Revista Eletrônica
Paulista de Matemática

ISSN 2316-9664
v. 27, 2026
Artigo de Pesquisa

Vinícius Liberato Erler
Universidade Estadual Paulista (UNESP), Instituto de Química, Araraquara

Juliana Maria da Silva
Universidade Federal de Alfenas, Instituto de Ciência e Tecnologia, Poços de Caldas

Érica Regina Filletti
Universidade Estadual Paulista (UNESP), Instituto de Química, Araraquara, erica.filletti@unesp.br

Artificial intelligence-based prediction of crossflow microfiltration with a graphical user interface

Predição da microfiltração tangencial baseada em inteligência artificial com uma interface gráfica de usuário

Abstract

The neural network model is used to estimate the permeate flux during the crossflow microfiltration process of aqueous solutions containing guar and xanthan gums at temperatures of 25°C and 40°C. For the simulations, membrane type, solution composition, filtration time, temperature, transmembrane pressure, and flow velocity were used as input variables. The model, developed in Python using the TensorFlow library, achieved an average percentage error of 2.71% for the training dataset and 5.79% for the test dataset. In addition, a graphical user interface (GUI) was developed using the Glade library, allowing users to easily and efficiently obtain permeate flux values.
Keywords: crossflow filtration; permeate flux; artificial neural networks; graphical user interface.

Resumo

O modelo de rede neural é utilizado para estimar o fluxo de permeado durante o processo de microfiltração tangencial de soluções aquosas contendo gomas guar e xantana, nas temperaturas de 25 °C e 40 °C. Para as simulações, foram utilizadas como variáveis de entrada o tipo de membrana, a composição da solução, o tempo de filtração, a temperatura, a pressão transmembrana e a velocidade de escoamento. O modelo, desenvolvido em Python utilizando a biblioteca TensorFlow, alcançou um erro percentual médio de 2,71% para o conjunto de dados de treinamento e de 5,79% para o conjunto de teste. Além disso, foi desenvolvida uma interface gráfica do usuário (GUI) utilizando a biblioteca Glade, permitindo aos usuários obter valores de fluxo de permeado de forma fácil e eficiente.
Palavras-chave: filtração tangencial; fluxo de permeado; redes neurais artificiai; interface gráfica.

Artigo recebido em jun. 2026 e aceito em nov. 2026

Este artigo está licenciado com uma Licença Creative Commons Attribution 4.0 International, podendo ser usado, distribuído e reproduzido, sem restrições, desde que o trabalho original seja devidamente citado.

1 Introduction

Crossflow membrane filtration is widely used in industrial processes. The performance of crossflow microfiltration is governed by complex physical phenomena, including permeate transport through the membrane, concentration polarization, mass transfer, and the development of additional resistance due to membrane fouling. Consequently, considerable research has focused on characterizing and modeling these phenomena, particularly their effects on permeate flux, which is a key indicator of filtration performance.

Mechanistic models have been developed to describe the physical phenomena governing membrane filtration and predict permeate flux under different operating conditions [1, 2, 3, 4]. Depending on the system under investigation, these models may account for membrane resistance, concentration polarization, osmotic pressure, mass transfer, and fouling mechanisms. Various approaches have been proposed, including film theory, gel polarization, boundary-layer adsorption, Brownian diffusion, shear-induced diffusion, and surface transport models. Although these approaches provide valuable insights into filtration behavior, they often involve simplifying assumptions, complex mathematical formulations, and parameters that must be determined experimentally. Moreover, analytical solutions are not always available, making numerical methods necessary to solve the governing equations. The applicability of these models may also be limited to specific feed characteristics and operating conditions. These limitations motivate the investigation of alternative approaches to permeate flux prediction, particularly when the primary objective is to obtain accurate estimates under different experimental conditions.

Artificial neural networks (ANNs) offer a data-driven approach to predicting filtration performance. By learning the relationships between input variables and experimentally measured outputs, ANNs can capture complex nonlinear patterns without explicitly describing the underlying transport mechanisms. This capability makes them particularly suitable when the primary objective is to predict process variables accurately rather than to provide a mechanistic explanation of the phenomena involved. Accordingly, numerous studies have investigated the application of ANNs to modeling and predicting the performance of membrane filtration processes.

Guadix *et al.* [5] developed an ANN to predict the temporal evolution of permeate flux during the filtration of milk through a ceramic membrane, considering filtration and cleaning cycles with varying degrees of aggressiveness. The model yielded satisfactory results, with a prediction error of approximately 10%. Ghandehari *et al.* [6] experimentally investigated flux decline under different operating conditions during the crossflow microfiltration of bovine serum albumin (BSA). The resulting experimental dataset was subsequently used to develop feedforward backpropagation (BP) and radial basis function (RBF) ANN models. The authors reported that the RBF model exhibited better predictive performance than the BP model, particularly for membrane rejection. Sekulić *et al.* [7] employed two ANN architectures, namely a General Regression Neural Network and a Backpropagation Neural Network, to predict rejection coefficients during crossflow microfiltration for the removal of heavy metal ions, including lead, cadmium, and zinc, from water.

In addition to these individual studies, several reviews have examined the application of artificial intelligence to membrane processes. Gaudio *et al.* [8] reviewed studies on membrane-based water treatment and fouling control, highlighting the use of artificial intelligence techniques, particularly ANNs and genetic algorithms. Jawad *et al.* [9] provided a comprehensive review of ANN models developed for membrane processes applied to wastewater treatment and desalination over approximately 25 years, discussing model development strategies and the comparative performance of different ANN architectures. More recently, Abuwatfa *et al.* [10] focused on artificial intelligence-based

membrane fouling prediction models, emphasizing ANN-based approaches and their derivatives, as well as their strengths, limitations, potential applications, and opportunities for improvement. Other studies have also investigated the use of ANN models to predict process variables and performance indicators in crossflow filtration [11, 12, 13, 14, 15, 16, 17].

Building on previous research, the present study investigates the use of an artificial neural network (ANN) to predict permeate flux during the crossflow microfiltration of aqueous solutions containing guar and xanthan gums. The model's predictive performance is assessed by comparing its predictions with experimentally measured permeate flux values, achieving an average prediction error below 10%. The objective is to evaluate the predictive capability of the ANN under the experimental conditions investigated, rather than to develop an explicit mechanistic description of the filtration process.

This study extends the methodology proposed by [18] for predicting permeate flux during crossflow microfiltration using artificial neural networks (ANNs). In addition to assessing the model's predictive performance, the present work incorporates the trained network into a graphical user interface (GUI), providing a practical and accessible computational tool. The interface allows users to estimate permeate flux for different guar–xanthan gum mixtures and tubular membrane configurations without requiring programming expertise. Therefore, the main contribution of this study is the development of a standalone graphical interface that integrates an ANN-based predictive model tailored to the experimental conditions investigated, facilitating its practical application.

2 Artificial neural network and graphic interface

For the development of the artificial neural network algorithm, the TensorFlow library was employed within the Python programming environment, together with auxiliary libraries including Matplotlib, Seaborn, Pandas, Scikit-learn, Joblib, NumPy, and OpenPyXL. These libraries were used for complementary tasks such as data acquisition, preprocessing, graphical analysis, variable manipulation, and spreadsheet handling in Microsoft Excel.

The initial stage of the algorithm development consisted of preprocessing the dataset to ensure proper training performance. This step involved converting all categorical variables into numerical representations, followed by the normalization of the numerical variables.

The dataset comprised two categorical variables: gum proportion and membrane type. The one-hot encoding technique was applied to transform these categorical variables into numerical vectors according to the number of categories present in each variable. Consequently, three additional columns were generated for membrane type and seven for gum proportion. In these columns, the values 0 and 1 indicate the absence or presence of a given category in a specific experiment.

Normalization of the numerical variables was performed using Equation (1),

$$X_{modified} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

where X represents the variable to be normalized, X_{min} is the minimum value of X in the dataset, and X_{max} is the corresponding maximum value. The purpose of normalization is to prevent the algorithm from becoming biased toward variables with larger magnitudes. As a result, all variables are scaled to the range between 0 and 1, which also contributes to faster and more stable training. In this study, the variables related to transmembrane pressure, temperature, flow velocity, and filtration time were normalized.

Subsequently, the dataset was divided into training, validation, and test subsets. The training subset was used to adjust the network weights, while the validation subset was used to monitor the model during training and to determine the stopping point through the early stopping procedure. A separate test subset was reserved exclusively for the final evaluation of the trained model. Of the total dataset comprising 891 experiments, 70% of the data were allocated to the model development stage, including training and validation, while the remaining 30% were reserved for testing. Thus, the test observations were not used for adjusting the network weights and were retained for an independent assessment of predictive performance.

One of the most critical steps in the development of an artificial neural network is defining the model architecture, including the number of layers, the number of neurons per layer, and the activation function. These parameters depend on the complexity of the problem and the preprocessing applied to the dataset and are typically determined empirically. In this work, the Rectified Linear Unit (ReLU) activation function was adopted for all neurons, as it maps negative input values to zero while keeping positive values unchanged.

During model configuration, the optimizer and its associated parameters—learning rate, decay, and clip value—were also defined. The learning rate controls the step size during weight updates and directly affects the convergence speed of the model. Higher learning rates accelerate training but may hinder convergence to an optimal solution, whereas lower rates slow training while improving convergence stability. The decay parameter gradually reduces the learning rate over successive epochs, facilitating effective convergence. The clip value limits the magnitude of gradients during training, preventing excessively large updates to the network weights. The Adam optimizer, a stochastic gradient descent–based method, was selected and configured with a learning rate of 0.001, a decay of 0.1, and a clip value of 0.5.

Another relevant parameter is the batch size, which defines the number of samples processed before updating the model weights. In this study, a batch size of 16 was adopted.

Additionally, callbacks were implemented during the training process to improve efficiency and prevent overfitting. An early stopping callback was configured to monitor the model's performance on both the training and validation datasets, interrupting the training process once optimal performance was achieved with minimal loss variation. A second callback was used to record performance metrics throughout the training epochs, enabling subsequent comparative analysis of the developed models.

Finally, a graphical user interface was developed using the Python programming language with the Glade library. This library allows the visual design of the interface to be created independently while enabling the underlying structure and logic to be implemented in Python. The graphical interface was compiled into an executable file using the cx_Freeze library, allowing the neural network model to be executed on any computer without requiring prior installation of Python or the libraries used in this study.

3 Crossflow microfiltration of xanthan gum and guar gum solution

The experimental procedures for the crossflow microfiltration of xanthan and guar gum solutions were developed and conducted in the master's dissertation by [19]. The data reported in that study were used to develop the artificial neural network models. These data were obtained under a wide range of operating conditions, including different membrane types, two temperature levels, varying

proportions of xanthan and guar gums in the feed solution, and different values of transmembrane pressure, filtration time, and flow velocity. Table 1 summarizes the ranges of time, temperature, transmembrane pressure, and flow velocity considered in the experiments.

Table 1: Range of time, temperature, transmembrane pressure, and flow velocity used in the experiments.

Time (min)	Temperature (^oC)	Pressure (kPa)	Flow velocity (m/s)
3 to 67.5	25	300	2.6
		400	3.7
	40	500	4.7
			5.7

Source: adapted from Queiroz (2004).

Table 2 presents the distribution of experiments by membrane type and solution composition, based on guar and xanthan gum concentrations and indicates that the experimental dataset comprises 891 microfiltration experiments and includes two membrane types: monochannel and multichannel. Monochannel membranes consist of a single filtering layer, whereas multichannel membranes are composed of multiple layers that create distinct flow channels.

Table 2: Distribution of experiments by membrane type and solution composition.

Solution Proportion/ Membrane Type	Monochannel Membrane 0.4 μm	Monochannel Membrane 0.2 μm	Multichannel Membrane 0.2 μm	Total
1000 ppm xanthan	167	24	24	215
1000 ppm guar	167	18	25	210
100 ppm xanthan and 900 ppm guar	24	18	24	66
250 ppm xanthan and 750 ppm guar	20	10	24	54
500 ppm xanthan and 500 ppm guar	168	20	24	212
750 ppm xanthan and 250 ppm guar	24	12	25	61
900 ppm xanthan and 100 ppm guar	24	24	25	73
Total	594	126	171	891

Source: adapted from Queiroz (2004).

In this study, permeate flux prediction was based on membrane type, solution composition, filtration time, temperature, transmembrane pressure, and flow velocity. All experimental data were stored in a Microsoft Excel spreadsheet and imported using the Pandas library for training the artificial neural network model.

4 Results and discussion

The performance of the artificial neural network developed for permeate flux prediction in crossflow microfiltration is evaluated using statistical and error-based metrics by comparing predicted and experimental values to assess accuracy and generalization. The implementation of a graphical user interface is also presented.

4.1 Development of the artificial neural network algorithm

Initially, the dataset comprising 891 experiments was loaded from a spreadsheet, and the categorical and numerical variables were processed. To facilitate the rapid identification of an optimal network architecture, a routine was developed to generate models with randomly defined numbers of neurons and hidden layers within predefined ranges. A wide variety of architectures was evaluated, ranging from simpler models with few neurons and a single hidden layer to more complex configurations with more than 2,000 neurons per layer and up to eight hidden layers.

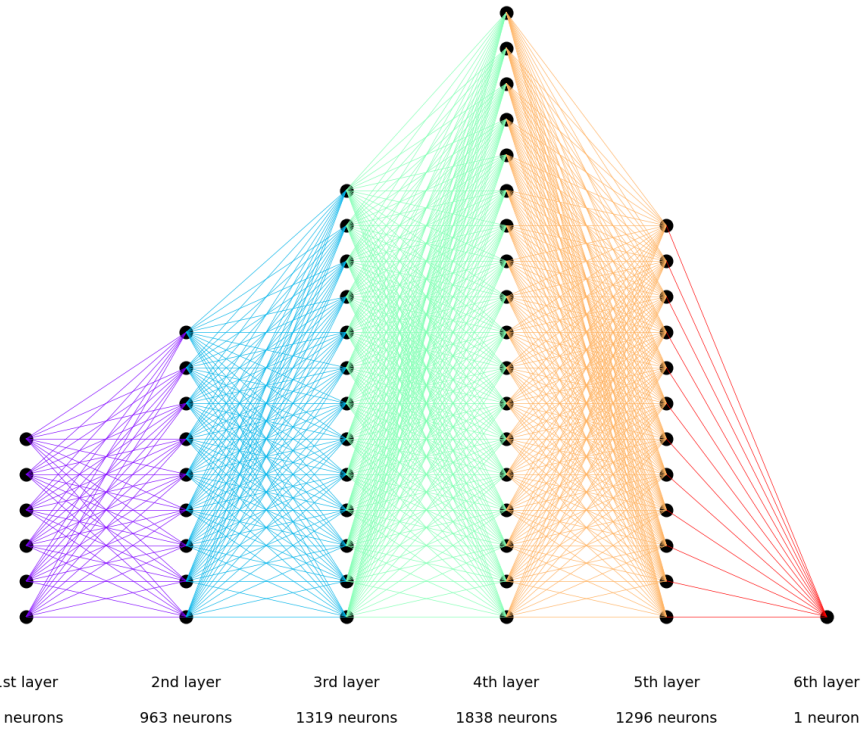
In the spreadsheet, selecting the cells in the "Model Structure," "Loss," and "Val_Loss" columns enabled visualization of a summary of the automatically generated network architecture, the training error values across epochs, and the validation error values across epochs, respectively. In addition, for each model, the average percentage difference between the predicted and experimental permeate flux values was computed for both the training and test datasets. Based on these metrics, the strategy for defining the number of neurons per layer and the number of hidden layers was iteratively adjusted to improve model performance.

After developing and evaluating multiple models, the most effective strategy consisted of randomly defining the number of neurons in the hidden layers within the range of 800 to 2,000, while varying the number of hidden layers between four and eight. The primary objective was to obtain a model in which the average percentage difference between predicted and experimental permeate flux values was below 10% for both the training and test datasets.

The architecture selection was based primarily on predictive performance within the experimental dataset. The objective of the architecture search was not to minimize the number of trainable parameters, but to identify a configuration capable of reproducing the experimentally measured permeate flux with an average prediction difference below the predefined 10% threshold for both the training and test datasets. Consequently, the final architecture should be interpreted as an empirically selected configuration for the experimental system investigated, rather than as a generally optimal neural network architecture. Although this configuration results in a relatively high number of trainable parameters, computational efficiency was not adopted as a selection criterion in the present study, whose primary objective was to achieve the predefined predictive-error target.

The best-performing model has an architecture composed of 14 neurons in the input layer, each corresponding to a predictor variable, followed by four hidden layers. Of the 14 input neurons, three represent the membrane type parameter, seven correspond to the solution composition parameter, and the remaining four represent numerical predictors (time, temperature, transmembrane pressure, and flow velocity). The hidden layers contain 963, 1319, 1838, and 1296 neurons, respectively. The output layer consists of a single neuron corresponding to the predicted permeate flux. The network architecture is illustrated in Figure 1, where the graphical representation of neurons in the hidden layers has been reduced by a factor of 100 to enhance visual clarity. Overall, the model comprises a total of 6,096,762 trainable parameters used to predict the permeate flux.

Figure 1: Schematic representation of the architecture of the best-performing artificial neural network.



Source: Prepared by the author.

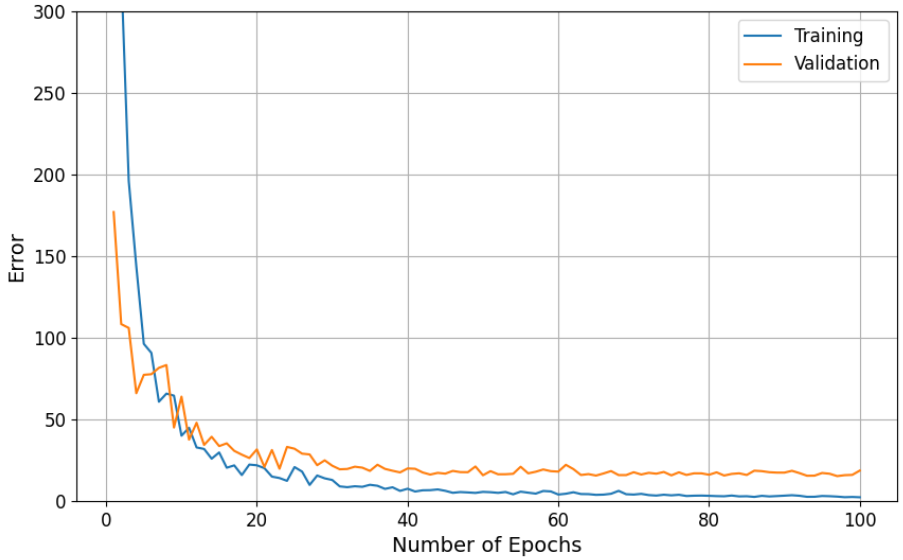
Regarding the average percentage difference between the experimental and predicted permeate flux values, the proposed model achieved values of 2.71% for the training set and 5.79% for the test set. These results are considered highly satisfactory, given the initial target threshold of 10%.

An analysis of the model predictions across the entire dataset indicated that only 7.29% of the experiments exhibited percentage differences exceeding the 10% threshold, with a maximum observed deviation of 48.51%. Considering all experiments, the mean percentage difference was $(3.64 \pm 5.21)\%$, where the standard deviation reflects the variability of the prediction errors.

Additionally, 70.03% of the experiments presented percentage differences below the overall mean value of 3.64%, further support the predictive capability of the model within the experimental domain investigated.

As shown in Figure 2, the error profiles for both the training and validation sets decrease rapidly over the epochs, indicating that convergence was achieved at approximately epoch 70. Nevertheless, the training process was extended to epoch 100, as initially defined, to account for the possibility that some randomly generated network architectures might require additional epochs to effectively minimize the error. As expected, the validation error remains higher than the training error, as the model tends to become increasingly biased toward the training data during the learning process. The relatively small difference between the training and validation error curves provides no clear evidence of substantial overfitting under the adopted training and validation procedure. In addition, the performance observed on the independent test subset provides a complementary assessment of the predictive behavior of the selected model. Nevertheless, the relatively high number of trainable parameters should be considered when interpreting the model complexity, and the present results are limited to the experimental domain represented in the dataset.

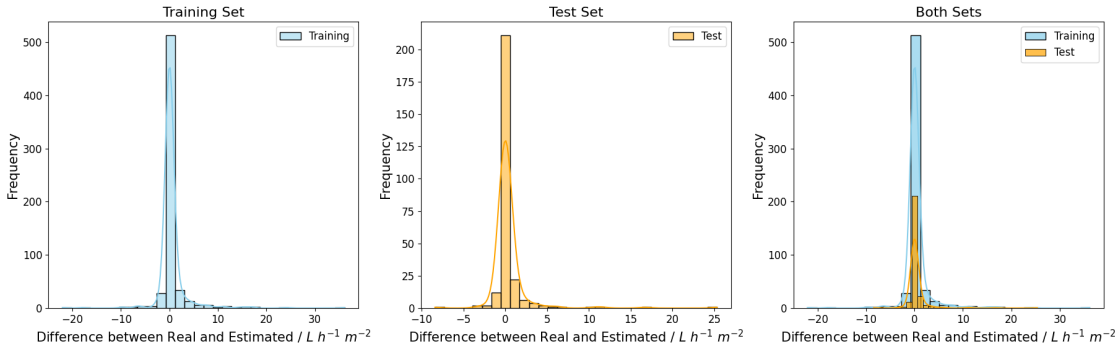
Figure 2: Training and validation error as a function of the number of epochs.



Source: Prepared by the author.

Regarding the error histogram presented in Figure 3, the prediction errors for both the training and test sets are concentrated around zero, exhibiting a highly homogeneous distribution. This behavior further indicates the satisfactory performance of the proposed model. It can also be observed that, for both datasets, the vast majority of errors fall within the range of -5 to $+5$ $Lh^{-1}m^{-2}$.

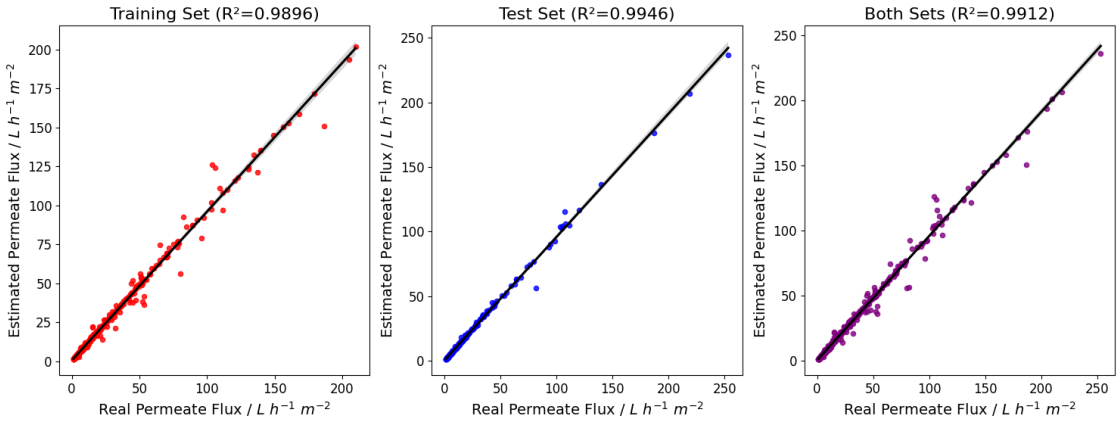
Figure 3: Histogram of prediction errors for the training and test sets.



Source: Prepared by the author.

Concerning the linear regression analysis of the model, shown in Figure 4, the coefficient of determination (R^2) ranges from 0 to 1, with values closer to unity indicating higher predictive accuracy in reproducing the experimental permeate flux. The obtained R^2 values were 0.9896 for the training set, 0.9946 for the test set, and 0.9912 when considering both datasets simultaneously.

Figure 4: Linear regression between predicted and experimental permeate flux values.



Source: Prepared by the author.

Only a small number of points deviate from the 45^o reference line, which represents ideal agreement between the predicted and experimental permeate flux values. These results further support the predictive capability of the developed neural network model within the experimental domain investigated.

The predictive capability demonstrated in this study should be interpreted within the experimental domain represented in the dataset. The model was developed using specific membrane configurations, guar-xanthan gum compositions, temperatures, transmembrane pressures, flow velocities, and filtration times. Therefore, the present results do not imply unrestricted generalization to membrane systems or operating conditions outside the range investigated experimentally. Application to substantially different experimental conditions may require additional experimental data and subsequent model retraining or recalibration.

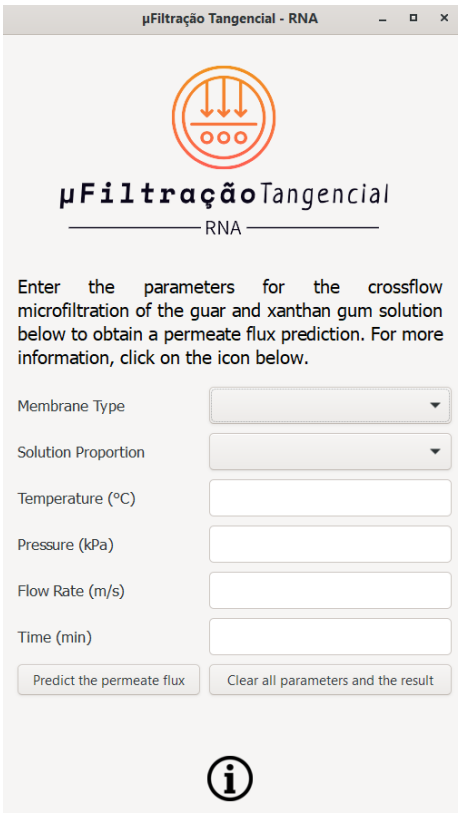
4.2 Development of the graphical interface

To develop the graphical user interface (GUI), the Glade library was employed within the Python environment. This library provides a dedicated application for designing the interface layout while allowing the integration of the necessary logic to ensure proper functionality.

During the interface development process, objects such as buttons and text boxes can be easily arranged to define the overall GUI structure. Within this environment, it is also possible to configure basic interface settings, such as adding essential buttons and images, as well as performing more specific adjustments, including window positioning and size configuration.

The logical component of the graphical user interface was developed using the Python programming environment, specifically within Jupyter Notebook. In this environment, functions corresponding to the actions associated with each interface button were implemented, along with the loading of the trained neural network model and the execution of the procedures required to ensure accurate permeate flux prediction (Figure 5).

Figure 5: Design of the developed graphical user interface.



Source: Prepared by the author.

A second validation step checks whether all interface fields have been completed, including the two selection lists corresponding to membrane type and guar–xanthan gum solution composition. Once field completeness is confirmed, the function verifies that the temperature, transmembrane pressure, flow velocity, and time inputs contain only numerical characters, allowing exclusively digits and decimal separators (comma or period). When all inputs are correctly provided and validated, the predicted permeate flux value is displayed on the interface. Otherwise, an error message stating "Missing or incorrectly entered parameters" is shown to the user.

Among the implemented functions, the permeate flux prediction function was the most complex, as it needed to account for several potential user input errors in the temperature, transmembrane pressure, flow velocity, and time fields. Initially, the function stores the user-provided values from all input fields into variables. The first validation step verifies whether numerical values were entered using a comma as the decimal separator, which is common in some locales. Since Python requires a period as the decimal separator, any commas are automatically converted to periods to ensure correct numerical processing. If periods are already used, no changes are applied.

The "Clear all parameters and the result" button was implemented to facilitate successive predictions by resetting all input fields and removing the previously displayed result.

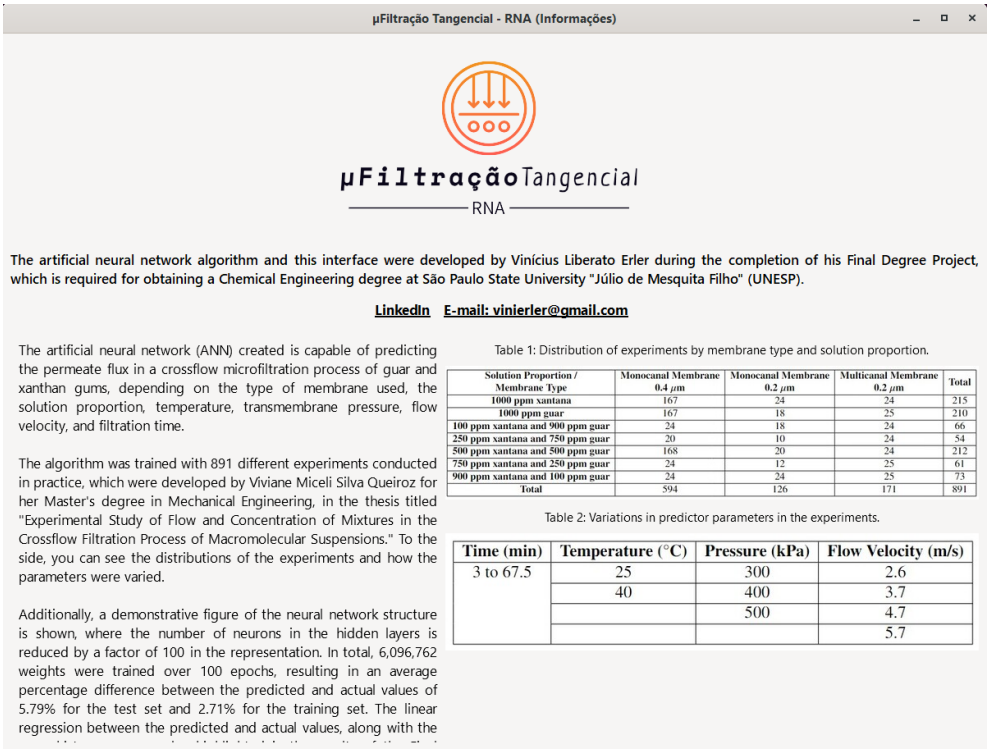
Finally, the information button located at the bottom of the interface provides access to explanatory content related to this work, including the figure depicting the developed neural network model structure, as well as the additional graphs presented throughout the study.

After completing the development of both the interface design and the application logic, the software was compiled into an executable file using the cx_Freeze library. This process enabled the

graphical user interface, embedded with the artificial neural network model, to be executed on any personal computer running the Windows operating system (version 10 or later). To ensure proper execution, the system may require Microsoft Visual C++ Redistributable packages (2005, 2008, 2010, 2012, 2013, and 2015–2022), as well as Microsoft DirectX. The Python interpreter and the libraries used in this work do not need to be installed separately, as the compilation process automatically identifies and includes all required files and dependencies within the application directory.

To run the interface, the user only needs to launch the executable application entitled "μFiltração Tangencial – RNA," which contains the compiled program and associated resources (Figure 6).

Figure 6: Final graphical user interface design.



Source: Prepared by the author.

The graphical user interface, together with the artificial neural network model, is available at the following link: https://bit.ly/micro-tangencial-rna-english.

5 Conclusion

The developed neural network exhibited satisfactory predictive performance, with average percentage differences of 2.71% for the training set and 5.79% for the test set, both below the predefined threshold of 10%. These results indicate good agreement between the predicted and experimentally measured permeate flux values. Analyses of error evolution over the training epochs, error distribution, and linear regression further support the predictive capability of the model within the experimental domain investigated.

In addition, the graphical user interface proved effective, providing a simple and intuitive tool that can be readily used by third-party users without requiring the installation of Python or its associated

libraries. The availability of the standalone software through the provided link further enhances the practical applicability of the proposed approach.

It should be emphasized that the predictive performance demonstrated in this study is specific to the experimental domain represented by the available dataset. Comparisons with alternative machine-learning regression methods, repeated cross-validation, systematic input-variable ablation, and formal uncertainty quantification could provide further insights into model selection, performance variability, and generalization capability. These analyses represent promising directions for future research.

References

[1] SILVA, J. M.; FERREIRA, V. G.; FONTES, S. R. An evaluation of three upwinding approximations for numerical modeling the flow in tubular membrane of newtonian and non-newtonian fluids. **Applied Mathematics and Computation**, New York, v. 217, p. 7955–7965, 2011.

[2] HSU, M. *et al.* Influence of pore size distribution and applied cross-flow on ion rejection and separation. **Separation and Purification Technology**, Amsterdam, v. 352, 2025.

[3] BRASEL, B. *et al.* Evolution of particle deposits at communicating membrane pores during crossflow filtration. **Journal of Membrane Science**, Amsterdam, v. 686, 2023.

[4] PAIPUR, M. *et al.* Numerical modelling of concentration polarisation and cake formation in membrane filtration processes. **Desalination**, Amsterdam, v. 365, p. 151– 159, 2015.

[5] GUADIX, A. *et al.* Predicting the flux decline in milk cross-flow ceramic ultrafiltration by artificial neural networks. **Desalination**, Amsterdam, v. 250, n. 3, p. 1118–1120, 2010.

[6] GHANDEHARI, S.; MONTAZER-RAHMATI, M. M.; ASGHARI, M. Modeling the flux decline during protein microfiltration: A comparison between feed-forward back propagation and radial basis function neural networks. **Separation Science and Technology**, New York, v. 48, n. 9, p. 1324–1330, 2013.

[7] SEKULIĆ, Z. *et al.* Application of artificial neural networks for estimating cd, zn, pb removal efficiency from wastewater using complexation microfiltration process. **International Journal of Environmental Science and Technology**, Tehran, v. 14, p. 1383–1396, 2017.

[8] GAUDIO, M. T. *et al.* Artificial intelligence-based optimization of industrial membrane processes. **Earth Systems and Environment**, Cham, v. 5, p. 385–398, 2021.

[9] JAWAD, J.; HAWARI, A. H.; ZAIDI, S. J. Artificial neural network modeling of wastewater treatment and desalination using membrane processes: a review. **Chemical Engineering Journal**, Amsterdam, v. 419, p. 1–21, 2021.

[10] ABUWATFA, W. H. *et al.* A review on membrane fouling prediction using artificial neural networks (anns). **Membranes**, Basel, v. 13, n. 7, p. 4–29, 2023.

[11] SHETTY, G. R.; CHELLAM, S. Prediction of membrane fouling during municipal drinking water nanofiltration using artificial neural networks. **Journal of Membrane Science**, Amsterdam, v. 217, p. 69–86, 2003.

[12] CURCIO, S. *et al.* Ultrafiltration of bsa in pulsating conditions: An artificial neural networks approach. **Journal of Membrane Science**, Amsterdam, v. 246, p. 235–247, 2005.

[13] CURCIO, S.; CALABRÒ, V.; IORIO, G. Reduction and control of flux decline in cross-flow membrane processes modeled by artificial neural networks. **Journal of Membrane Science**, Amsterdam, v. 286, p. 125–132, 2006.

[14] CHELLAM, S. Artificial neural network model for transient crossflow microfiltration of polydispersed suspensions. **Journal of Membrane Science**, Amsterdam, v. 258, p. 35–42, 2005.

[15] SAHOO, G. B.; RAY, C. Predicting flux decline in crossflow membranes using artificial neural networks and genetic algorithms. **Journal of Membrane Science**, Amsterdam, v. 283, p. 147–157, 2006.

[16] SILVA, I. N.; FLAUZINO, R. A. An approach based on neural networks for estimation and generalization of crossflow filtration processes. **Applied Soft Computing**, Amsterdam, v. 8, p. 590–598, 2008.

[17] HILAL, N.; OGUNBIYI, O. O.; AL-ABRI, M. Neural network modeling for separation of bentonite in tubular ceramic membranes. **Desalination**, Amsterdam, v. 228, p. 175–182, 2008.

[18] ERLER, V. L.; SILVA, J. M.; NASCIMENTO, E. R. F. Desenvolvimento de um modelo neural para prever o fluxo de permeado de um sistema de microfiltração tangencial de gomas guar e xantana. *In*: ENCONTRO REGIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, 7., 2024, Bauru. **Caderno de trabalhos completos e resumos** [...]. Bauru: Unesp, Faculdade de Ciências, 2024. p. 107–113.

[19] QUEIROZ, V. M. S. **Estudo Experimental do Escoamento e da Concentração de Mistura no Processo de Filtração Tangencial de Suspensões Macromoleculares**. 2004. 119 f. Dissertação (Mestrado em Engenharia Mecânica) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2004.