35th European Modeling & Simulation Symposium
20th International Multidisciplinary Modeling & Simulation Multiconference

2724-0029 © 2023 The Authors.
doi: 10.46354/i3m.2023.emss.007

Artificial Neural Networks approach for Digital Twin modelling of an ejector

Ilaria Pietrangeli[1,*], Giovanni Mazzuto[1], Filippo Emanuele Ciarapica[1], Maurizio Bevilacqua[1]

[1] Department of Industrial Engineering and Mathematical Science, Università Politecnica delle Marche, 60131 Ancona, Italy;

[*]Corresponding author. Email address: i.pietrangeli@staff.univpm.it

Abstract

Digital Twin (DT) is an underused tool in the Oil & Gas industry. Today, the behaviour of Oil and Gas plants is realised by the non-real-time analysis software. In contrast, the DT is a framework capable of controlling and managing a plant in real-time by exploiting sensors, virtual spaces, and the continuous connection between real and digital parts. In this paper, the DT of an experimental plant is presented; the DT is based on a model for evaluating the behaviour of an ejector. In contrast to research on DT in the literature, the proposed model is derived from the use of three Artificial Neural Networks (ANNs) and obtains the values of water pressure (ANN1), airflow (ANN3) and water flow (ANN2) at the ejector inlet. The three Multi Layers Perceptron networks, trained on a dataset obtained from the plant, represent the ejector behaviour at 97.85%, 97.79% and 97.94%, the score of each ANN. This modelling approach for DT is currently not widely used but, given the results, is a good alternative to the traditional techniques used.

Keywords: Oil and Gas; Multi Layers Perceptron; Regression; Experimental Plant

1. Introduction & Background

Industry 4.0 and Industrial Internet of Things (IIoT) technologies are changing the plant world dramatically. The "Digital Twin" (DT) concept was illustrated for the first time in 2002 by Grives (Grieves & Vickers, 2016). According to Pierluigi (Pierluigi Sandonnini, 2022), DT is one of the five emerging trends driving technological innovation. In 2012, NASA revisited the concept of DT, which defined it as a multiphysics, probabilistic, high-fidelity, multiscale simulation that promptly reflects the state of a corresponding twin based on historical data, realtime sensor data, and the physical model (Glaessgen & Stargel, 2012). The need to perform monitoring, control, simula-tion, and analysis on plants is driving the development of the Digital Twin of industrial systems. The DT can be seen as a framework that implements the digital counterpart of a real system/plant/phenomenon. The digital and virtual parts are constantly connected and exchange information. Whenever a change is made to the virtual space, that change must also be recorded in real space and vice versa. Based on the literature review conducted by Wanasinghe et al. (Wanasinghe et al., 2020), the manufacturing sector was one of the first (with automotive and aviation industries) to grasp the importance of the Digital Twin as a tool capable of con-verging simulative analysis, visualisation of the risks and key performance indicators of the asset into a single interface. In most cases, the area of application of a DT in the Oil & Gas sector concerns the monitoring and maintenance of systems. In fact, this tool is rare in the Oil & Gas industry. However, evidence of this can be found in the literature. Analysing the documents

in Scopus and accessible with the key " 'digital Twin' AND 'oil gas' ", it's possible to get 31 papers. Of these results, 34.4% are from the "Energy" sector and 17.2 % from the "Engineering" sector. Most of the documents were published between the years 2019-2022. Most articles contain information on DT used for monitoring the oil and gas pipeline network. However, articles highlighting the use of DT to make analyses or predictions about the plants' condition are uncommon. What was found in the literature also reflects the current state of DT use by oil companies. Only a few major energy/oil companies use DT to manage oil facilities. Exxon Mobil and Kongsberg signed an agreement with a DT solution contextualised to dynamic simulation and collaboration capabilities. The Digital Twin software-as-a-service improves decision-making and the decentralisation of knowledge and innovation (Hart Energy, 2021). Among the first to apply DT, British Petroleum (BP) used the technology to solve the problem of monitoring and maintaining oil and gas facilities located in remote areas. BP used DT to improve the reliability of an oil exploration and production facility in Alaska. But the DT could not only enable remote control: it is also enabling predictive maintenance, risk and production time assessment, better collaboration among teams and better financial decision-making; it is working to achieve results in terms of safety (Asokan Ashok, 2021).

In the Oil & Gas sector, system modelling is conducted with several tools such as Aspen HYSYS Hydraulics which performs CFD (Computational Fluid Dynamics) fluid dynamic analysis (Rilby, 2016), or Life Cycle Simulator (Heloin et al., 2013), but also ANSYS or Abaqus (Tao et al., 2022). All these tools work separately, one by one, generally based on mathematical models that require numerous parameters that are difficult to identify. For these reasons, it is easy to see that the Digital Twin is an excellent alternative that allows you to have a single digital platform that contains the digital model of a system in which all the features and rules present in the real part are shown.

This paper will report an example of constructing a simulation-based DT of an experimental oil and gas plant present in the DIISM (Dipartimento di Ingegneria Industriale e Scienze Matematiche) laboratories, Università Politecnica delle Marche. Even though the techniques most used in the Oil & Gas sector are different, the aim is to demonstrate the merits of this technology applied to this sector. This article aims to highlight the potential of such technology for the control, monitoring, analysis and study of Oil & Gas plants. It will be described the plant, the acquisition systems and the model of the principal component of the plant: the ejector.

In contrast to common DTs, to model the behaviour of the ejector, it was used of Artificial Neural Networks (ANN) of the Multi-Layer Perceptron (MLP) Regressor type. This article proposes a model based on ANNs trained on the same dataset. The ANNs structure is realised with 3 networks trained on the data of the same test and they can exchange data to estimate the parameters of interest.

The complete dataset, from which some tests for training are selected, consists of only 5 tests; each presents 17 columns containing the values of the measured variables. Of all the measured variables, those of interest to our 3 networks are the percentage of valve opening downstream of the pump, the pressure of the diffuser cone, the inlet water pressure (ANN1), and the flow rate of water (ANN2) and air entering the system (ANN3).

Section 2 describes the experimental setup, particularly, the ejector, the acquisition system and the dataset. In Section 3, the main difficulties in modelling the ejector are reported. Section 4 describes the network types, hyper parametrisation, training phase, and the specific characteristics of the 3 networks. The previsions are reported in Section 5. Finally, in Section 6 are the conclusions.

2. Materials and Methods

The experimental plant, the ejector and the acquisition system will be described below.

2.1. Plant Description

The plant of interest is an experimental plant located in the laboratories of the DIISM of the Università Politecnica delle Marche. The plant was built in the 1990s to study oil extraction from " inactive" wells. To avoid installing a (very expensive) pump at the bottom of the reservoir, it was studied a system using an ejector and pressure from an "active" well adjacent to the "inactive" one (Figure 1).

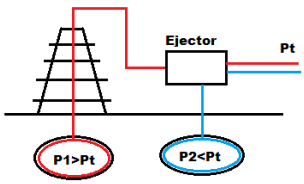

Figure 1. Scheme of extraction plant

For safety, the experimental plant will use water and air instead of oil and gas. The system is shown in Figure 3 and presents an opened water-collection tank; the water is drawn from the opened tank by a pump (in green in Figure 2) that sends it to the ejector; air and water converge in the ejector and mix in a two-phase mixture; the mixture is sent to a vertical tank that acts as a vertical separator: in this way, the air can be sent back to the opened tank, and the air can exit through the appropriate channel. Along the whole system, there are 3 electro-valve (in turquoise), 8 sensors (in light blue) and manual valves (in orange) useful for simulating system faults.

Table 1 shows all the characteristics of the sensors

and Table 2 describes the electro-valves.

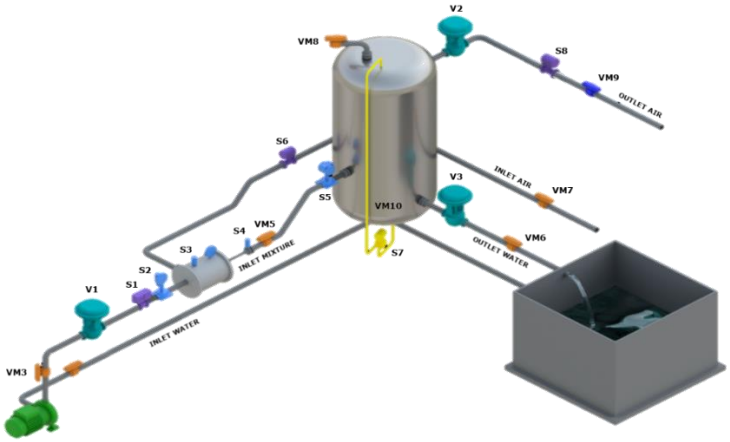

Figure 2. Plant scheme

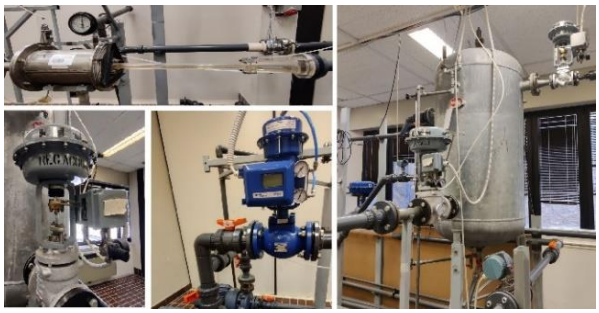

Figure 3. Experimental Plant photos

Table 1. Plant Sensor

ID	Description	UM	Type	Tag
S1	Inlet water pressure	[bar]	OUTPUT	Endress+ Hauser Cerabar M PMP51
S2	Inlet water flow rate	[m³/h]	OUTPUT	Endress+ Hauser Promag W
S3	Ejector pressure	[bar]	OUTPUT	Setra 280E
S4	Mixture pressure in the diffuser	[bar]	OUTPUT	Foxboro 841GM CI1
S5	Tank pressure	[bar]	OUTPUT	Foxboro 841GM-CI1
S6	Inlet air flow rate	[m³/h]	OUTPUT	Foxboro Vortez DN 50
S7	Water level in the tank	[mm]	OUTPUT	Foxboro IDP-10
S8	Air flow rate at the outlet	[m³/h]	OUTPUT	Endress+ Hauser Prowirl 200

Table 2. Plant electro-valves

ID	Description	UM	Type
V1	Valve 1 closure	[%]	INPUT
V2	Valve 2 closure	[%]	INPUT
V3	Valve 3 closure	[%]	INPUT

2.1.1. *Ejector*

An ejector is a hydraulic element with no moving parts that can be used to pressure one fluid, called motive fluid, by taking advantage of a second fluid, called inlet fluid (Dipartimento di Energetica - UNIVPM, 1990). The operation can be explained by referring to the Venturi effect: the pressure of a fluid increases with decreasing velocity. The motive fluid is water, while the inlet fluid is air. According to Bernoulli's theorem, the motive fluid is conveyed inside the nozzle, which has a change in cross-section along its axis: this will increase velocity and decrease pressure. Due to the Venturi effect, the fluid exiting from the nozzle with a certain velocity causes a vacuum inside the mixing chamber that causes the inlet fluid (air) to be drawn into the chamber. The two fluids then come into contact, creating the mixture. Due to the energy imparted to it by the motive fluid, the mix will move with a certain speed along the mixing tube to the divergent cone, known as the diffuser. The gradual increase of the section, due to the diverging cone, causes the slowdown of the fluid: the kinetic energy is transformed into pressure energy. This element has no moving parts: it is very simple, versatile, requires little maintenance, and has easy installation and no sealing problems. To also simplify the interpretation of operation, assumptions are introduced on which the operating model of a liquid-gas ejector is defined:

- isothermal compression of the gas between suction conditions (s-suction) and discharge conditions (l-line), with no loss of gas pressure between the machine inlet and the mixer tube inlet;
- negligible increase in liquid temperature due to gas compression;
- passage through the mixing tube produces a homogeneous mixture of the two liquid-gas phases, with no flow between them;
- the influences of the vapour pressure of the liquid phase and the viscosity of the gas and liquid are considered negligible.

Given the following conditions, it will be possible to write the following governing equations:

- Efflux through nozzle:

$$p_d - p_o = \frac{1}{2}\rho_l V_{lo}^2 (1 + K_{nz}) \quad (1)$$

- Flow in the mixing tube:

$$p_t - p_o = \frac{1}{2}\rho_l V_{lo}^2 [2b + 2\Upsilon\Phi_o^2 \frac{b^2}{1-b} - (2 + K_{th})b^2(1 + \Phi_t)(1 + \Upsilon\Phi_o)] \quad (2)$$

- Flow in the diffuser tube:

$$p_l - p_t = Z(1 + \Upsilon\Phi_o)[\, b^2(1 + \Phi_t)^2 - b^2a^2(1 + \Phi_l)^2 - K_{dl}\, b^2\, (1 + \Phi_t)] - p_o\, \Phi_o\, \ln\frac{p_l}{p_t} \quad (3)$$

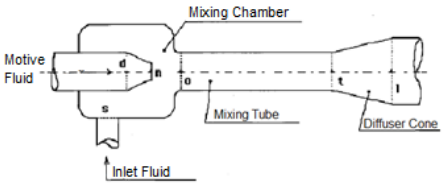

Figure 4. Ejector scheme

Figure 4 shows the simplified diagram in which the various sections referred to in the previous equations are highlighted. Table 3 and Table 4 explain the indices and parameters in the equations.

Table 3. Equations Index

Index	
d	Motor fluid adduction section - drive
n	Motor fluid outflow section
o	Mixer tube inlet section
t	Mixer tube outlet section
l	Diffuser end section - line
s	Suction fluid adduction section - suction
nz	Nozzle
di	Diffuser
th	Mixer tube (throat)
en	Suction box inlet

Table 4. Equations Parameters

Parameters	
a	Ratio of At/Al diffuser sections
b	Ratio of the nozzle and mixing sections An/At
K	Coefficient of friction loss
p	Fluid pressure [Mpa]
Q	Fluid flow rate [m³/s]
V	Fluid velocity [m/s]
ρ	Fluid density [kg/m³]
Φ	Ratio of the volumetric flow rates of the fluid sucked in and the motor fluid (*)
Υ	Ratio of the densities of the sucked and motor fluids in the section or the machine (**)
Φ_o	Q_{2o} / Q_1
Φ_t	Q_{2t} / Q_1
Φ_s	Q_{2s} / Q_{1o}
Φ_{eq}	$\Upsilon\ \Phi_o$
Υ	ρ_{2o} / ρ_1
Υ_{eq}	ρ_{L2} / ρ_1
η	Ejector machine efficiency
η'	Yield of two-phase mixture pumping system

The ejector is the most mathematically complex element to model, especially for the two fluids that affect it (air and water, in our case). Of course, the equations are known and have already been studied, but very often, it is the coefficients that characterise them that are difficult to identify.

For this purpose, very often, when the mathematics of the system is too complex to be solved, it is possible to use machine learning algorithms or ANNs that, with training done on experimental data of the system, provide us with an optimal model of the object of interest.

2.2. Dataset

The dataset is acquired by exploiting the sensors connected to a device called Revolution Pi (RevPi) which has an AIO-type module to which the plant's sensor pins are connected. The RevPi communicates with the server via WebSocket. Web sockets allow browser-server interaction and enable the implementation of applications that provide real-time data without the client continuously requesting information from the server. The server receives and resends the data to all other servers, including the DT interface accessed from a PC connected to the same internet line as RevPi. In this interface, there are tachometer graphs and animation in the interface showing changes in the levels, pressures and flow rates of water and air within the system; there is a button on the left side of the interface to record and store in a ".csv" file the strings read during the time frame. This command was useful for acquiring the data for training our neural network.

To obtain a useful dataset, it is possible to set parameters like water pressure at the plant inlet, tank level and tank pressure (Table 5).

- P_{IN} [bar]: Water pressure at the tank inlet
- L_S [mm]: Tank level
- P_S [bar]: Tank pressure

Table 5. Dataset

P_{IN} [bar]	L_S [mm]	P_S [bar]
3	0	0
3	0	1.3

3	250	1.5
3	350	1.5
3	450	1.3

All tests contain data from sensors throughout the plant and are recorded so that there are approximately 1600 acquisitions for each. The data are stored within each ".csv" file. The first line of each file is the "header" which has all the abbreviated names of the respective sampled variables. The measured variables are shown in Table 6.

Table 6. Header Dataset

Name	Description	Unit
PwatIN	Inlet water pressure	[bar]
QwatIN	Inlet water flow rate	[m3/h]
Peje	Ejector pressure	[bar]
Pdiff	Diffuser mixture pressure	[bar]
Ps	Tank pressure	[bar]
QairIN	Inlet air flow rate	[m3/h]
Ls	Tank water level	[mm]
QairOUT	Outlet air flow rate	[m3/h]
%Vwat	Valve 1 closure	[%]
%Vair	Valve 2 closure	[%]
%VwatOUT	Valve 3 closure	[%]
Test	Identification Number of the Test	-

3. Modelling difficulties

As already pointed out in Section 1, today, the modelling of Oil & Gas plants is almost exclusively carried out with software (also not open sources) that mainly performs fluid-dynamic or thermodynamic analyses. In addition, control, analysis, and anomaly detection are often not carried out by a single technology but by multiple software and devices that must communicate and collaborate, with all its attendant difficulties. The best alternative in this context proposed in our article is the construction of a DT of the system.

In the case proposed in this article, the model for a DT of the most complex element of the experimental plant is realised: the ejector model. The complexity of the ejector's characteristic equations and the difficulty in identifying the parameters that characterise them directed the construction of the ejector model towards using neural networks.

In addition, as already mentioned in Section 2.1, water and air are handled in the ejector of the experimental plant instead of oil and gas for safety reasons. The presence of a compressible fluid such as air causes greater difficulty in applying the equations (1)-(2)-(3), which can be used under stringent constraints or in finding a mathematical model. For this reason, it was also decided to use ANNs to model the ejector. Even ANNs cannot perfectly reproduce the behaviour of the ejector and the air inside it, but they are a quick and easy method to extract the first results accurately.

4. Artificial Neural Networks model

As mentioned in Section 2.1.1, the equations that characterise the behaviour of the ejector are very complex and difficult to use for real-time analysis. For this reason, it is possible to model the behaviour of this component through ANNs. In particular, three networks were constructed to derive three fundamental parameters of the ejector: inlet water pressure (PwatIN), inlet water flow rate (QwatIN) and air flow rate at the ejector inlet (QairIN). The three networks were implemented in Python and have a similar structure.

4.1. General description

A system with 3 Multiple Input Single Output (MISO) ANNs has been constructed to allow us to derive the variables of interest. All 3 networks exploit the MLP Regressor function, train on the same dataset and have similar functionalities. The general operation is explained below.

First, a function was created to load the dataset: of the 5 tests in Table **5**, only the first is loaded to train and test the networks (the remainder will be used to validate them).

In each ANNs, the single selected test is then normalised according to the values in Table 7 with the *MinMaxScaler()* function. Normalisation is necessary to reduce the computational load and better predict the outcome. The dataset is then divided into the *X* and *Y* matrices containing the values of the independent and dependent variables (to be estimated).

Table 7. Normalisation data

%Vwat	Pdiff	PwatIN	QwatIN	QairIN
0	0	0	0	0
100	10	10	100	100

The data are split into "train data" and "test data" with the *train_test_split()* function in which the argument is passed the matrices X and Y, the *test_size* and the *random_state*. The *test_size* is a value between 0-1 corresponding to the percentage of data allocated for the test. The *random_state* is a number corresponding to the seed of the random sequence with which the train and test data will be divided.

Hyper parameterisation with a search grid is then performed, and the network function from the *scikit-leran* library, *MLPRegressor*, is loaded, to which the hyper- parameterisation parameters are entered as an argument.

4.2. MLPRegressor and hyper-parameterisation

MLPRegressor is a neural network from the open-source library Scikit-learn that can be used to perform regression. Regression in machine learning can be seen as a mapping from one space to another, each space having a different number of dimensions. "MLP" refers to a particular kind of neural network called a Multi-

Layer Perceptron. The ANN is trained using supervised learning to predict output data points based on input data points by supplying input and output data as data sets (*Scikit-Learn Documentation*).

Some of the fundamental parameters that the MLPRegressor function takes as arguments are reported in Table 8.

Table 8. MLPRegressor parameters

Name	Description	Choices
hidden_layer_sizes	The element represents the number of neurons in the hidden layer	Natural Number In the case reported in this article, a natural number in the range (0, 501) with step 5 was chosen.
activation	The activation function for the hidden layer	- 'identity', returns f(x) = x - 'logistic', returns f(x) = 1 / (1 + exp(-x)). - 'tanh', returns f(x) = tanh(x). - 'relu', returns f(x) = max(0, x)
solver	The solver for weight optimization	- 'lbfgs' is an optimiser in the family of quasi-Newton methods. - 'sgd' refers to stochastic gradient descent. - 'adam' refers to a stochastic gradient-based optimiser proposed by Kingma, Diederik, and Jimmy Ba
learning_rate	Learning rate schedule for weight updates.	- 'constant' is a constant learning rate given by 'learning_rate_init'. - 'invscaling' gradually decreases the learning rate learning_rate_ at each time step 't' using an inverse scaling exponent of 'power_t'. effective_learning_rate = learning_rate_init / pow(t, power_t) - 'adaptive' keeps the learning rate constant to 'learning_rate_init' as long as training loss decreases.
max_iter	The maximum number of iterations. The solver iterates until convergence or this number of iterations	Natural number. In the case reported in this article, 500 was chosen as max_iter according to the documentation on Scikit-Learn

(Sklearn.Neural_network.MLPRegressor — Scikit-Learn Documentation) and (Scikit-Learn Documentation - MLPRegression Example)

Among the functions available in Scikit-learn, this regression function was chosen because there could be a linear relationship between the input and output parameters of each network: in the first network, the values of %Vwat and Pdiff are passed in to obtain the trend of the variable PwatIN; the second network takes the value of PwatIN as input to obtain the value of QwatIN; the third network takes the value of Pdiff and PwatIN as input to obtain the value of QairIN. It was decided to use only one hidden layer in each network because the relationship between the different parameters is linear.

A hyper-parameterisation with a search grid is used to choose the best parameters to pass as an argument to the MLPRegressor function. The GridSearchCV function from sklearn.model_selection is imported to perform the hyper-parameterisation. The arguments of this function are:

- estimator: the object to be optimised;
- parameters: which must be in the form of a dictionary where each key represents a variable assignable to the estimator and each list accessible with the keys contains the admissible parameters (Table 8);
- scoring: a strategy for evaluating model performance with cross-validation on the test set. The neg_mean_squared_error was chosen from the eligible Regression strategies.
- cv: cross-validation splitting strategy; it was chosen cv=3;
- n_jobs: number of jobs that can run in parallel. It was chosen -1 to use all processors;
- return_train_score: get insights on how different parameter settings impact the overfitting/underfitting trade-off and, for this reason, was set on *True*.

The GridSearchCV function tests all possible combinations of the number of neurons, activation functions, solvers, etc... and finds the best combination that can then be used to obtain the best score from each network.

The following sections will specify each network's parameters obtained by hyper-parameterisation (Sklearn.Model_selection.GridSearchCV — Scikit-Learn Documentation). As anticipated, all 3 networks were trained with the same test: the P_{IN}:3 bar, L_S :0 mm, and P_S:0 bar test (which will be referred to as *3_0_0* for convenience).

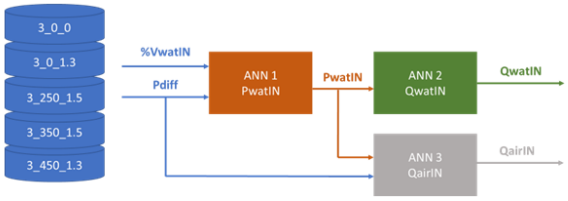

Figure 5. ANNs working structure

After the training phase, the 3 networks work together as shown in Figure 5: the first network takes %Vwat and Pdiff data as input from the original dataset to estimate PwatIN. The PwatIN evaluated by ANN1 is given to ANN2 to estimate QwatIN. Finally, ANN3 takes the Pdiff values from the original dataset and the PwatIN values estimated by ANN1 to predict QairIN.

4.3. ANN1: PwatIN

The first step in building the first network was to import the training dataset and train the ANN with it. For this purpose, it was implemented a code (schematised in Figure 6) that allows to:

1. load the test of interest in ".csv" format;

2. select only the columns of interest for the first network i.e. %Vwat, Pdiff, %PwatIN and save them as dataset_interest;
3. save also the columns %Vwat, Pdiff, %PwatIN and Test like dataset_test;
4. normalise the dataset_interest with the MinMaxScaler() function and the parameters of %Vwat, Pdiff, %PwatIN in Table 7 and save the result as dataset_interest_norm.

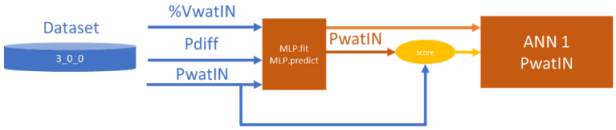

Figure 6. ANN1: train

After being run, this part of code returns dataset_interest, dataset_test, dataset_interest_norm.

It was necessary to divide dataset_ineterst into dependent variable Y and independent variable X to which the measured values of PwatIN and the values of Pdiff, %VwatIN, respectively, had to be associated. To do this, it was used the function train_test_split (illustrated in Section 4.1). The test_size chosen is 0.30, and the random_state is 0 (to achieve reproducibility).

Then, the hyper-parameterisation was run, which resulted in 470 neurons, activation function 'relu', solver 'lbfgs', learning_rate 'constant', having chosen as random_state 22 and max_iter 500. The values were fed back into the MLPRegressor function to build the MLP model. The 'fit' method was then used to train the model on the x_train and y_train data.

With the 'score' method, it is possible to calculate the network score, i.e. the coefficient of determination of the prediction. The score is 0.9785, corresponding to 97.85% correspondence between predicted and measured data. With the 'predict' method and the values of X_train and X_test, it was possible to calculate the predictions of the train and test data. The mean square errors were calculated, which were found to be 0.01997 for the train and 0.02243 for the test. These errors are normalised to the PwatIN values in Table 7.

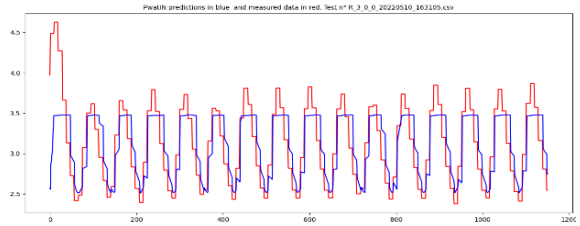

Figure 7. Prediction PwatIN - train dataset

Once the MLP model has been trained and tested and initial predictions have been made with train and test data, it is possible to use MLP to predict PwatIN values from Pdiff and %Vwat data of the entire dataset.

Columns of Pdiff, %Vwat and PwatIn from each test are then loaded, normalised and the MLP.predict method is applied to make the predictions (Figure 7). The errors committed in prediction by each trial are then calculated.

PwatIN forecast results are saved in 'csv' files with data from QairIN, QwatIN and Pdiff to be passed on to ANN2 and ANN3.

4.4. ANN2: QwatIN

The second network works similarly to the first, but differently, the PwatIN and QwatIN columns are selected in the training phase (Figure 8). As in the previous network, the data are normalised (with the corresponding values in Table 7), divided into the X and Y matrices, and re-subdivided into train and test data with a test_size of 0.3. Hyper-parameterisation results in 175 neurons, the activation function 'relu', the solver 'adam' and 'constant' as the learning_rate. The calculated score for this network is 0.9794 (97.94%), while the train and test normalised errors are 0.00307 and 0.00287, respectively.

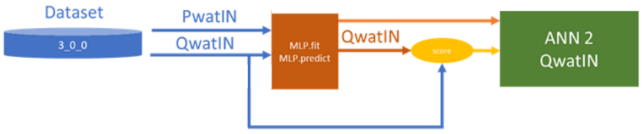

Figure 8. ANN2: train

Using the 'predict' method, it is possible to estimate the trend of the test used in the training phase: tests "3_0_0".

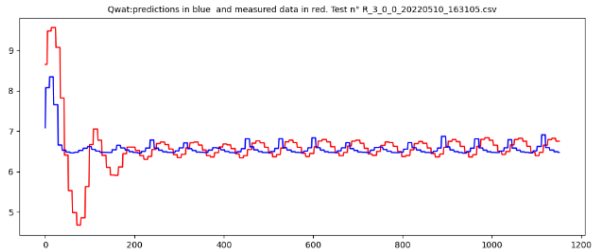

Figure 9. Prediction QwatIN - train dataset

The error in the prediction (Figure 9) of the test is 0.4612 m^3/h.

4.5. ANN3: QairIN

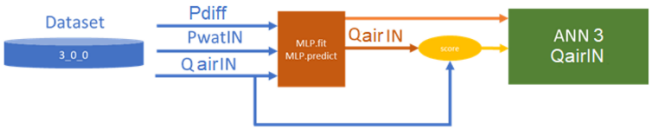

Figure 10. ANN3: train

Like the first two ANNs, the third ANN uses the data-loading function to obtain the PwatIN, Pdiff and QairIN values of the training trial (Figure 10). Again, once the 3 columns of data are selected, they can be divided into

the X and Y matrices: to the X matrix, it is assigned the values of PwatIN and Pdiff, while to the Y matrix, it is assigned the vector of the variable QairIN. Then, it is possible to split train and test data with test_size of 0.3.

The MLPRegressor function is then used in which the following are given as arguments: number of neurons 185, activation function 'relu', solver 'lbfgs' and learning rate 'constant'. These parameters were obtained by hyper parametrisation. The other parameters are the same as for the last networks.

The calculated score for this network is 0.9779 (97.79%), while the train and test normalised errors are 0.02636 and 0.02637, respectively.

Again, it is possible to use our network to initially redraw the trend of the test used in the training phase (Figure 11). The average error over the whole trend is 4.028 m³/h.

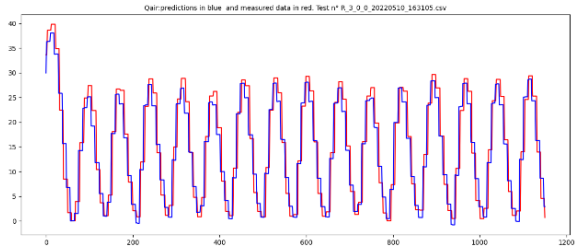

Figure 11. Prediction QairIN - train dataset – normalised

5. Results and Discussion

As mentioned above, once the three networks have been trained, it is possible to predict the performance of each test. The dataset is therefore formed: it is possible to concatenate all tests and normalise them according to the values in Table 7.

Now, by selecting only the columns for Pdiff and %VwatIN values, it is possible to predict PwatIN values with the 'predict' method of regressor, obtained from the first ANN. This prediction yields the trends shown in Figure 12 with the errors shown in Table 9.

Table 9. ANN1 prediction errors

Test name	Real error [bar]
3_0_0	0.03057
3_0_1.3	0.02456
3_250_1.5	0.04541
3_350_1.5	0.03177
3_450_1.3	0.04570

At the end of the training, testing and prediction of the first network, the predicted values of PwatIN of the 5 tests are saved to two csv files: in the first file in addition to PwatIN the values of QwatIN of the 5 tests are saved (*file1*), in the second file in addition to PwatIN the values of Qair and Pdiff of the 5 tests are saved(*file2*). The next 2 ANNs will also call up these files to estimate those data.

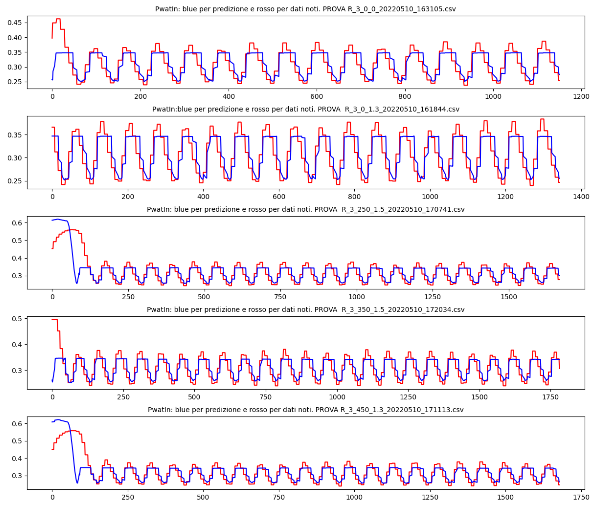

Figure 12 . ANN 1 predictions-original dataset -normalised

ANN2 predictions are made, first of all, from the original PwatIN data of the dataset (trends in Figure 13; errors in Table 10) and then from the dataset saved in *file1*. The QwatIN values of *file1* will be useful for error evaluation.

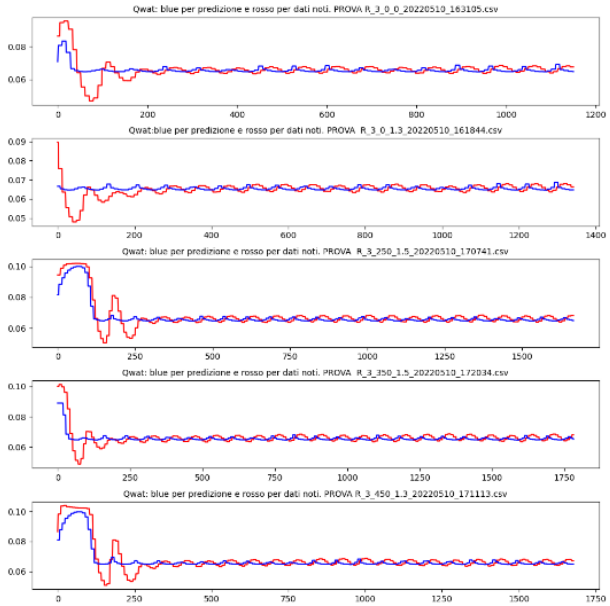

Figure 13. ANN2 predictions – original dataset -normalised

Table 10 . ANN2 prediction errors– original dataset

Test name	Normalised error	Real error [m³/h]
3_0_0	0.004612	0.4612
3_0_1.3	0.003597	0.3597
3_250_1.5	0.003859	0.3859
3_350_1.5	0.003825	0.3825
3_450_1.3	0.004278	0.4278

The results of ANN2, obtained with the PwatIN values from ANN1, are the following (errors in Table 11; trends in Figure 14).

Table 11. ANN2 prediction errors– ANN1 dataset (*file1*)

Test name	Real error [m³/h]
3_0_0	0.5869
3_0_1.3	0.3413
3_250_1.5	0.6695
3_350_1.5	0.5461
3_450_1.3	0.6898

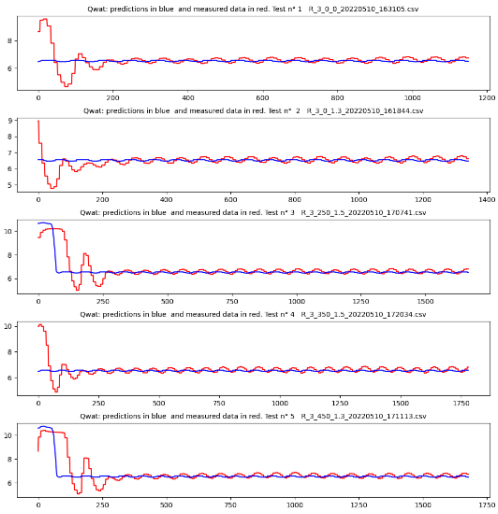

Figure 14. ANN2 predictions – ANN1 dataset (file1)

Also for ANN3, the predicted QairIN values of the 5 tests are given from both, the data of the original dataset (trends in **Figure 15**; errors in **Table 12**) and the data obtained from ANN1 (trends in **Figure 16**; errors in **Table 13**).

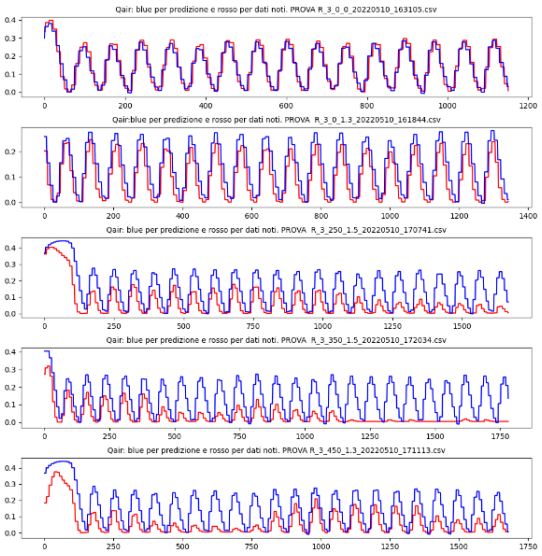

Figure 15. ANN3 predictions – original dataset -normalised

Table 12. ANN3 prediction errors– original dataset

Test name	Normalised error	Real error [m³/h]
3_0_0	0.040281	4.0281
3_0_1.3	0.058875	5.8875
3_250_1.5	0.107093	10.7093
3_350_1.5	0.127389	12.7389
3_450_1.3	0.1138191	11.3819

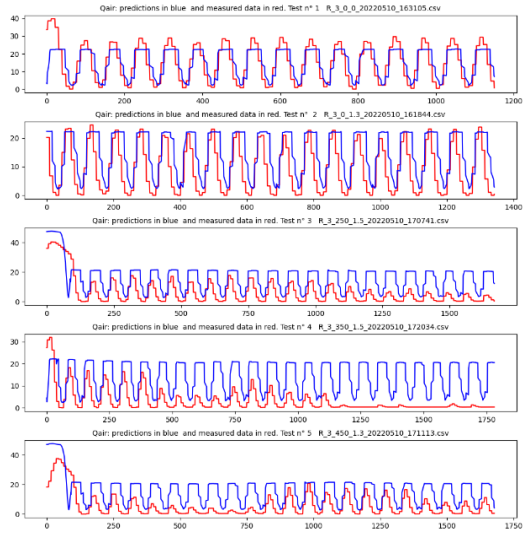

Figure 16. ANN3 predictions – ANN1 dataset (*file2*)

Table 13. ANN3 prediction errors– ANN1 dataset (*file2*)

Test name	Real error [m³/h]
3_0_0	6.0405
3_0_1.3	7.8256
3_250_1.5	11.1661
3_350_1.5	12.8702
3_450_1.3	11.7109

As can be seen from the results, the first ANN1 and ANN2 perform good predictions. In particular, the errors are small even when ANN2 processes data derived from ANN1. Obviously, these last results are negatively affected by the error that comes from ANN1. In fact, since ANN1 is not 100% accurate, it returns data with an error that is accentuated by ANN2, which also does not perform 100%.

Even though this inconvenience arises in the interaction between the networks, it was decided to operate in this way to use the two networks also separately (in the case where it is necessary to operate with only one network). In fact, if the input parameters are already known, each network can calculate the respective results, regardless of the operation of the other ANNs.

ANN3 is the worst-performing ANN compared to the other ANNs. The fact that the errors are larger than those of the others is not surprising: capturing air behaviour in a system is extremely difficult due to the compressibility characteristics of this fluid. In addition, the air is a fluid highly susceptible to environmental conditions outside the system: it is strongly affected by ambient temperature and atmospheric pressure, for example. In addition to the room temperature, it must be considered that the air

heats up during plant operation and causes more difficulties. For these reasons, the behaviour of air is captured by ANN3 with larger errors than in the case of water flow or water pressure.

6. Conclusions

The aim of this paper is highlighting how a DT could be a good alternative to monitor, analyse and simulate the behaviour of an Oil & Gas plant without the use of dedicated software or a non-open sources system. The main advantage of the use DT is that all these studies can be done in just one digital platform. In this article, it's reported the model for a DT of an ejector: a simple component that can be studied only with complex equations. To avoid this difficulty, an ANNs system is proposed to predict the behaviour of water pressure, water flow and air flow inlet the ejector from the value of diffuser pressure, and the percentage of closure of Valve 1.

Even if the ANNs system doesn't perform the ejector behaviour perfectly, prediction errors are limited and justified by the characteristics (compressibility, in particular) of the fluids processed. The major limitations of our work relate to the choice of ANNs type, the limited size of the dataset and the difficulties involved in using two fluids. Therefore, as future developments, it is possible to: develop the DT model of the ejector by exploiting Multiple Input-Multiple Output ANNs to estimate all variables of interest in one shot; improve the process of hyper parameterisation; construct a larger dataset with more parameters: a temperature sensor could be included to track the ambient temperature in which the system operates and correlate it to the airflow.

Funding

This paper is supported by European Union's Horizon Europe research and innovation programme under grant agreement No 101057294, project AIDEAS (AI-Driven industrial Equipment product life cycle boosting Agility, Sustainability and resilience).

This paper is also supported by European Union's Horizon Europe research and innovation programme under grant agreement No 101092043, project AGILEHAND (Smart Grading, Handling and Packaging Solutions for Soft and Deformable Products in Agile and Reconfigurable Lines).

References

Asokan Ashok. (2021). *Digital Twins in 2021: 15 Amazing Examples*. https://unfoldlabs.medium.com/digital-twins-in-2021-15-amazing-examples-3e492d4852f5

Dipartimento di Energetica -UNIVPM. (1990). *STUDIO DI EIETTORI BIFASE PER L'INDUSTRIA PETROLIFERA*.

Glaessgen, E. H., & Stargel, D. S. (2012). The digital twin paradigm for future NASA and U.S. air force vehicles. *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 2012*.

Grieves, M., & Vickers, J. (2016). *Origins of the Digital Twin Concept*. https://doi.org/10.13140/RG.2.2.26367.61609

Hart Energy. (2021). *Exxon Mobil to Deploy Kongsberg Digital Twin Software*. https://www.hartenergy.com/news/exxon-mobil-deploy-kongsberg-digital-twin-software-194928

Heloin, N. R., Haarklou, E., Thiabaud, P., Jalby, G., & Schiefloe, T. (2013). Peregrino FPSO - Operation Challenges Overcome by the Use of Dynamic Simulation from Studies to Training. In *OTC Brasil* (p. OTC-24329-MS). https://doi.org/10.4043/24329-MS

Pierluigi Sandonnini. (2022). *Report Gartner: 5 tendenze chiave per il futuro dell'AI - AI4Business*. https://www.ai4business.it/news/report-gartner-5-tendenze-chiave-per-il-futuro-dellai/

Rilby, E. (2016). *Dynamic Simulations of an Oil and Gas Well Stream*.

scikit-learn documentation - MLPRegression example. (n.d.). Retrieved March 10, 2023, from https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_partial_dependence_visualization_api.html#sphx-glr-auto-examples-miscellaneous-plot-partial-dependence-visualization-api-py

Scikit-learn documentation. (n.d.). Retrieved March 9, 2023, from https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron

sklearn.model_selection.GridSearchCV — scikit-learn documentation. (n.d.). Retrieved March 9, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

sklearn.neural_network.MLPRegressor — scikit-learn documentation. (n.d.). Retrieved March 9, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

Tao, F., Xiao, B., Qi, Q., Cheng, J., & Ji, P. (2022). Digital twin modeling. *Journal of Manufacturing Systems*, *64*, 372–389. https://doi.org/10.1016/J.JMSY.2022.06.015

Wanasinghe, T. R., Wroblewski, L., Petersen, B. K., Gosine, R. G., James, L. A., de Silva, O., Mann, G. K. I., & Warrian, P. J. (2020). Digital Twin for the Oil and Gas Industry: Overview, Research Trends, Opportunities, and Challenges. *IEEE Access*, *8*, 104175–104197. https://doi.org/10.1109/ACCESS.2020.2998723