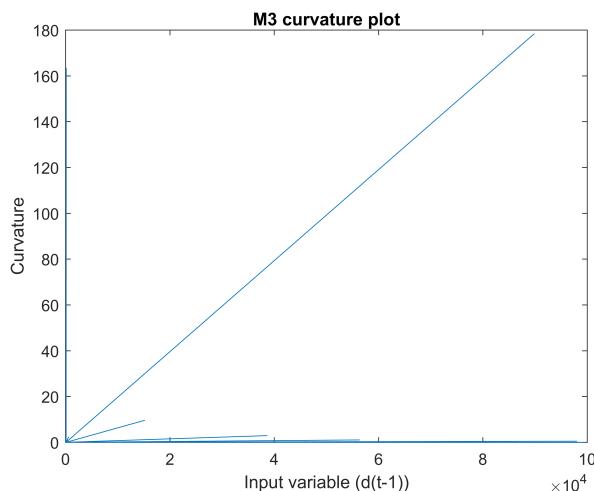
Figure 9 is a scatter plot of input  $d(t-1)$  vs. output  $s(t)$ . The derivative of input  $d(t-1)$  with regard to output  $s(t)$  can be seen in Figure 10. There appear to be sudden changes in the derivative. The curvature of this mapping can be seen in Figure 11. The curvature does not appear to be smooth.



**Figure 9.** Scatter plot (M3) of input  $d(t-1)$  vs. output  $s(t)$ .



**Figure 10.** Derivative of input  $d(t - 1)$  with regard to output  $s(t)$ .



**Figure 11.** Curvature of M3.

A lineal model, as described in Equation (7), is another technique to analyze this mapping. The obtained adjusted  $R^2$  in this model is 0.01819; see Table 6.

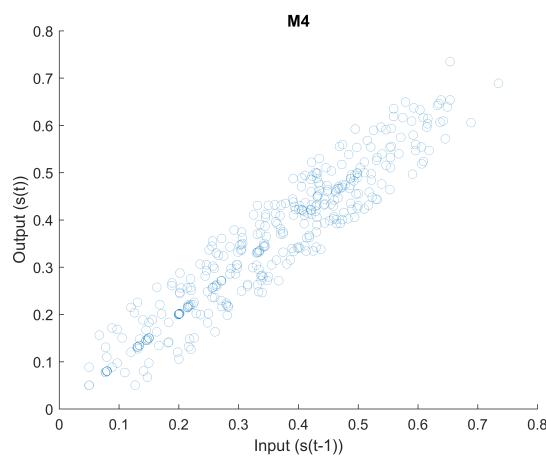
$$s(t) = \beta_0 + \beta_1 d(t - 1) + \epsilon \quad (7)$$

**Table 6.** Linear regression. Equation (7).

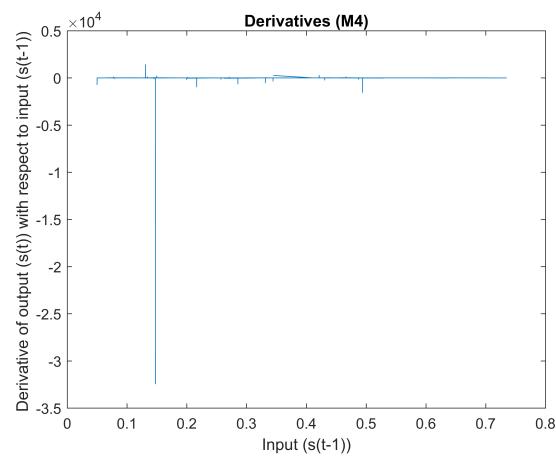
Variable	Estimate	$\sigma$	$t$	$p$
Intercept	$3.332 \times 10^{-1}$	$1.013 \times 10^{-2}$	32.894	$<2 \times 10^{-16}$
$d(t - 1)$	$6.816 \times 10^{-7}$	$2.418 \times 10^{-7}$	2.819	0.00507
Multiple $R^2$	0.02081			
Adjusted $R^2$	0.01819			
RSE	0.1498			
$p$	<0.005068			

#### 2.1.4. Mapping 4

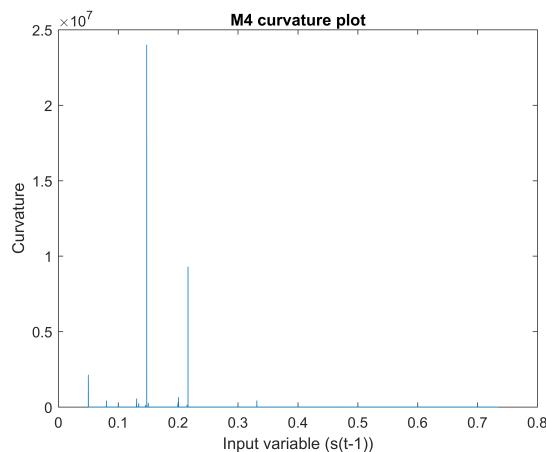
Figure 12 is a scatter plot of input  $s(t - 1)$  vs. output  $s(t)$ . The derivative of input  $s(t - 1)$  with regard to output  $s(t)$  can be seen in Figure 13. The curvature of this mapping can be seen in Figure 14.



**Figure 12.** Scatter plot (M4) of input  $s(t - 1)$  vs. output  $s(t)$ .



**Figure 13.** Derivative of input  $s(t - 1)$  with regard to output  $s(t)$ .



**Figure 14.** Curvature of M4.

A lineal model, as described in Equation (8), is another technique to analyze this mapping. The obtained adjusted  $R^2$  in this model is 0.9018; see Table 7. Please notice that this time series, as will be shown in the next section, is not stationary. Hence, a direct implementation of a linear model is not appropriate.

$$s(t) = \beta_0 + \beta_1 s(t - 1) + \epsilon \quad (8)$$

**Table 7.** Linear regression. Equation (8).

Variable	Estimate	$\sigma$	t	p
Intercept	0.018002	0.006187	2.91	0.00383
$s(t - 1)$	0.946893	0.016131	58.70	$<2 \times 10^{-16}$
Multiple $R^2$	0.9021			
Adjusted $R^2$	0.9018			
RSE	0.04738			
p	$<2.2 \times 10^{-16}$			

## 2.2. Linear Approach

The variable that we are trying to forecast ( $s(t)$ ) is not stationary. This was formally tested with an augmented Dickey–Fuller test; see Table 8.

**Table 8.** The p value of augmented Dicker–Fuller test (stationary test).

Variable	$x(t - 1)$	$y(t - 1)$	$d(t - 1)$	$s(t - 1)$	$s(t)$
p value	<0.0010	0.4344	<0.0010	0.1375	0.1389

Before using an ANN, it is advisable to use a lineal model. In order to do this, the three non-stationary variables ( $y(t - 1), s(t - 1)$  and  $s(t)$ ) were made stationary by taking the differences: for example, using  $(y(t - 1) - y(t - 2))$  instead of  $y(t - 1)$ . These variable transformations to make them stationary will be denoted with the operator “ $\Delta$ ”. Hence, in the previous example  $(y(t - 1) - y(t - 2))$ , the stationary variable will be denoted as  $\Delta y(t - 1)$ . The same notation is used for the speed. In this case,  $(s(t - 1) - s(t - 2))$  is denoted by  $\Delta s(t - 1)$ . The target variable that we are forecasting  $s(t)$  follows the same notation  $\Delta s(t)$  (denoting  $(s(t) - s(t - 1))$ ).

Having a status of being non-stationary was once more tested using an augmented Dicker–Fuller test. As can be seen in Table 9, now all the variables are stationary.

**Table 9.** The p value of augmented Dicker–Fuller test (stationary test).

Variable	$x(t - 1)$	$\Delta y(t - 1)$	$d(t - 1)$	$\Delta s(t - 1)$	$\Delta s(t)$
p value	<0.0010	<0.0010	<0.0010	<0.0010	<0.0010

A linear model of the type shown in Equation (9) can be tested.

$$\Delta s(t) = \beta_0 + \beta_1 x(t - 1) + \beta_2 \Delta y(t - 1) + \beta_3 d(t - 1) + \beta_4 \Delta s(t - 1) + \epsilon \quad (9)$$

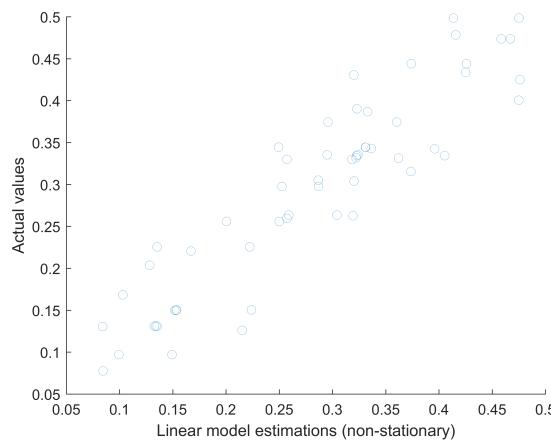
The results of the linear regression are as follows (Table 10). The multiple  $R^2$  and the adjusted  $R^2$  are 0.020230 and 0.009641, respectively.

**Table 10.** Linear regression of the above-mentioned model (Equation (9)).

Variable	Estimate	$\sigma$	t	p
Intercept	$-5.755 \times 10^{-3}$	$5.502 \times 10^{-3}$	-1.046	0.2963
$x(t - 1)$	$2.912 \times 10^{-6}$	$2.880 \times 10^{-6}$	1.011	0.3127
$\Delta y(t - 1)$	$4.090 \times 10^{-6}$	$8.055 \times 10^{-6}$	0.508	0.6119
$d(t - 1)$	$4.132 \times 10^{-8}$	$7.872 \times 10^{-8}$	0.525	0.5999
$s(t - 1)$	$1.317 \times 10^{-1}$	$5.207 \times 10^{-2}$	2.530	0.0118
Multiple $R^2$	0.020230			
Adjusted $R^2$	0.009641			
RSE	0.04815			
p	0.1081			

### Lineal Model with Non-Stationary Data

If the lineal model is used with non-stationary data, then it is important for comparison purposes to use a similar approach to that in the ANN model that will be illustrated in the following section. For comparison purposes, a lineal model can be built, and then its generalization capabilities can be tested with 15% of the data (not used to build the linear regression). This approach is similar to the one followed in the next section. In this case, the obtained  $R^2$  (test data) is 0.8371 and the  $RMSE$  is 0.0483. A scatter plot of the prediction of the linear model (using non-stationary data) and the actual values (test set) can be seen in Figure 15.



**Figure 15.** Scatter plot of linear regression model (using non-stationary data) and the actual values.

### 2.3. ANN Approach

Authors such as Kim et al. [55] have successfully applied ANNs to non-stationary data. Several other papers, such as Cheng et al. [56], Ghazali et al [57] and Marseguerra et al. [58], also illustrate the application of ANN to non-stationary data. It is necessary to carry out an ANN parameter optimization task to try to make the estimated velocity  $\hat{s}_i$  and the actual (measured) velocity  $s_i$  as similar as possible. There are several metrics ( $\zeta$ ) that can be used for this type of task, such as  $R^2$  or the  $RMSE$ . In this case, we selected  $R^2$  as the metric to measure the similarity ( $\zeta$ ) of these values, but we also estimated the  $RMSE$ . These metrics are impacted by the architecture of the neural network, such as the number of neurons ( $n$ ), the number of layers ( $m$ ) and the training algorithm ( $a$ ) used. This is an optimization problem that can be solved by using the Algorithm 1 shown below.

---

#### Algorithm 1 An optimization problem

---

**Require:**  $n_i$ ,  $m_i$  and  $a_i$ .

**Ensure:**  $\hat{s}(t)$ .

- 1: Compute  $\hat{s}(t)$  for each configuration
- 2: Compare  $\hat{s}(t)$  with  $s(t)$  for each configuration
- 3: Solve the following problem:

$$\begin{aligned} & \max_{n_1, n_2, \dots, n_{max}} \zeta(n_l, m_l, a_l) \\ & \max_{m_1, m_2, \dots, m_{max}} \\ & \max_{a_1, a_2, \dots, a_{max}} \\ & \text{s.t. } 1 \leq n_i < n_{max}, \\ & \quad 1 \leq m_i < m_{max}, \\ & \quad 1 \leq a_i < a_{max}, \end{aligned}$$

- 4: Obtain  $n_l$ ,  $m_l$  and  $a_l$
-

It should be noted that in this case, because the selected similarity metric ( $\zeta$ ) is the  $R^2$  comparing  $\hat{s}(t)$  with  $s(t)$ , we have a maximization type of optimization problem. If we would have chosen the  $RMSE$  as the metric, we would have had a minimization problem. Additionally, it also should be explicitly mentioned that we are solving this problem using a grid approach. For each network configuration  $ANN(n_i, m_i, a_i)$ , there is an associated training process with a related training time, which is an important consideration. It will be shown that there is a non-linear relationship between the number of neurons ( $n$ ) and the time required to train such a neural network ( $ANN(n_i, m_i, a_i)$ ).

The use of ANNs also has its disadvantages. The initial weights in an artificial neural network are created randomly. This is an inherent part of the process. During the training phase, those weights are then modified to make the forecast as close as possible to the actual output (in this case, the velocity ( $s$ )). Because the weights are initially randomly generated, each ANN, even when having identical architectures (same number of neurons, layers and training algorithm), is likely to generate slightly different outputs. In order to reduce the impact of this initial random generation of the weights, each configuration is trained  $\gamma$  times. The similarity values are the mean values of these  $\gamma$  simulations for each ANN configuration. The  $R^2$  and the  $RMSE$  reported are those obtained with the testing dataset. The testing dataset was not used during the training phase. After the ANN was trained, the testing data were used as input (no further training), the similarity metrics, such as the  $R^2$  and the  $RMSE$ , were calculated, and the mean level was reported. We used a value of  $\gamma$  equal to 100, i.e., each configuration ( $ANN(n_i, m_i, a_i)$ ) was simulated 100 times. The range for the number of neurons was  $10 \leq n \leq 100$  in increments of one neuron at a time. The hidden layers included neurons with a *tansig* transfer function as described in Equation (10). The output layer consists of one linear transfer function.

Configurations with one to three layers were tested. Twelve different training algorithms were used (see Table 11). Hence, 3276 different configurations were simulated (100 times each), for a total of 327,600 simulations.

$$tansig(z) = \frac{2}{1 + e^{-2z}} - 1 \quad (10)$$

**Table 11.** Twelve training algorithms were used.

Training Algorithm	Training Algorithm
BFG	GDM
BR	GDX
CGB	LM
CGF	OSS
CGP	RP
GDA	SCG

Following this approach, the top 10 models were identified. After the model identification section, a robustness analysis was carried out. The robustness analysis consisted in simulating each of the ten top models identified in the previous section 1000 times. The batch size was 30.

#### 2.4. Impact of Noise

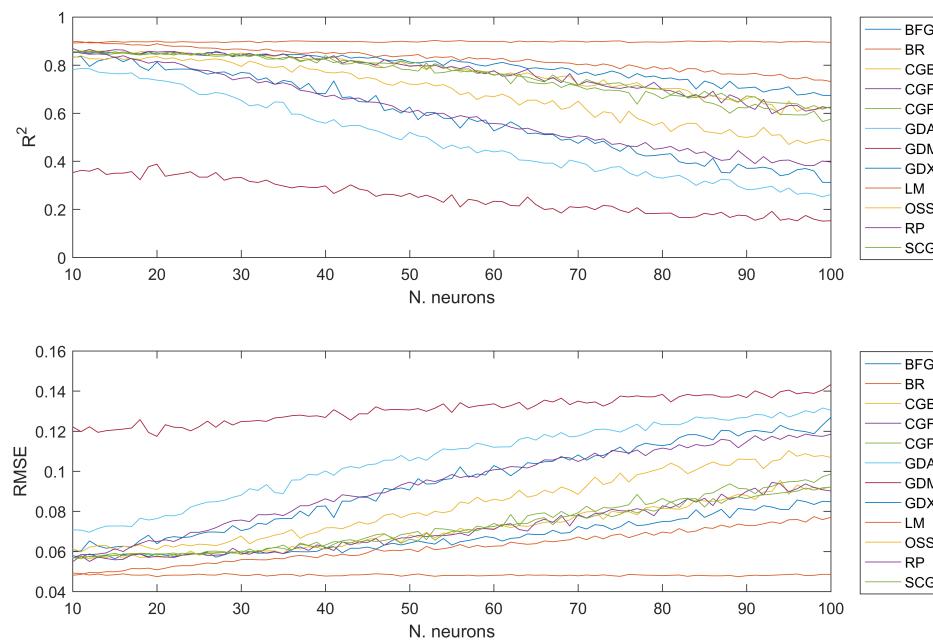
We studied the goodness of the fit of the model when additional noise was added to the signal. Given the experimental challenges in achieving these measurements (at these scales), it is important to have an understanding of how well the model works when there is noise. This was tested by adding to the signal normally distributed noise ( $N(0, \sigma)$ ) with mean zero ( $\mu = 0$ ) and  $\sigma$  proportional to  $\sigma_{signal}$  of the original signal, according to Equation (11).

$$\sigma = \alpha \cdot \sigma_{signal} \quad (11)$$

where  $\alpha$  is a parameter  $\alpha = \{0.1, 0.2, \dots, 1\}$ . Several noise levels with zero means and increasing standard deviations were simulated. The standard deviations of the noise were increased from 1% to 100% of the standard deviation of the original signal in 1% increments. For each of these levels of noise, the model was simulated 100 times, and then, its goodness of fit was estimated using the  $R^2$  and  $RMSE$  metrics. This process was repeated for all top 10 configurations obtained in the previous section.

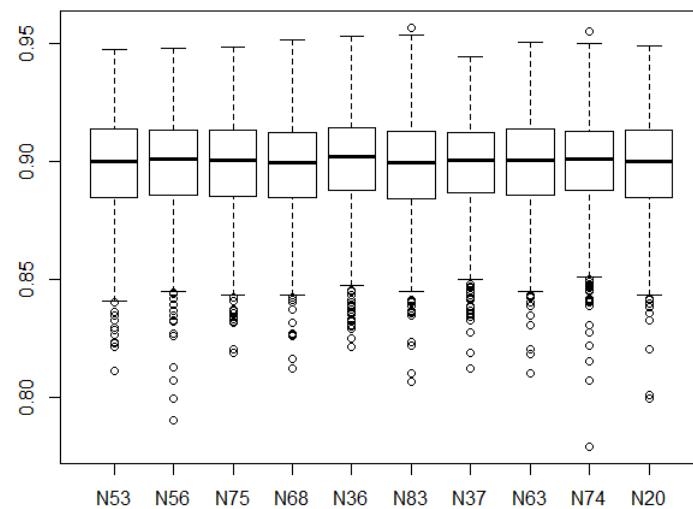
### 3. Results

As shown in Figure 16 the selection of the training algorithm is important. All top ten models (highest  $R^2$ ) use a BR (Bayesian regression) training algorithm and have only one hidden layer. In this table, both the  $R^2$  and the  $RMSE$  are shown as a function on the number of neurons. As previously mentioned, the results reported are the mean values after 100 simulations for each configuration. Other models, such as GDM (see the abbreviations table at the end of the paper for a key to the names of the algorithms), generate much less accurate models. The top 10 models obtained are  $\{N53, N56, N75, N68, N36, N83, N37, N63, N74, N20\}$ . As an example, in this notation  $N53$ , represents a model using 53 neurons in one hidden layer. There is no mention of the number of layers or the training algorithm in the notation because among the top 10 models, they all use the BR training algorithm and all have one layer. The best model obtained was  $N53$ , with a mean  $R^2$  of 0.9029 and a mean  $RMSE$  of 0.0476. A table with the results of all these 10 models can be seen in Table A1 in the Appendix A. The results for the most accurate models for all the other configurations (other training algorithms as well as models with two or three layers) can be seen in Table A2 in the Appendix A. All the reported data refer to the testing dataset, which was not used during the training phase and contains approximately 15% of the total data.

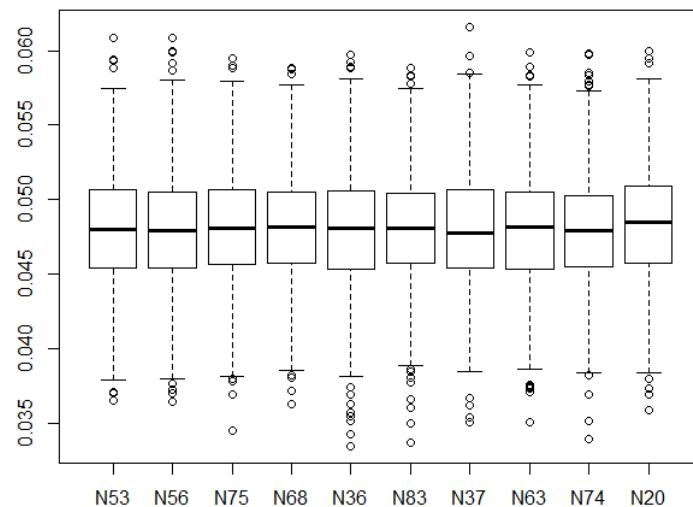


**Figure 16.** Neural network precision for different training algorithms.

After the best 10 models were identified, it was necessary to carry out a robustness analysis. Each of these 10 top configurations were simulated 1000 times. The resulting  $R^2$  and  $RMSE^2$  comparing the predictions of the testing dataset with the actual data were estimated for each simulation. The results (testing dataset) can be seen in Figures 17 and 18. After the robustness analysis, the best model was N36 with a mean and a median  $R^2$  of, respectively, 0.8994 and 0.9017. The mean and median value for the  $RMSE$  for this model were 0.0480 and 0.0481, respectively. For the same ANN configuration, the validation dataset obtained an  $R^2$  and  $RMSE$  of 0.9150 and 0.0433, respectively. The results for all the 10 models (test dataset) can be seen in Table A3 in Appendix A.



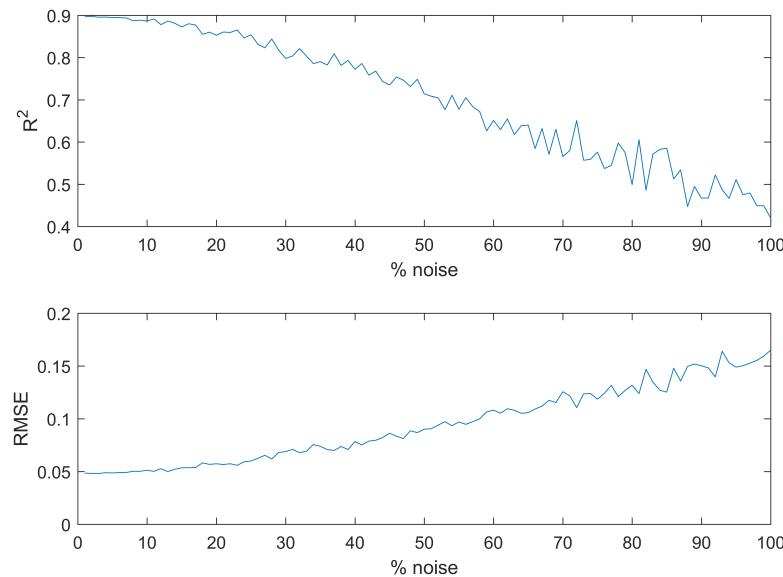
**Figure 17.** Robustness testing for top 10 models ( $R^2$ ).



**Figure 18.** Robustness testing for top 10 models (RMSE).

It also should be mentioned that the training time becomes an important factor to take into consideration. As the number of neurons increases, the related computational requirements increase. In the case of the robustness analysis (1000 simulations per configuration), the same non-linear relationship was observed (Figure A1 in Appendix A). In Figure A1, it can be seen that the related training time increases (as we passed from 100 to 1000 simulations per configuration), but the shape of the curve remains similar, with an exponential increase in time as the number of neurons increases. The calculations were carried out in MATLAB on a laptop with an 11th Generation Intel i7-11700 @ 2.5 GHZ and 16 GB RAM.

As can be seen in Figure 19, when noise is added, the forecast velocity becomes less accurate. The bigger the  $\alpha$ , the less accurate the forecast velocity becomes. As previously mentioned, the parameter  $\alpha$  relates (according to Equation (11)) the standard deviation ( $\sigma$ ) of the added noise with the standard deviation of the original signal ( $\sigma_{signal}$ ). A value of  $\alpha = 0.01$  is 1% of the signal's standard deviation, and  $\alpha = 1$  is 100% of the signal's standard deviation. The  $R^2$  value comparing the forecast velocity and the actual velocity remains above 0.8 for  $\alpha \leq 0.33$  (33%).



**Figure 19.** Impact on noise (as % of signal) in the  $R^2$  and RMSE metrics. Model with 1 layer, 36 neurons and BR training algorithm.

#### 4. Discussion

When analyzing the results, it should be taken into account that measuring blood microflows remains, from an experimental point of view, a challenging tasks. Not only are the scales involved small, but there are factors, such as the deformation of the red blood cells as they squeeze through the capillaries, that make the flow rather complex to model. There are also many different types of cells in the blood, further adding complexity to the model. Hence, there can be some inaccuracies in the measurements. In this regard, the model needs not only to be reasonably accurate but also robust and not too sensitive to noise.

The proposed ANN-based estimations (N53 configuration) obtained a mean  $R^2$  of 0.9029 and a mean RMSE of 0.0476 in its estimations of blood velocity in capillaries (ranging from approximately 25 to 500  $\mu\text{m}$ ). For comparison purposes, the configuration with the lowest mean  $R^2$  (0.512) was an ANN with two hidden layers and a GDM training algorithm (the results for all the configurations can be found in Table A2 in Appendix A). A total of 327,600 simulations were carried out in order to select an appropriate ANN architecture. The proposed grid approach selected a neural network with one hidden layer, 36 neurons and Bayesian regression (BR) as the training algorithm. Better performance of BR against other training algorithms has been observed in other applications [59]. Deeper ANNs (with two or three layers) were not able to improve the forecasts generated by a one-layer structure. This highlights the need to select the appropriate network for the appropriate task, as blindly increasing the number of layers does not translate into a better model. This is likely due to overfitting. The importance of selecting the appropriate parameters in an ANN has been mentioned by several authors, such as Bernardos and Vosniakos [60]. The selected model generated accurate forecasts when tested during the robustness analysis. When comparing these forecasts with the actual (measured) values of the velocity, the mean and median  $R^2$  were 0.8994 and 0.9017, respectively, while the mean and median value for the RMSE were 0.0480 and 0.0481, respectively. These values were obtained by doing 1000 simulations with this final configuration. From a robustness analysis point of view it is always possible to do more simulations (in this case, we did 1000 per selected model). However, given the similarity of the results for the top 10 selected models, it would appear that 1000 iterations is likely a reasonable amount, particularly when considering the computational cost when substantially increasing the number of simulations per configuration. The BR training algorithm appears to consistently outperform all the other training algorithms tested (12 in total). Similarly, there were some

training algorithms that consistently underperformed, such as GDM, GDA and the GDX. The best results using two hidden layers were obtained for the BR training algorithm, with a mean  $R^2$  value of 0.8992. This value was obtained with 100 simulations; using the same number of simulations, the one-layer model obtained a mean  $R^2$  of 0.9029 (which slightly decreased to a value of 0.8994 after the robustness test using 1000 simulations). The worst result using a two-layer configuration was obtained using the GDM algorithm ( $R^2$  of 0.512). When using three layers, the best results were obtained with the Levenberg–Marquardt training algorithm, with a mean  $R^2$  of 0.860. Similarly to the two-layer case, when using three layers, the worst results were obtained when applying the GDM training algorithm ( $R^2$  of 0.510). The top 10 models obtained using the grid approach generated comparable results. Even after the robustness analysis, these 10 models seemed to perform roughly in a similar fashion (see Figures 17 and 18). These ten models generated forecasts that are statistically significantly better than those obtained with different ANN architectures.

It was observed that the computational time required to train the network increased non-linearly with the number of neurons. This was observed during the model selection phase as well as during the robustness analysis. While the computational times required in these two sections were different (in one case, there were 100 simulations per ANN configuration, and in the other case, there were 1000) the shape of the curve representing computational time vs. the number of neurons was similar (exponential curve). Given this observation and the fact that models with more neurons did not necessarily perform better, it highlights the need for adequate ANN parameter selection. Computational time is an important factor, with some training processes requiring more than a day for a single configuration. This limits the total number of testable configurations. These computational times were obtained using standard laptops, and hence, with more computational power, these training times could be reduced. Another additional consideration is that the exponential trend was observed within a particular range of neurons. It was not tested with more than 100 neurons per layer.

It was also interesting to see the performance of the model when introducing noise in the signal. The introduced noise was normally distributed with zero mean and a standard distribution proportional to the original standard distribution of the signal (in 1% increments from 1% to 100%). The model managed to generate accurate forecasts, with a mean  $R^2$  above 0.8, until reaching 33% of the standard deviation of the original signal. This type of analysis, adding some artificial noise to the signal, was carried out in order to better understand how well the model reacts to noise. This is important, as measurements in the analyzed scale are experimentally difficult to obtain and could have, in principle, substantial error that might be difficult to estimate. The model seems adequate for the above-mentioned levels of noise (less than 33% of the standard deviation of the original signal). If there are indications that the noise is bigger than that, then the proposed model might not be accurate enough. The precision of the presented model is high (with an  $R^2$  of 0.8992). The usefulness of this model to other researchers will depend on their required precision. While the scale of the analysis is likely enough for many pharmaceutical applications, in some instances, there might be a need for higher precision. This would require not only further improvements in the modeling but also more precise experimental data, which can be challenging to obtain.

Kloosterman et al. [54] mentioned that a vascular network approach can be used to investigate vascular development. They also concluded that vascular remodeling could be observed, and that the changes in network parameters, such as the velocity, could be related to structural changes. The authors also mentioned that some networks became denser, but other networks became less dense.

## 5. Conclusions

Artificial neural networks appear to be a reasonable option for modeling blood microflows and generate accurate forecasts (with an  $R^2$  of 0.8992). This type of forecast might be useful in several different applications, such as drug delivery and diffusion analysis as well as changes in microflows, which have been an area of recent research interest. The analysis shows that a key step in the model selection process is choosing the appropriate parameters for the ANN, such as the number of neurons, the number of layers and the training algorithm. Another major factor to take into account is the computational time required to train the network. This is particularly important, as there is a large number of potential configurations. It is also important to assess the robustness of the model as well as how sensitive it is to noise. This is important given the experimental complexity of making these measurements and the potential for some measurement uncertainties. It was shown that the model can handle additional noise relatively well. It also should be noted that several different configurations generated accurate forecasts, but that all the top models used the same training algorithm: Bayesian regression.

**Author Contributions:** Conceptualization, G.A.P. and J.V.C.P.; methodology, G.A.P.; software, G.A.P.; validation, G.A.P. and J.V.C.P.; formal analysis, G.A.P. and J.V.C.P.; investigation, G.A.P. and J.V.C.P.; resources, G.A.P. and J.V.C.P.; data curation, G.A.P.; writing—original draft preparation, G.A.P.; writing—review and editing, G.A.P. and J.V.C.P.; visualization, G.A.P. and J.V.C.P.; supervision, G.A.P. and J.V.C.P.; project administration, G.A.P. and J.V.C.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** The experimental data used in this article were obtained from Kloosterman et al. [54].

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artificial Neural Network
BFG	BFG Quasi Newton
BR	Bayesian Regularization
CGB	Conjugate Gradient with Powell–Beale Restarts
CGF	Conjugate Gradient Fletcher–Powell
CGP	Conjugate Gradient Polak–Ribiere
CS	Cardiogenic Shock
GDA	Backpropagation Gradient Descent with Adaptive Learning
GDM	Backpropagation Gradient Descent with Momentum
GDX	Backpropagation with Variable Learning
LDF	Laser Doppler Flowmetry
LM	Levenberg–Marquardt
OSS	Secant One-Step
RP	Resilient Backpropagation
SCG	Scale Conjugate Gradient

## Appendix A

The numerical values of the mean  $R^2$  and RMSE values in the top 10 models can be found in Table A1.

**Table A1.** Top 10 models (1-layer). All 10 top models used the BR training algorithm.

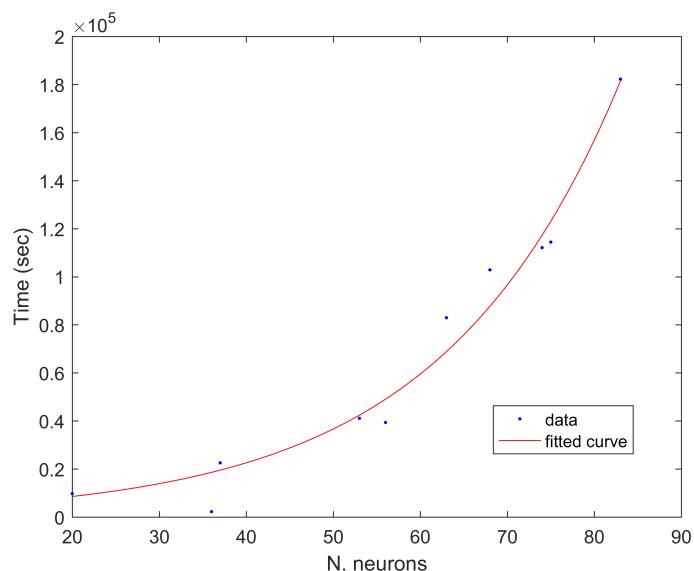
N. Neurons	Algorithm	Mean R-Squared	Mean RMSE
53	BR	0.903	0.048
56	BR	0.902	0.048
75	BR	0.901	0.048
68	BR	0.900	0.048
36	BR	0.900	0.048
83	BR	0.900	0.048
37	BR	0.900	0.048
63	BR	0.900	0.048
74	BR	0.900	0.048
20	BR	0.900	0.048

**Table A2.** Top models per network architecture with 2 or 3 layers.

N. Layers	Algorithm	Mean R-Squared
2	BFG	0.843
2	BR	0.899
2	CGB	0.850
2	CGF	0.847
2	CGP	0.845
2	GDA	0.752
2	GDM	0.512
2	GDX	0.795
2	LM	0.866
2	OSS	0.821
2	RP	0.843
2	SCG	0.837
3	BFG	0.837
3	BR	0.675
3	CGB	0.836
3	CGF	0.838
3	CGP	0.829
3	GDA	0.735
3	GDM	0.510
3	GDX	0.733
3	LM	0.860
3	OSS	0.819
3	RP	0.834
3	SCG	0.824

**Table A3.** Top 10 models (1-layer) after robustness analysis. All 10 top models used the BR training algorithm.

Neurons	Algorithm	Mean $R^2$	Median $R^2$	Mean RMSE	Median RMSE
53	BR	0.898	0.900	0.048	0.048
56	BR	0.898	0.901	0.048	0.048
75	BR	0.899	0.900	0.048	0.048
68	BR	0.898	0.900	0.048	0.048
36	BR	0.899	0.902	0.048	0.048
83	BR	0.897	0.899	0.048	0.048
37	BR	0.898	0.900	0.048	0.048
63	BR	0.899	0.900	0.048	0.048
74	BR	0.899	0.901	0.048	0.048
20	BR	0.898	0.900	0.048	0.048



**Figure A1.** Number of neurons vs. training time (robustness analysis).

## References

- Karagounis, V.A.; Pries, A.R. Micro Flows in the Cardiopulmonary System: A Surgical Perspective. In *Micro and Nano Flow Systems for Bioanalysis*; Springer: New York, NY, USA, 2012; pp. 69–76. [CrossRef]
- Liu, X.; Gao, Q.; Zhang, Y.; Li, Y.; Li, B. In vivo optofluidic switch for controlling blood microflow. *Adv. Sci.* **2020**, *7*, 2001414. [CrossRef] [PubMed]
- Jie, M.; Li, H.; Lin, L.; Zhang, J.; Lin, J. Integrated microfluidic system for cell co-culture and simulation of drug metabolism. *RSC Adv.* **2016**, *6*, 54564–54572. [CrossRef]
- Hemmilä, S.; Ruponen, M.; Toropainen, E.; Tengvall-Unadike, U.; Urtti, A.; Kallio, P. Microflow-Based Device for In Vitro and Ex Vivo Drug Permeability Studies. *SLAS Technol. Transl. Life Sci. Innov.* **2020**, *25*, 455–462. [CrossRef] [PubMed]
- Le, A.V.; Fenech, M. Image-Based Experimental Measurement Techniques to Characterize Velocity Fields in Blood Microflows. *Front. Physiol.* **2022**, *13*, 886675. [CrossRef] [PubMed]
- Abay, A.; Recktenwald, S.M.; John, T.; Kaestner, L.; Wagner, C. Cross-sectional focusing of red blood cells in a constricted microfluidic channel. *Soft Matter* **2020**, *16*, 534–543. [CrossRef] [PubMed]
- Agrawal, R.; Balne, P.K.; Tun, S.B.B.; Wey, Y.S.; Khandelwal, N.; Barathi, V.A. Fluorescent dye labeling of erythrocytes and leukocytes for studying the flow dynamics in mouse retinal circulation. *JoVE* **2017**, *125*, e55495. [CrossRef]
- Bishop, J.J.; Popel, A.S.; Intaglietta, M.; Johnson, P.C. Effects of erythrocyte aggregation and venous network geometry on red blood cell axial migration. *Am. J. Physiol.-Heart Circ. Physiol.* **2001**, *281*, H939–H950. [CrossRef] [PubMed]
- Morini, G.L.; Yang, Y.; Chalabi, H.; Lorenzini, M. A critical review of the measurement techniques for the analysis of gas microflows through microchannels. *Exp. Therm. Fluid Sci.* **2011**, *35*, 849–865. [CrossRef]
- Lingadahalli Kotreshappa, S.; Nayak, C.G.; Krishnan Venkata, S. A review on the role of microflow parameter measurements for microfluidics applications. *Systems* **2023**, *11*, 113. [CrossRef]
- Batista, E.; Godinho, I.; Martins, R.F.; Mendes, R.; Robarts, J. Development of an experimental setup for microflow measurement using interferometry. *Flow Meas. Instrum.* **2020**, *75*, 101789. [CrossRef]
- Mizeva, I.; Makovik, I.; Dunaev, A.; Krupatkin, A.; Meglinski, I. Analysis of skin blood microflow oscillations in patients with rheumatic diseases. *J. Biomed. Opt.* **2017**, *22*, 70501. [CrossRef] [PubMed]
- Gourley, M.; Miller, F.W. Mechanisms of disease: Environmental factors in the pathogenesis of rheumatic disease. *Nat. Clin. Pract. Rheumatol.* **2007**, *3*, 172–180. [CrossRef] [PubMed]
- Zherebtsov, E.A.; Zherebtsova, A.I.; Doronin, A.; Dunaev, A.V.; Podmasteryev, K.V.; Bykov, A.; Meglinski, I. Combined use of laser Doppler flowmetry and skin thermometry for functional diagnostics of intradermal finger vessels. *J. Biomed. Opt.* **2017**, *22*, 40502. [CrossRef] [PubMed]
- Wang, T.; Lü, S.; Hao, Y.; Su, Z.; Long, M.; Cui, Y. Influence of microflow on hepatic sinusoid blood flow and red blood cell deformation. *Biophys. J.* **2021**, *120*, 4859–4873. [CrossRef]
- Schuppan, D.; Afshar, N.H. Liver cirrhosis. *Lancet* **2008**, *371*, 838–851. [CrossRef] [PubMed]
- Lewis, J.D.; Paproski, R.J.; Pink, D.B.; Vasquez, C.; Hyndman, E.; Fairey, A. Abstract PR12: Detection of EV-based signatures in prostate cancer using microflow cytometry and machine learning. *Clin. Cancer Res.* **2020**, *26*, 12. [CrossRef]
- Ilic, D.; Neuberger, M.M.; Djulbegovic, M.; Dahm, P. Screening for prostate cancer. *Cochrane Database Syst. Rev.* **2013**, *1*. [CrossRef]
- Pitts, K.L.; Abu-Mallouh, S.; Fenech, M. Contact angle study of blood dilutions on common microchip materials. *J. Mech. Behav. Biomed. Mater.* **2013**, *17*, 333–336. [CrossRef]

20. Stosseck, K.; Lübbbers, D.W.; Cottin, N. Determination of local blood flow (microflow) by electrochemically generated hydrogen: Construction and application of the measuring probe. *Pflügers Archiv* **1974**, *348*, 225–238. [[CrossRef](#)]
21. Leniger-Follert, E. Mechanisms of regulation of cerebral microflow during bicuculline-induced seizures in anaesthetized cats. *J. Cereb. Blood Flow Metab.* **1984**, *4*, 150–165. [[CrossRef](#)]
22. Ingvar, M.; Siesjö, B.K. Local blood flow and glucose consumption in the rat brain during sustained bicuculline-induced seizures. *Acta Neurol. Scand.* **1992**, *85*, 129–144. [[CrossRef](#)]
23. Soares-Silva, B.; Beserra-Filho, J.I.; Morera, P.M.; Custódio-Silva, A.C.; Maria-Macêdo, A.; Silva-Martins, S.; Alexandre-Silva, V.; Silva, S.P.; Silva, R.H.; Ribeiro, A.M. The bee venom active compound melittin protects against bicuculline-induced seizures and hippocampal astrocyte activation in rats. *Neuropeptides* **2022**, *91*, 102209. [[CrossRef](#)]
24. Faingold, C.L.; Riaz, A. Neuronal networks in convulsant drug-induced seizures. In *Drugs for the Control of Epilepsy*; CRC Press: London, UK, 2019; pp. 213–252. [[CrossRef](#)]
25. Ju, M.; Leo, H.L.; Kim, S. Numerical investigation on red blood cell dynamics in microflow: Effect of cell deformability. *Clin. Hemorheol. Microcirc.* **2017**, *65*, 105–117. [[CrossRef](#)]
26. Mojarrab, A.; Kamali, R. Design, optimization and numerical simulation of a MicroFlow sensor in the realistic model of human aorta. *Flow Meas. Instrum.* **2020**, *74*, 101791. [[CrossRef](#)]
27. Ostrom, M.P.; Gopal, A.; Ahmadi, N.; Nasir, K.; Yang, E.; Kakadiaris, I.; Flores, F.; Mao, S.S.; Budoff, M.J. Mortality incidence and the severity of coronary atherosclerosis assessed by computed tomography angiography. *J. Am. Coll. Cardiol.* **2008**, *52*, 1335–1343. [[CrossRef](#)] [[PubMed](#)]
28. Möhlenkamp, S.; Lehmann, N.; Moebus, S.; Schermund, A.; Dragano, N.; Stang, A.; Siegrist, J.; Mann, K.; Jöckel, K.H.; Erbel, R. Quantification of coronary atherosclerosis and inflammation to predict coronary events and all-cause mortality. *J. Am. Coll. Cardiol.* **2011**, *57*, 1455–1464. [[CrossRef](#)]
29. Stenvinkel, P.; Barany, P.; Heimbürger, O.; Pecoits-Filh, R.; Lindholm, B. Mortality, malnutrition, and atherosclerosis in ESRD: What is the role of interleukin-6? *Kidney Int.* **2002**, *61*, S103–S108. [[CrossRef](#)]
30. Jung, C.; Rödiger, C.; Fritzenwanger, M.; Schumm, J.; Lauten, A.; Figulla, H.R.; Ferrari, M. Acute microflow changes after stop and restart of intra-aortic balloon pump in cardiogenic shock. *Clin. Res. Cardiol.* **2009**, *98*, 469–475. [[CrossRef](#)]
31. Vahdatpour, C.; Collins, D.; Goldberg, S. Cardiogenic shock. *J. Am. Heart Assoc.* **2019**, *8*, 011991. [[CrossRef](#)]
32. Thiele, H.; Ohman, E.M.; Desch, S.; Eitel, I.; de Waha, S. Management of cardiogenic shock. *Eur. Heart J.* **2015**, *36*, 1223–1230. [[CrossRef](#)]
33. Tewelde, S.Z.; Liu, S.S.; Winters, M.E. Cardiogenic shock. *Cardiol. Clin.* **2018**, *36*, 53–61. [[CrossRef](#)] [[PubMed](#)]
34. Haykin, S. Neural networks expand SP's horizons. *IEEE Signal Process. Mag.* **1996**, *13*, 24–49. [[CrossRef](#)]
35. Gupta, N. Artificial neural network. *Netw. Complex Syst.* **2013**, *3*, 24–28.
36. Ramanujan, V.; Wortsman, M.; Kembhavi, A.; Farhadi, A.; Rastegari, M. What's hidden in a randomly weighted neural network? In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 11893–11902.
37. Kello, C.T.; Plaut, D.C. A neural network model of the articulatory-acoustic forward mapping trained on recordings of articulatory parameters. *J. Acoust. Soc. Am.* **2004**, *116*, 2354–2364. [[CrossRef](#)]
38. Zhang, Z.; Zhang, Z. Artificial neural network. In *Multivariate Time Series Analysis in Climate and Environmental Research*; Springer: Cham, Switzerland, 2018; pp. 1–35. [[CrossRef](#)]
39. Alfonso, G.; Carnerero, A.D.; Ramirez, D.R.; Alamo, T. Receding Horizon Optimization of Large Trade Orders. *IEEE Access* **2021**, *9*, 63865–63875. [[CrossRef](#)]
40. Paproski, R.J.; Pink, D.; Sosnowski, D.L.; Vasquez, C.; Lewis, J.D. Building predictive disease models using extracellular vesicle microscale flow cytometry and machine learning. *Mol. Oncol.* **2023**, *17*, 407–421. [[CrossRef](#)]
41. Ananda, A.; Ngan, K.H.; Karabağ, C.; Ter-Sarkisov, A.; Alonso, E.; Reyes-Aldasoro, C.C. Classification and visualisation of normal and abnormal radiographs; a comparison between eleven convolutional neural network architectures. *Sensors* **2021**, *21*, 5381. [[CrossRef](#)] [[PubMed](#)]
42. Pontes, F.J.; Amorim, G.F.; Balestrassi, P.P.; Paiva, A.P.; Ferreira, J.R. Design of experiments and focused grid search for neural network parameter optimization. *Neurocomputing* **2016**, *186*, 22–34. [[CrossRef](#)]
43. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305. [[CrossRef](#)]
44. Qiu, F.; Jensen, J.R. Opening the black box of neural networks for remote sensing image classification. *Int. J. Remote Sens.* **2004**, *25*, 1749–1768. [[CrossRef](#)]
45. Yoon, Y.; Guimaraes, T.; Swales, G. Integrating artificial neural networks with rule-based expert systems. *Decis. Support Syst.* **1994**, *11*, 497–507. [[CrossRef](#)]
46. Müller, V.C.; Bostrom, N. *Fundamental Issues of Artificial Intelligence*; Springer: Cham, Switzerland, 2016; Volume 376, ISBN 978-3-319-26483-7.
47. Lugnan, A.; Gooskens, E.; Vatin, J.; Dambre, J.; Bienstman, P. Machine learning issues and opportunities in ultrafast particle classification for label-free microflow cytometry. *Sci. Rep.* **2020**, *10*, 20724. [[CrossRef](#)] [[PubMed](#)]
48. Guo, J.; Wang, J.; Wen, C.; Jin, S.; Li, G.Y. Compression and acceleration of neural networks for communications. *IEEE Wirel. Commun.* **2020**, *27*, 110–117. [[CrossRef](#)]

49. Xing, J.; Luo, K.; Pitsch, H.; Wang, H.; Bai, Y.; Zhao, C.; Fan, J. Predicting kinetic parameters for coal devolatilization by means of Artificial Neural Networks. *Proc. Combust. Inst.* **2019**, *37*, 2943–2950. [[CrossRef](#)]
50. Wu, B.; Han, S.; Shin, K.G.; Lu, W. Application of artificial neural networks in design of lithium-ion batteries. *J. Power Sources* **2019**, *395*, 128–136. [[CrossRef](#)]
51. Shrivastava, G.; Karmakar, S.; Kowar, M.K.; Guhathakurta, P. Application of artificial neural networks in weather forecasting: A comprehensive literature review. *Int. J. Comput. Appl.* **2012**, *51*, 17–29. [[CrossRef](#)]
52. Baek, M.; DiMaio, F.; Anishchenko, I.; Dauparas, J.; Ovchinnikov, S.; Lee, G.R.; Wang, J.; Cong, Q.; Kinch, L.N.; Schaeffer, R.D. Accurate prediction of protein structures and interactions using a three-track neural network. *Science* **2021**, *373*, 871–876. [[CrossRef](#)] [[PubMed](#)]
53. Wang, Y.; Zou, R.; Liu, F.; Zhang, L.; Liu, Q. A review of wind speed and wind power forecasting with deep neural networks. *Appl. Energy* **2021**, *304*, 117766. [[CrossRef](#)]
54. Kloosterman, A.; Hierck, B.; Westerweel, J.; Poelma, C. Quantification of blood flow and topology in developing vascular networks. *PLoS ONE* **2014**, *9*, e396856. [[CrossRef](#)]
55. Kim, T.Y.; Oh, K.J.; Kim, C.; Do, J.D. Artificial neural networks for non-stationary time series. *Neurocomputing* **2004**, *61*, 439–447. [[CrossRef](#)]
56. Cheng, C.; Sa-Ngasoongsong, A.; Beyca, O.; Le, T.; Yang, H.; Kong, Z.; Bukkapatnam, S. Time series forecasting for nonlinear and non-stationary processes: A review and comparative study. *Iie Trans.* **2017**, *47*, 70501. [[CrossRef](#)]
57. Ghazali, R.; Hussain, A.J.; Nawi, N.M.; Mohamad, B. Non-stationary and stationary prediction of financial time series using dynamic ridge polynomial neural network. *Neurocomputing* **2009**, *72*, 2359–2367. [[CrossRef](#)]
58. Marseguerra, M.; Minoggio, S.; Rossi, A.; Zio, E. Neural networks prediction and fault diagnosis applied to stationary and non stationary ARMA modeled time series. *Prog. Nucl. Energy* **1992**, *27*, 25–36. [[CrossRef](#)]
59. Burden, F.; Winkler, D. Bayesian regularization of neural networks. In *Artificial Neural Networks: Methods and Applications*; Humana Press: Totowa, NJ, USA, 2009; pp. 23–42. [[CrossRef](#)]
60. Benardos, P.G.; Vosniakos, G.C. Optimizing feedforward artificial neural network architecture. *Eng. Appl. Artif. Intell.* **2007**, *20*, 365–382. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.