Unsteady parametrized Stokes equations in a 2D arterial bifurcation with stenosis: design of an Autoencoder for data compression

Camille Frayssinhes, Assia Ouanaya, Théau Vannier
Department of Computer Sciences, EPFL, Switzerland

Abstract

This project is done in the context of the Partial Differential Equation with Deep Neural Networks (PDE-DNN) project run by Professor's Deparis lab. Using pre-existing code, we simulate the blood flow in an arterial bifurcation featuring stenosis relying on the Stokes equations and the finite element method. Our goal is to implement an autoencoder (AE) to compress spatio-temporal samplings of the mathematical solutions of the blood flow. Ideally, we would expect the product of the compression to lay on a 5-dimensional subspace. Indeed, we know that there are exactly 5 physical parameters that characterize the equations, so theoretically we could get lossless compression with 5 neurons in the latent space. Additionally, we investigate the relationship between the number of "abstract parameters" and the 5 initial physical parameters, using standard regression models. Indeed, modelling this relationship enables us to recover the 5 initial parameter values from the learned compressed solution and, therefore, to reconstruct the blood flow in the whole spatio-temporal computational domain, despite knowing only a subsample of it.

INTRODUCTION

In the past few years, machine learning has become an increasingly attractive tool for developing low-dimensional models of complex systems. Deep recurrent AEs have already been used for learning low-dimensional feature dynamics of fluid systems. [1] In this project based on the work of Deparis et al. [2], we aim to identify a low-dimensional representation of a high-dimensional spatio-temporal dataset. The dataset models the blood flow in an arterial bifurcation with stenosis, i.e., a narrowing of the artery which may partially or completely occlude it, thus hampering the blood flow.

The report is organized as follows. The first section formulates the problem of interest. The second section reviews the core concepts of deep learning used in this work. The key contributions of the present work are introduced in the third section, namely, the construction of the feed-forward AE and the long short-term memory (LSTM) AE for dimensionality reduction of blood flow dynamics. In section 4, we show the results obtained with the aforementioned methods. Section 5 focuses on the interpretation of the latent neurons in terms of the initial physical parameters. Finally, in the last section, we present a summary and a discussion of our work.

I. PROBLEM FORMULATION

Our project aims at investigating the relationship between classical numerical methods and deep-learning techniques, with the final purpose of efficiently providing solutions to parametrized PDE problems, being fully unaware of the values of the characteristic parameters. The lab headed by professor Deparis has developed efficient methods for the numerical approximation of solutions to parametrized time-dependent PDEs. In the following, we develop DNNs that take advantage of these numerical methods to compress the solutions of the parametrized unsteady PDEs. In particular, we employ PyTorch feed-forward AEs and LSTM AEs to learn an optimal low-dimensional representation of the full state of the system.

A. Data generation

The data used in our work describes the blood flow in an artery with a bifurcation and a stenosis, which is modelled as the union of two semi-elliptical plaques. The dataset is generated by running MATLAB numerical simulations, which solve unsteady Stokes equations with random parameters characterizing the stenonis. The output of each simulation is the discrete approximation of the time-evolution of the blood velocity and pressure in the considered domain, which is discretized with a suitable mesh grid. There are 5 parameters that can be set: μ_1 stenosis width, μ_2 stenosis height, μ_3 distance between the 2 stenosis centers, R_1 and R_2 resistive term of the upper and lower bifurcation branch respectively (Figure 1). The dataset is generated by performing 175 simulations corresponding to randomly selected combinations of the 5 model parameters.

B. Data exploration

One simulation provides 3 matrices. Each entry of these matrices corresponds to a specific mesh-grid node at a specific time step. Each column of the matrices corresponds to a time step, while each row corresponds to a mesh-grid node. The first and the second matrix store the velocities components along the x- and the y-axis, the third one stores the pressure. An additional fourth matrix saves the parameters used for each simulation. In the following, we will only focus on the velocities. The inputs of the AE are the results of one simulation, which means that one input is given by two matrices (the velocities along x and y). These matrices have 5509 rows, which corresponds to the number of nodes on the mesh-grid of the bifurcation, and 110 columns, which corresponds to the number of time steps. As we ran several simulations, the dataset finally consists of N data points, stacked one below

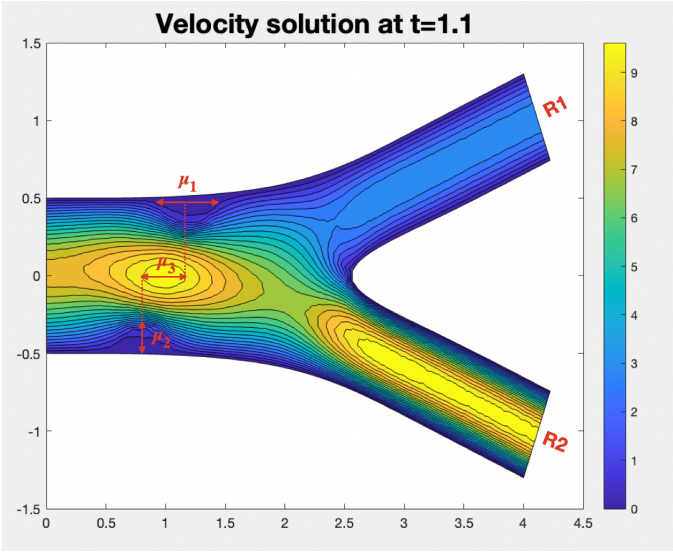

Fig. 1: 2D geometry of the arterial bifurcation. The 5 model parameters are indicated.

the other. Therefore, the two matrices have 110 columns and N*5509 rows. These matrices need to be suitably transformed for the AEs to be able to handle them.

II. Deep learning background

A. Autoencoder and feed-forward autoencoder

AEs are DNNs that seek to learn a compressed representation of the input. They are unsupervised feature extraction algorithms which work by compressing the input into a latent-space representation (encoder) and then reconstructing the output from this representation (decoder). The feed-forward model is the simplest form of AE as information is only processed in one direction. While the data may pass through multiple hidden nodes, it never moves backwards.

B. LSTM Autoencoder

A natural extension of feed-forward AEs is recurrent AEs. Recurrent AEs process a sequence of inputs one element at a time, maintaining in the hidden state an implicit history of previous inputs [3]. Despite providing better data compression, the training of recurrent neural networks is more challenging. The main difficulty is due to the exponential growth or decay of gradients as they are backpropagated through each time step. Indeed, over many time steps, they will either explode or vanish. The problem is typically addressed by using LSTM networks. These networks feature additional paths through which gradients neither vanish nor explode, allowing for efficient back-propagation across multiple time-steps [4]. In this project, we will consider the performances of both a feed-forward and a LSTM AE.

III. Autoencoders for model reduction

A. Sampling

We will not use the solutions of all the mesh points for the following reasons: first, models using a large number of data points with many features are harder to optimize, second, they are also computationally expensive. In this project, one of our goals is to show that by subsampling the data points in time and space, we could still recover a physically informative compression. Moreover, the MATLAB simulations provide solutions on a very fine grid of positions and time steps but it is usually not the case with real-life measurements. Therefore, we will compare the performances of the AEs on different sampling of time and space of the solutions.

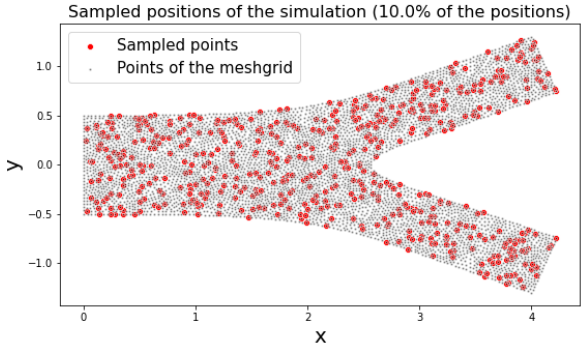

Fig. 2: In this figure, 10% of the positions are sampled. The sampled positions cover mostly all the important parts of the geometry of the artery.

B. Feed-forward autoencoder

1) Flattening in 1D: Each data point consists of 2 matrices of size *(sampled positions)* x *(sampled time steps)*. In order to create a feed-forward AE, the inputs need to be a single vector in one dimension. To do so, let N_s, N_u, N_t being the number of simulations, of spatial points per simulation and of time steps per simulation respectively. Let $\mathbf{u}_x^{(i)}, \mathbf{u}_y^{(i)} \in R^{N_u}$ being the vectors containing the values of the x and y components of the velocity at all the mesh-grid nodes at the time step $i \in \{0, .., N_{t-1}\}$. We can then represent the inputs as in Figure 3. The number of inputs is equal to the number of simulations we ran with different values for the model parameters. Therefore, the input matrix is of dimension $(N_s, (2N_uN_t))$.

$$\mathbf{M_{1D}} = \begin{bmatrix} \mathbf{u}_x^{(0)}\mathbf{u}_y^{(0)} & \mathbf{u}_x^{(1)}\mathbf{u}_y^{(1)} & \dots & \mathbf{u}_x^{(N_t-1)}\mathbf{u}_y^{(N_t-1)} \end{bmatrix}^T$$

Fig. 3: One input data point for the feed-forward autoencoder is a 1D vector with $(2N_uN_t)$ rows.

2) Architecture: The architecture of the feed-forward AE is displayed in Figure 4, each layer is a linear layer. We tried different transformations of the inputs and activation functions for the AE (scaling the input between 0 and 1 and using ReLU and/or sigmoid, normalizing the input with mean 0 and standard deviation 1 and using ReLU, no transformation of the input and using ReLU). The best results were obtained without transforming the inputs and using only linear layers. The input size - which is the same as the output size - depends

on the sampling which is applied on time and space. We tested different latent sizes during the training.

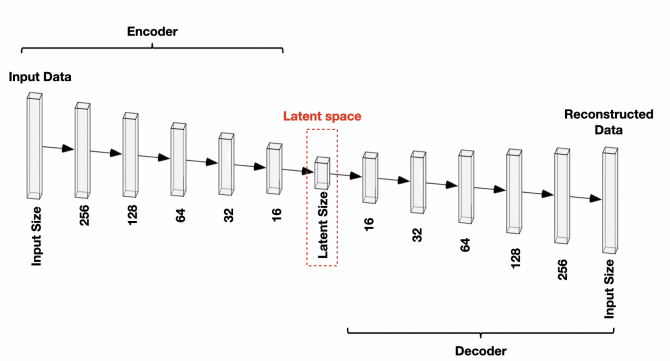

Fig. 4: Network architecture of the feed-forward AE.

3) Training: Adam optimizer was used. We first applied 5-fold cross-validation to tune the learning rate of the feed-forward AE using 5 latent neurons. The following learning rates were tested: 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}. Then, we used 5-fold cross-validation with this optimal learning rate to find the best number of latent neurons. The following number of latent neurons were tested: 3, 4, 5, 6, 7, 8, 9, 10. As displayed in Figures 5 and 6, the best test-error was obtained using a learning rate of 10^{-4} and 10 latent neurons. The metric used to evaluate the AE is a relative error defined as follows:

$$\frac{1}{N}\sum_{n=1}^{N}\frac{\|y_n - y_{pred,n}\|_2^2}{\|y_n\|_2^2}$$

.

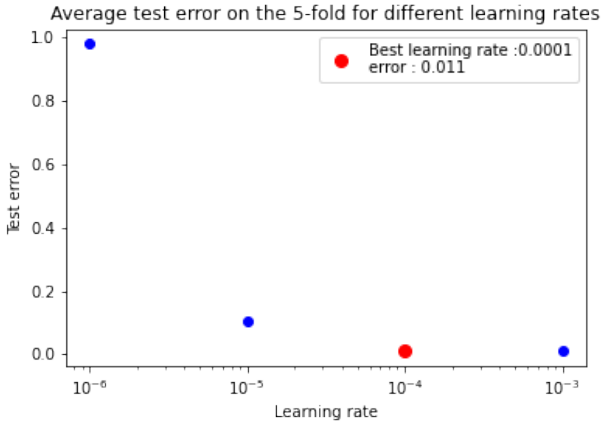

Fig. 5: 5-fold cross-validation for tuning the learning rates of the feed-forward AE. The learning rates tested were: 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}. The best learning rate obtained is 10^{-4} and gives a test error of 0.011.

C. LSTM autoencoder

1) Flattening in 2D: Each input of the LSTM AE needs to be a 2D matrix of dimension $(2N_u, N_t)$. Therefore, each data point has the shape displayed in Figure 7.

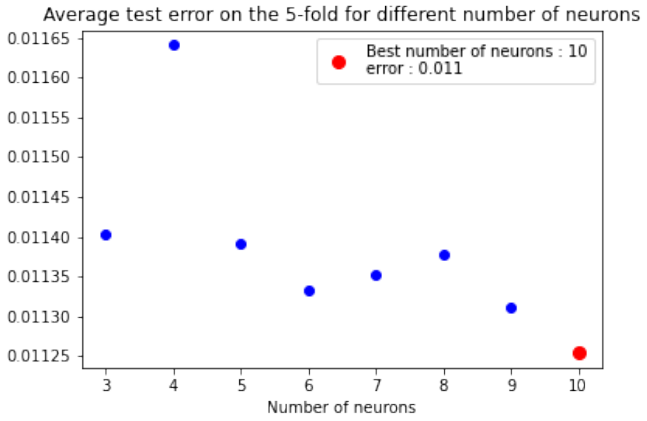

Fig. 6: 5-fold cross-validation for tuning the number of latent neurons of the feed-forward AE. The numbers of latent neurons tested were: 3, 4, 5, 6, 7, 8, 9, 10. The best number obtained is 10 and gives a test error of 0.011.

2) Scaling: Each input of the LSTM AE is scaled so that all values lie in the range $[0, 1]$.

$$\mathbf{M_{2D}} = \begin{bmatrix} \mathbf{u}_x^{(0)} & \mathbf{u}_x^{(1)} & \cdots & \mathbf{u}_x^{(N_t-1)} \\ \mathbf{u}_y^{(0)} & \mathbf{u}_y^{(1)} & \cdots & \mathbf{u}_y^{(N_t-1)} \end{bmatrix}$$

Fig. 7: One input data point for the LSTM autoencoder is a 2D matrix of size $(2N_u, N_t)$.

3) Architecture: The architecture of the LSTM AE is displayed in Figure 8. Again, the input and the output size depend on the sampling and we tested different latent sizes during the training. Each layer of the encoder and of the decoder is a LSTM layer.

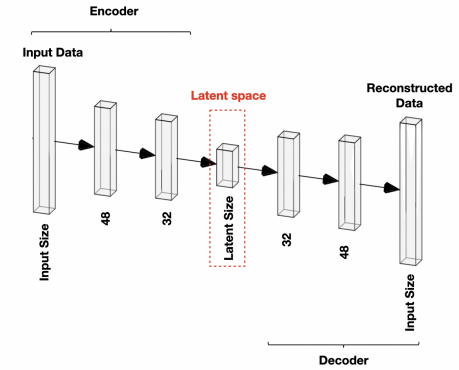

Fig. 8: Network architecture of the LSTM AE.

4) Training: The training of the LSTM AE is based on the same logic as the training for the feed-forward AE except that because stochastic gradient descent optimizer was used, we applied 5-fold cross-validation to tune not only the learning

rate but also the momentum and to find the best number of latent neurons. The following values were tested for the learning rates: 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, and for the momentums: $0.1, 0.5, 0.7, 0.9, 0.99$. For the hidden size, we tested: $3, 5, 6, 8, 10$ neurons in the latent space. The best test error was obtained using a learning rate of 0.1, a momentum of 0.99 and 3 latent neurons.

IV. Results

Tables I and II summarize the test errors obtained using different sampling fractions of space and time.

f_t \ f_s	10%	5%	2%
50%	0.0048	0.0079	0.0085
25%	0.0058	0.0145	0.0075
10%	0.0070	0.0056	0.0072

TABLE I: Table summarizing the relative errors obtained with the feed-forward AE for the different sampling fractions in space (f_s) and time (f_t). The rows represent the percentages used for sampling of the positions, and the columns sampling of time.

f_t \ f_s	10%	5%	2%
50%	0.1656	0.1915	0.1660
25%	0.1654	0.1788	0.1920
10%	0.1918	0.1789	0.1790

TABLE II: Table summarizing the results obtained with the LSTM AE for the different sampling fraction in space (f_s) and time (f_t).

V. Interpretation of the latent neurons in the physical space

We showed that the solutions can be encoded in a low-dimensional space, which corresponds to the latent size of the AEs (10 for the feed-forward, 3 for the LSTM). In a second step, we investigated the relationship between these "abstract parameters" and the 5 initial physical parameters of the haemodynamic problem. For this purpose, we fitted a linear model defined as follows: $y = w * x$ with y being the true parameter from the physical problem and x being the output of the compression done by the encoder. As the quality of the compression depends on the quantity of training we gave to the encoder - i.e. the number of epochs - we looked at the quality of the fit according to the training of the encoder. The ridge regression performed quite well when the number of epochs increased (Figure 9).

Then we asked: how good is our fit according to the sampling of the initial full solution? Indeed, in practice, only a few measurements can be made. Therefore, it would be interesting to recover the general solution with only these few measurements. Table III summarizes the R^2 obtained with ridge regression using different sampling fractions of space and time.

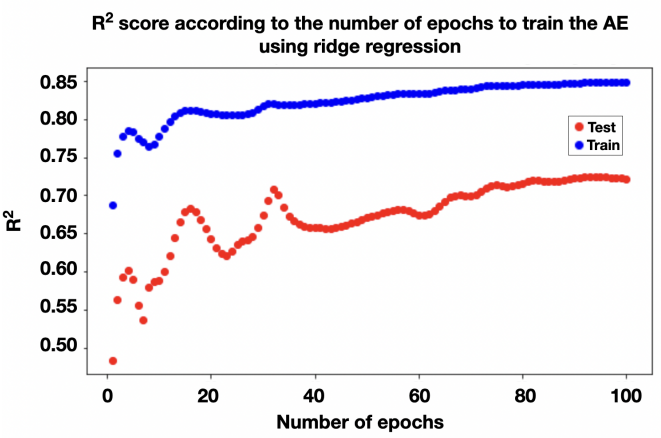

Fig. 9: Quality of the ridge regression fit (R^2) according to the number of epochs.

f_t \ f_s	10%	5%	2%
50%	0.02	0.63	0.33
25%	0.57	0.48	0.60
10%	0.41	0.65	0.51

TABLE III: Table summarizing the ridge regression fit (R^2 for the different sampling fraction in space (f_s) and time (f_t).

VI. Discussion and conclusion

We showed that the solutions can be efficiently compressed in a 10-dimensional subspace using a feed-forward AE. The best test error was 0.0048 and was obtained with 50% and 10% sampling of time and space respectively.

We also explored a more complex model: the LSTM AE. The results were less conclusive compared to the feed-forward AE. The best test error was 0.1654 and was obtained with 25% and 10% sampling of time and space respectively, and a 3-dimensional subspace. Nonetheless, further work would be to optimize deeper the LSTM AEs. Indeed, they are designed for learning the representation of time-series data, which is the case in this project. They might lead to even better performances when optimally designed and with more data to train them.

In fine, we looked at the physical interpretation of the compressed parameters by fitting a ridge regression between the physical and the compressed parameters of the feed-forward AE. The best R^2 obtained was 0.65 and was obtained with 10% and 5% sampling of time and space respectively. This allows us to give an insight about the relationship between the physical and the compressed parameters. Surprisingly, the less sampled dataset - i.e. the closest to the full dataset - did not lead to the best quality of the fit. Indeed, we expected that the closer we are to the original dataset, the better we would be able to precisely recover the 5 physical parameters. Future work would be to further explore this relationship, for instance by optimizing the regression, using non-linear models and considering more data.

REFERENCES

[1] F. J. Gonzalez and M. Balajewicz, "Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems." [Online]. Available: http://arxiv.org/abs/1808.01346

[2] N. Dal Santo, S. Deparis, and L. Pegolotti, "Data driven approximation of parametrized PDEs by reduced basis and neural networks," vol. 416, p. 109550. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999120303247

[3] D. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation."

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," vol. 9, no. 8, pp. 1735–1780. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735