RESEARCH ARTICLE

Facies Classification using Artificial Neural Network

Jacopo Montefusco (✉)[1],
School of Engineering University of Aberdeen Fraser Noble Building Kings College Old Aberdeen United Kingdom AB24 3UE

Abstract In this article the machine learning Artificial Neural Network (ANN) method is applied to automate the facies classification process. The ANN receives a dataset composed by seven wells of the Hugoton and Panoma fields in Kansas US which includes well logging and nine recognized facies data, this information is given as input to the ANN to create a model in order to predict the facies classes simulating a real facies classification study. First, six of the seven wells included in the dataset were used to build the model using a supervised training process while the remaining well was used as a test well to evaluate the model performance. Subsequently, the dataset was changed to equalize the unbalanced facies class samples trying to enhance the model performance and finally, the influence of the dataset size on the network performance was explored by applying a sample complexity study. The results achieved confirm that the ANN is a reliable tool in facies classification, however it also brings the main drawback of all the machine learning techniques which is that the developed models are highly error-prone due to the lack of technical information inside them.

Keywords Machine Learning, Neural Networks, Facies Classification, Lithofacies

E-mail: montefuscojacopo@gmail.com

1 Introduction

During a petroleum exploration the goal is to identify whether a well is commercially valuable or not, and the process used to determine that is called Formation Evaluation (FE). During a FE the ability of a borehole to produce petroleum is determined; this includes the evaluation of potential reservoir rocks, estimating the petroleum reserves, and measuring the reservoir properties to design the better engineering solution during all the field life (Hill, 2017). Petrophysicists and Engineers who are in charge in a FE have two main methods to study the petrophysical properties: core analysis and well logging. Well loggings are conducted using a string of measurements tools lowered in the borehole. The tools contain sensors used to detect and record the petrophysical properties of the rock layers (Fanchi, 2002). Otherwise by performing core analysis small borehole samples (cores) are extracted from the well and subsequently the physical and chemical property are investigated in the laboratory. Between these two methods, core analysis is definitively more reliable because it is a direct measurement of the reservoir properties instead, well logging data must be interpreted to characterize the reservoir. Nevertheless, core analysis has some drawbacks in fact almost 70% of the core analysis data obtained are useless due to: poor core condition, schedule deficiency, insufficient laboratory practice, and cores do not well represent the rock layer from where they have been extracted (McPhee et al., 2015).

Moreover, well logging is cheaper than taking cores so the industry always tries to focus on this technology and a small number of cores are usually taken only to calibrate the logging data (Rao and Knight, 2017). One fundamental part of the FE is the lithofacies identification. Facies (i.e., lithofacies) identification dwell in allocate a rock class along the well depth. This process is crucial because by understanding the lithology the reservoir can be easier modelled because the rock properties are all dependent by that (Hyne, 1991). Certainly, core analysis is the more obvious method to carry out facies classification however, as already said cores extraction is an expensive operation thus an indirect process as well logging is essential. The standard method includes the interpretation of the well logging by experienced Petrophysicists. This process brings a series of disadvantages such as: is very time consuming because is made by humans; high costly experts are needed; personal biases of the experts are expected to be included in the interpretation; and it is not scalable so cannot be applied for high data dimension (Halotel et al., 2020). Consequently, alternatives methods based on statistical approach have been developed to face the facies classification from well logging (Ma et al., 2019). Nowadays in the era of the big data a new technique is gaining interest in the geophysical community: machine learning. As a matter of fact, the problem of facies classification is suitable for the supervised classification which is a common task managed by machine learning algorithms (Bestagini et al., 2017). In this context, in 2016 the Society of Economic Geologists (SEG) organized a contest where the participants had to use machine learning algorithms to solve a facies classification problem (Hall, 2016). The accuracy obtained by the best team was almost 63% (Hall and Hall., 2017).

This work aims to study the feasibility of a machine learning method called Artificial Neural Network (ANN) in lithofacies classification problem based on the dataset presented in the mentioned contest. The model will take the recorded well logging data as input and will output the predicted lithofacies. The project presents the following principal objectives.

1. To build different ANN models based on diverse architectures by using part of the dataset used in the mentioned contest and analysing their performance.
2. To change the dataset by balancing the facies classes to analyse whether this technique increases the model's performance.
3. To determine the influence of the size of the dataset on the model's performance

2 Materials and methods

2.1 Dataset presentation

The dataset used in this work it's a portion of the one proposed by Dubois et al, (2007) and comes from seven of the more than 12,000 wells of the Hugoton and Panoma fields situated in southwest Kansas and northwest Oklahoma, which are together the first gas producing area in the US. In Fig. 1 the distribution of the wells location is shown. The seven mentioned wells were subjected to both core analysis and well logging thus the respectively formation lithologies were studied and assigned using the manual traditional technique (Halotel et al., 2020). Nine facies were recognized as listed in Table 1 (Hall, 2016). A specific mention must be done for facies 6, 7, 8, 9 which are the most important for hydrocarbons production because they form the major part of the pay zone. The presented facies classification is made aiming to obtain the minimum number of facies to correctly characterize the reservoir and to achieve this goal the facies have been distinguished by the petrophysical properties of the formation layers measured both directly and indirectly through core analysis and well logging. Besides the rigid classification, clearly the nine facies show similarities which are underlined by Dubois et al., (2007) and reported as adjacent facies in Table 1. Together all the seven wells include 3164 facies samples which are not equally distributed along the wells as is shown in Fig. 2(a). By analysing both Fig. 1 and Fig. 2(b) a lateral trend of the facies is shown in fact, in the north-west of the field the non-marine facies prevail instead marine facies are predominant in the south-east. This behaviour is due to the changing in the depositional environment during the geological periods and represents an asymmetry in the dataset which will add another level of complexity in the classification problem (Halotel et al., 2020).

As already said beside core analysis, well logs of the seven wells are included in the dataset. The digital logs are recorded with an interval of 0.5 feet along a portion of the total depth of each well and together count 3164 samples, and this is the reason why the facies samples count the same value. There are 5 conventional well logs included in the dataset, in particular:

- The gamma ray log (GR).
- The deep induction resistivity log in logarithmic form (ILDlog10).
- The difference between neutron porosity and density porosity by using the neutron log (ΔPHI).
- The average value from neutron porosity and density porosity by using the neutron log (PHIND).
- The photoelectric log (PE).

In Fig. 2(b) the cross-plots between the well logs as a function of the facies are shown. The figures clearly evidence the non-linear relationships between the logs which results on the complexity of the dataset. For clearness, the plots placed on the diagonal of Fig. 2(b) are the cross-plot of the same logs thus are represented as histograms.

The complete dataset made of 3164 samples and 7 wells composes a 3164 x 7 matrix which will be named D^{3164x7}. Each row contains 7 columns: 6 scalar wireline logs values (5 well logs and the depth vector), and 1 facies target. In machine learning jargon the first 6 column of the matrix *D* forms the feature vector F^{3164x6} which represents the input data that are associated with the facies target represented in the last column of the vector *D* called class vector C^{3164x1}.

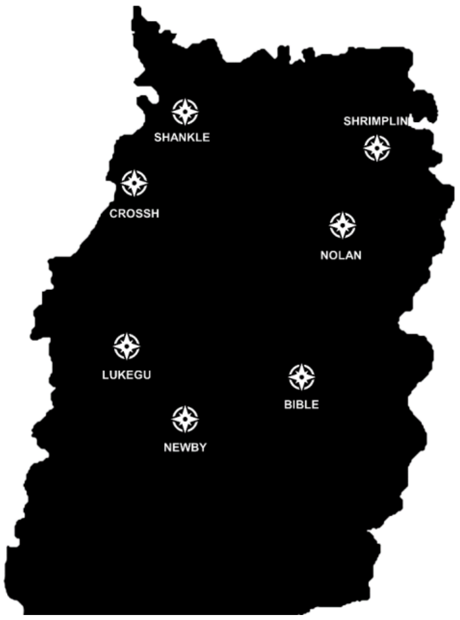

Fig. 1 Position of the studied wells respect to the Hugoton and Panoma field.

Table 1 Facies and adjacent facies

Facies number	Facies	Adjacent Facies
1	Nonmarine sandstone (SS)	2
2	Nonmarine coarse siltstone (CSiS)	1, 3
3	Nonmarine fine siltstone (FSiS)	2
4	Marine siltstone and shale (SiSh)	5
5	Mudstone (MS)	4, 6
6	Wackestone (WS)	5, 7, 8
7	Dolomite (D)	6, 8
8	Packstone-grainstone (PS)	6, 7, 9
9	Phylloid-algal bafflestone (BS)	7, 8

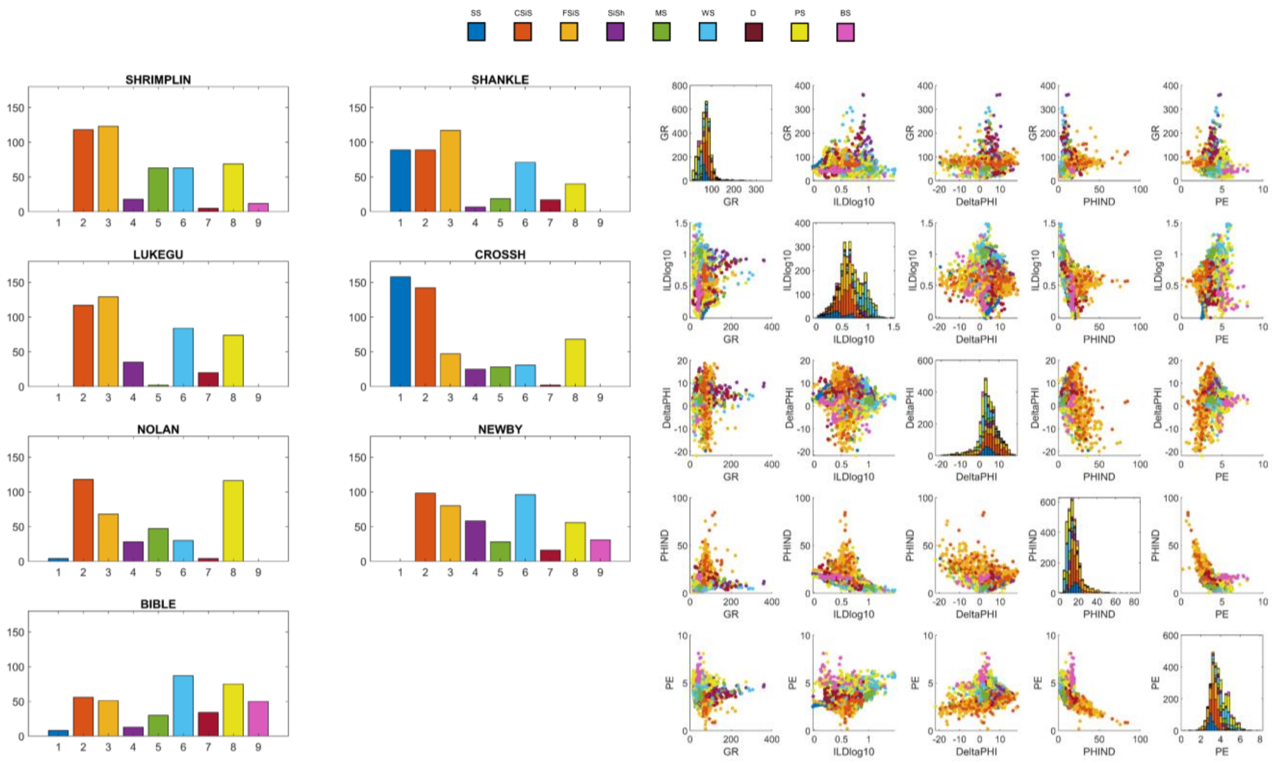

Fig. 2 (a) Facies distribution on the studied wells; (b) Well logs cross-plots.

The relations between the matrices are synthetises in equation 1. For a clearer comprehension of the dataset in Fig. 3 the *Shrimplin* well data are shown. As already mentioned in section 2.1 the dataset used in this work is not equal to the original dataset of Dubois et al. (2007) in fact, one well and two well features are missing. This choice has been taken for two main reasons. Firstly, the missing well was an "artificial well" added by the author and secondly, the two well features are geological features coming from core analysis which cannot be known using only well logging analysis. As a matter of fact, the aim of this project is to simulate a real facies classification evaluation where only the well logging data are available.

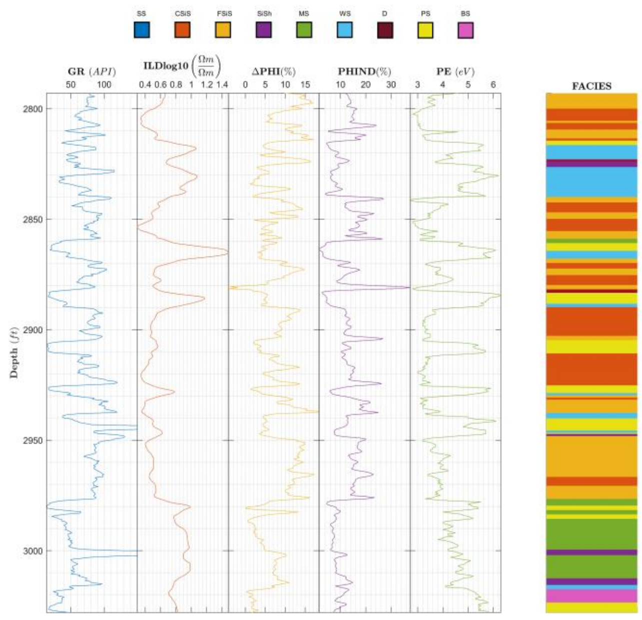

Fig. 3 Dataset of the *Shrimplin* well

$$D^{3164x7} = \left[\begin{array}{cc} \underbrace{\begin{matrix} \text{depth}_1 & GR_1 & \text{ILDlog}\,10_1 & \Delta PHI_1 & PHIND_1 & PE_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \text{depth}_{3164} & GR_{3164} & \text{ILDlog}\,10_{3164} & \Delta PHI_{3164} & PHIND_{3164} & PE_{3164} \end{matrix}}_{F^{3164x6}} & \underbrace{\begin{matrix} FACIES_1 \\ \vdots \\ FACIES_{3164} \end{matrix}}_{C^{3164x1}} \end{array}\right] \qquad (1)$$

2.2 Methodology and evaluation metrics

Creating a machine learning solution for a problem involves first, preparing the data and designing the model then the second part is coding the model to run it on a machine. For this purpose, in this project the MATLAB programming language with its Framework Machine Learning and Deep Learning was used. To define how the model results will be evaluated the F-measure, the Accuracy, and the ROC curve methods will be applied to the test data and the results will be judge based on three metrics, in particular:

- M1 – Absolute
- M2 – Adjacent
- M3 – Pay zone

These three metrics refer to the facies focus on the test data: the pay zone metric M3 will evaluate the performance of the model only on how it classifies the pay zone facies which are facies class 6, 7, 8, 9. The adjacent M2 metric includes all the facies classes and considers an adjacent facies misclassification as a *True Positive* case, which means that the adjacent facies are considered as the same facies. The absolute metric M1 is the standard evaluation which distinguishes between all the facies classes.

2.3 Data pre-processing

The data pre-processing consists of elaborating the initial dataset to ensure it is suitable for the ANN training process. As already mentioned, the *D* matrix contains all the data used to model the ANN which are the well logging data and the associated facies target. Different methods are generally used to refine the raw data included in the *D* matrix aiming to adapt the facies classification problem to the ANN and enhancing the performance.

As shown in Fig. 3 the input data have different ranges of variation thus before being applied in the ANN these data are usually rescaled to the same ranges of variation. This process is named normalization and it is a fundamental part of all the machine learning methods because it changes the data which are the most relevant component of the whole workflow and so it has an important influence on the overall performance (Ioffe and Szegedy, 2015). There are different normalization methods but in this work was applied the most popular one used in the machine learning classification problems which normalize the data in the range -1 to 1. This can be done by applying to each of the six wireline logging data vector the following equation

$$p^n = \frac{2\left(p - p^{\min}\right)}{\left(p^{\max} - p^{\min}\right)} - 1, \qquad (2)$$

where $p^{\min}$ and $p^{\max}$ are the minimum and maximum values of each input components, and p^n is the normalized input (Hagan et al., 2014a).

Another fundamental process of data pre-processing in a machine learning classification problem is coding the components of the class vector *C*. For this work this means assigns at each of the nine different facies included in the dataset a unique code which is used to identify the facies during all the machine learning process. The simpler method is the one scalar targets, which consists of classifying the facies targets with a number from 1 to 9 as was done in Table 1 however, this coding targets method does not give the best results when applied to the network (Hagan et al., 2014a). The method used in this work is the one who tends to enhance the network performance which is coding the targets as a vector which has one column and as many rows as the number of facies; for instance, the nonmarine sandstone (SS) lithology which would be marked as 1 by using the one scalar target method would become [1, 0, 0, 0, 0, 0, 0, 0, 0].

2.4 Architecture

The architecture of the supervised learning ANN depends on the type of problem it is trying to solve which in this case is a facies classification. The multilayer perceptron (MLP) is the most suitable architecture for a classification problem (Hagan et al., 2014a), thus this is the choice for this work. The dimensions of the input and output layers are decided by the number of inputs and targets which are respectively represented by the six wireline logs and the nine facies. The number of neurons for each hidden layer and the number of hidden layers depend on the complexity of the problem and cannot be known before the end of the training process. However, there are no theoretical evidence that justify the use of more than two hidden layer (Masters, 1993) moreover, the training process would become strongly time consuming thus only one and two hidden layer numbers were the possible choices. Therefore, instead of choosing only one architecture several combinations of one and two hidden layers with different number of neurons were explored and trained in what is called a "brute force attack".

Another fundamental part in designing the network architecture consists of deciding how to elaborate the input in each network layer, this implies the choice of the combination of weights and biases and the transfer function. After receiving the input vector p from the neurons of the previous layer, each neuron combines these inputs with its own weight matrix W and the bias b to obtain the net input n as

$$n = W\,p + b. \tag{3}$$

Subsequently, the net input is elaborated by the transfer function to obtain the final output of the neuron. Beside the input layer which does not have a transfer function, the choice of the transfer function for each layer depends on the data pre-processing method used to normalize the input and coding the targets. The inputs were normalized in a range [-1,1] so the transfer function used in the hidden layer is the hyperbolic tangent sigmoid function which shares the same range with the inputs as shown in Fig. 4. Differently, the transfer function used in the output layer is the *softmax*, this function normalizes the net sum of each neuron of the output layer in a range [0,1] simply dividing the net sum of the neuron *i* by the sum of all the 9 net sum of the output layer as

$$a_i^o = \frac{n_i}{\sum_1^9 n_i},\ a_i^o \in [0,1] \tag{4}$$

where a_i^o is the output of the neuron *i* of the output layer, and n_i is the net sum of the neuron *i* of the output layer. After applying the *softmax*, the final output of the network is composed by the nine a_i^o which are scalar values in a range 0 to 1, thus the higher value between them is used as the facies prevision of the network. This is done simply converting the higher a_i^o to 1 and the other eight to zero obtaining a vector like the facies codes presented in Table 1 which is associated with a particular facies target that represents the prevision of the network. In Fig. 5 and Fig. 6 the chosen architecture in case of one or two hidden layers are shown. The output of the network is calculated in cascade starting from the input layer, and for the one hidden layer network this is formulated as

$$a^o = softmax(W^o\, tansig(W^1 p + b^1) + b^o),$$

$$\mathrm{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_6 \end{bmatrix},\ b^1 = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_S \end{bmatrix},\ b^o = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_9 \end{bmatrix},\ a^o = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_9 \end{bmatrix},$$

$$W^1 = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,6} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,6} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,6} \end{bmatrix},$$

$$W^o = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,S} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,S} \\ \vdots & \vdots & & \vdots \\ w_{9,1} & w_{9,2} & \cdots & w_{9,S} \end{bmatrix}, \tag{5}$$

where a^o is the final facies prevision, p is the input vector, the superscript over W and b stands for the type of layer (1 is the hidden layer, *o* is the output layer) *softmax* and *tansig* are the softmax and the hyperbolic tangent sigmoid transfer functions. The dimensions of these entities depend on the six inputs number and the S number of neurons in the hidden layers. The same reasoning is valid for two hidden layers model thus the output of the network is calculated as

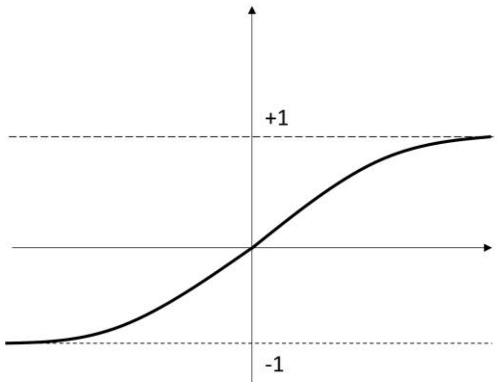

Fig. 4 Hyperbolic tangent sigmoid transfer function

$$a^o = \text{softmax}\,(W^o tansig(W^2 tansig(W^1 p + b^1) + b^2) + b^o), \qquad (6)$$

where in this case are also included the weight and the bias of the second hidden layer marked with the superscripts 2.

2.5 Data division

Before training the supervised learning ANN is necessary to define which part of the dataset will be used to train the model and which one will be used to test it. Usually, in a supervised learning ANN the dataset samples are randomly divided in 75% for training and 25%

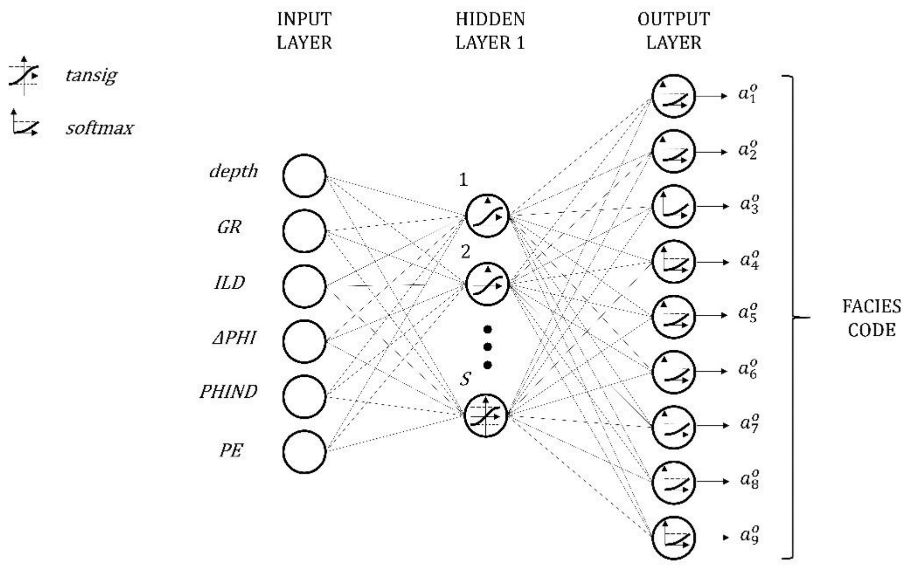

Fig. 5 One hidden layer MLP

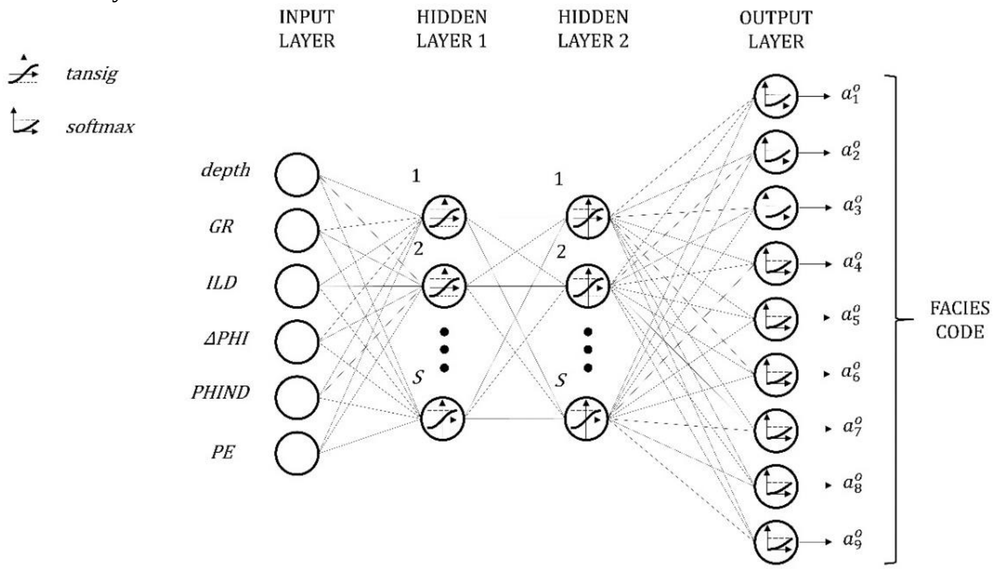

Fig. 6 Two hidden layers MLP

for testing however, this is not the case of this work. As a matter of fact, although the random division of the dataset produces a more performing model (Hagan et al., 2014a) this would be meaningless in a real facies classification problem because the test data samples would be spread in all the wells included in the dataset. This means that the model would make predictions on random parts of the seven wells which imply a scenario where in some points of the well the lithologies are known and in others are unknown. Oppositely, the hypothetical scenario that this work aims to simulate is the case where the ANN model is applied to a well where the lithologies are fully unknown and it is included in a field with other wells with already classified lithologies. To accomplish this scope the test data was defined by all the samples of one of the seven wells, and the data from the others six wells were used to train the model. Instead of choosing between one of the wells to test the model, seven different model were trained therefore in this way was possible to analyse how changing the test well influences the performance of the network.

2.6 Training

The training process is the core of a supervised learning ANN workflow, and it is concern on minimizing the performance function updating the network weights and biases at each training iteration until the training is finished. The final weights and biases constitute the trained network which represents the mathematical model that tries to solve the classification problem. The training process has included the selection of the following: the possible architectures trained, the performance function, the training algorithm, and the techniques to improve the network generalization. As already mentioned in section 2.4, different network architectures were trained using a "brute force attack". A brute force attack is a term used in the information technology to describe the use of all the possible solution to solve a problem for instance, trying all the possibilities to guess a password. In this case study the possibilities are represented by the different MLP network architectures which were trained to obtain a network good in generalization which is the capacity of the model of achieving high performance without overfitting. The architectures trained are the two in Fig. 5 and Fig. 6 which are respectively composed by one and two hidden layers, no more than two hidden layers were used to the reasons explained in section 2.4. The number of neurons for each hidden layer were selected always doubling the number of neurons in a range from 1 to 128, where 128 neurons were selected as the maximum neurons number to keep the training time manageable. The combinations of hidden layers and neurons in each hidden layers are outlined in Table 2. The performance function selected for this training method is the *crossentropy* function which is calculated as

$$F(x) = -\frac{1}{N}\sum_{i=1}^{N} [t_i \ln(a_i^o) + (1 - t_i)\ln(1 - a_i^o)] \quad (7)$$

where t_i is the target of the output neuron *i*, a_i^o is the output of the neuron *i* of the output layer, N is the number of neurons in the output layer multiplied by the training samples. The *crossentropy* performance function was used instead of the minimum square error -*mse*- because initially the weights and biases of the network are randomly initialized before the first training iteration thus this would create a very high error in the output neurons. In this case by using the *mse* the training process would slow down biasing the network performance (Nielsen, 2015).

To train the network the conjugate gradient backpropagation algorithm (CGBP) was used because this was proven very efficient in the classification problems (Hagan et al., 2014b). CGBP algorithm tries to find the minimum of the performance function reducing the size of

Table 2 Trained architectures

Hidden layer number	Neurons in the hidden layer
1	1, 2, 4, 8, 16, 32, 64, 128
2	1, 2, 4, 8, 16, 32, 64, 128

the interval where a minimum point is placed (Hagan et al., 2014c). This algorithm is applied in batch mode which means that at each iteration all the training inputs are used to calculate the error in the performance function to update the new network weights and biases. Additionally, the early stopping technique is applied to the CGBP as a stopping criteria to avoid overfitting. The early stopping technique restrict the network complexity by stopping the training before the minimum is reached trying to obtain the optimum point. To apply this technique first the training data - which include the total dataset beside the test data - were divided in two parts: the effectively training data which were used to apply the CGBP, and a validation dataset which was used as an indicator of the error during the training iterations. The validation data were selected randomly between the total training data (total dataset minus the test data) and counts the 10% of the total training data samples leaving the other 90% to train the network with the CGBP. When the training starts, at each training iteration the performance function (which represents the error) is evaluated for both the effectively training data and the validation data as is shown in the example of Fig. 7. Until iteration *n* the performance function calculated on both training and validation data are decreasing thus the training continue. However, after iteration *n* the performance function on the validation data starts increase and after a specific amount of consequently increasing iteration - also called validation checks - the training stops at point *s*. In this case study the validation checks are set to eight which is a higher value compared with the one suggested by the MATLAB documentation (MathWorks, 2020), this choice is taken because by experimenting the training process with less validation checks the network immediately reached the stop point *s*. For completeness, to set up the CGBP two parameters have been selected: λ and β. These have been set to the default values suggested by the MATLAB documentation because otherwise a specific study about them would have been necessary (MathWorks, 2020).

Finally, another technique to improve the network generalization based on committees of networks was used. As a matter of fact, a single training run could not reach the minimum of the performance function because before the training starts all the network weights and biases are randomly initialized to be ready for the first training iteration. This initialization represents the starting point of the training process, and it strongly influences the training journey to the minimum point and thus the training results (Hagan et al., 2014a). To avoid this effect committees of networks were trained with different starting point, validation data, and effectively training data. Subsequently, before coding the outputs the average of the raw network response was calculated and then the effectively outputs were coded. The committees of networks are made by 10 networks to maintain a manageable duration of the training processes. To be clear, this process is repeated for each architecture described in Table 2 therefore the total training processes performed are

$$\begin{aligned} Training\ Combinations &= h\ l\ c\ w \\ &= 8 * 2 * 10 * 7 \\ &= 1120, \end{aligned} \tag{8}$$

where h is the total possible combination of neurons in the hidden layer (1,2,4,8,16,32,64,128), l are the two possible number of hidden layers tested (1,2), c are the number of networks in the committees, w is the number of well used to test the model which include all the wells in the dataset because each

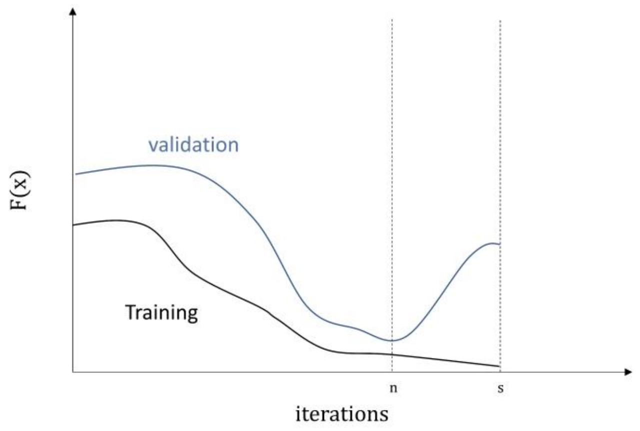

Fig. 7 Example of Early Stopping technique

of the 7 wells were used to test the model one by one, when the other 6 were used for training. In sum the training process has included multiple choices and techniques used to try to improve the performance of the classifier and creating a model good in generalization. Precisely, the "brute force attack", the early stopping, and the committee of networks were used together in the same training process.

2.7 Balancing the dataset

A fundamental aspect in the machine learning classification problems is the proportion between the observed classes included in the dataset. As a matter of fact, if the dataset samples are imbalanced between all the classes the learning algorithm will be biased to the class with the greater number of samples ignoring the classes with samples deficit (Krawczyk, 2016). This is a direct consequence of the training process in fact, the network weights and biases will be updated by the training algorithm trying to minimize the performance function towards the class with the greater number of samples. For this reason, the resulting model of an imbalanced dataset will not generalize good the test data. Indeed, the dataset presented in section 2.1 used in this work is - as many real-life situation - an imbalanced dataset, this is noticeable in Table 3 where the facies class count for each of the seven wells are reported. In particular:

- All the facies classes beside class 1 and class 9 are included in all the seven wells.
- The Phylloid-algal bafflestone (BS) which is marked as class 9 is the facies with the least number of samples with a count of 93 in all the dataset. Moreover, this facies class is present in only three wells.
- The Nonmarine sandstone (SS) which is facies class 1 is present in only 4 wells.
- Only one well (Bible) includes all the facies classes.

Table 3 Original facies count per class in each well

Well	Facies class									Total per well
	1	2	3	4	5	6	7	8	9	
SHRIMPLIN	0	118	123	18	63	63	5	69	12	471
SHANKLE	89	89	117	7	19	71	17	40	0	449
LUKEGU	0	117	129	35	2	84	20	74	0	461
CROSSH	158	142	47	25	28	31	2	68	0	501
NOLAN	4	118	68	28	47	30	4	116	0	415
NEWBY	0	98	80	58	28	96	16	56	31	463
BIBLE	8	56	51	13	30	87	34	75	50	404
All Dataset	259	56	51	13	30	87	34	75	50	3164

Table 4 Facies count per class in each well after oversampling

Well	Facies class									Total per well
	1	2	3	4	5	6	7	8	9	
SHRIMPLIN	0	158	158	158	158	158	158	158	369	1475
SHANKLE	277	158	158	158	158	158	158	158	0	1383
LUKEGU	0	158	158	158	158	158	158	158	0	1106
CROSSH	277	158	158	158	158	158	158	158	0	1383
NOLAN	277	158	158	158	158	158	158	158	0	1383
NEWBY	0	158	158	158	158	158	158	158	369	1475
BIBLE	277	158	158	158	158	158	158	158	369	1752
All Dataset	1108	1106	1106	1106	1106	1106	1106	1106	1107	9957

For the mentioned reasons the dataset can be classified as high imbalanced thus the resulting model performance will be severely influenced by this effect. Nevertheless, to avoid the imbalanced classes phenomenon different techniques has been developed. The approach used in this work is named random oversampling and consists of balancing the dataset adding randomly repeated data samples to the facies class in the minority groups until all the classes count the same amount of data samples (Krawczyk, 2016). To achieve this, it was decided that the total number of samples of each facies class must be equal to

$$facies\ per\ class = 158 * 7 = 1106, \qquad (9)$$

where 158 is the higher count of facies in all the dataset (represented in facies class 1 of the CrossH well) and 7 are the wells number. The results of applying this oversampling technique to our dataset is shown in Table 4 where the sum of the facies count for each class is equal to 1106 for all the classes beside class 1 and 9 due to rounding. It is important to notice that the total facies samples (9957) after the random oversampling are three time the original dataset samples (3164), so this will bias on longer training processes. Balancing the data has been demonstrated to improve the performance of the classifier (He and Garcia, 2009), and moreover, studies have shown that the random oversampling applied in this work is a very reliable method to avoid imbalance data (Ling and Li, 1998).

2.8 Sample complexity study

One crucial problem in machine learning is to determine how many data are needed to generate a successful model, this problem is known as sample complexity (Bartlett, 1998). As a matter of fact, the performance of a machine learning classifier increases with the training data size according to a power law (Mitsa, 2019) thus understanding how the network performs as a function of the dataset size is fundamental. To study this effect in our facies classification problem, different models were build reducing the number of wells used in the training process starting from the model with the complete original dataset until the one which includes only the data of two of the total seven wells. All the combinations of the wells used to develop the models are reported in Table 5 where the combinations number is calculated as

$$\frac{7!}{k!\,(7-k)!}, \qquad (10)$$

Where 7 are the wells included in the dataset, and k is the number of wells used to develop the model. For each combination k models were created to evaluate all the possible test wells for instance, by using all the seven wells (first line of the Table 5) the same amount of models were created to use all the wells as a test well one by one. Moreover, each model was processed by applying the training process of section 2.6 with the two possible hidden layers architectures and all the possible neurons number. Therefore, the total number of models trained are

$$\sum_{k=2}^{7} \frac{7!}{k!\,(7-k)!} k\ h\ l\ c = 70560, \qquad (11)$$

where h is the eight possible hidden neurons number for each hidden layer (1, 2, 4, 8, 16, 32, 64, 128), l are the two possible hidden layer architectures (1,2), c are the ten networks used to create the committee of networks, k are the combinations of test wells already explained. Finally, the average of the accuracy of the seven possible number of wells used (the seven

Table 5 Combinations of training models

Hidden layer number	Combinations
7	$\frac{7!}{7!\,(7-7)!} = 1$
6	$\frac{7!}{6!\,(7-6)!} = 6$
5	$\frac{7!}{5!\,(7-5)!} = 21$
4	$\frac{7!}{4!\,(7-4)!} = 35$
3	$\frac{7!}{3!\,(7-3)!} = 35$
2	$\frac{7!}{2!\,(7-2)!} = 21$

lines in table) were calculated for all the possible architectures on the three metrics M1, M2, M3. The process described above aims to obtain a complete view of how the accuracy changes as a function of the dataset size (number of wells used), and to understand which architecture best fit the various model trained with different dataset size.

3 Results and discussion

3.1 Original dataset results

In Fig. 8 and Fig. 9 the F-measure of the resulting models trained using the original dataset and the architectures reported in Table 2 are shown. Fig. 8 shows the F-measure calculated with one hidden layer network architecture for all the seven wells used one by one for testing. The F-measure is function of the number of neurons in the hidden layer and is calculated for all the three metrics M1, M2, M3. Equally, Fig. 9 shows the same results calculated with a two hidden layer network architecture. For clearness, in both the mentioned figures the test well is reported as a number from 1 to 7, these number are linked with the wells name as described in Table 6. As expected, the trained network has higher performance on metrics M2 and M3, and this is true whatever test well is used. This effect is easy to understand because metrics M2 and M3 do not classify the samples in all the nine facies classes as metric M1 does. As a matter of fact - as already said in section 2. - metric M2 do not consider an adjacent facies classification as an error but as a *True Positive* case thus the classification classes are reduced. Similarly, metric M3 only measures whether a data sample is a pay zone facies which is effectively a binary classification, so this is also the reason why the metric M3 reaches the higher F-measure value in all the architectures and test wells. Another important aspect is the shape of the F-measure curves. Indeed, the M2 and M3 metrics have a flatter trend respect to M1 which means that they are not highly influenced by the changing in the number of neurons in the hidden layers, and this is generally valid for both the one and two hidden layer network architecture and all the test wells. A general trend can be also seen for the metric M1 which follows an increasing path until around 8 hidden neurons number to almost stabilize for highly neurons number. The described patterns are consequence of the complexity of the

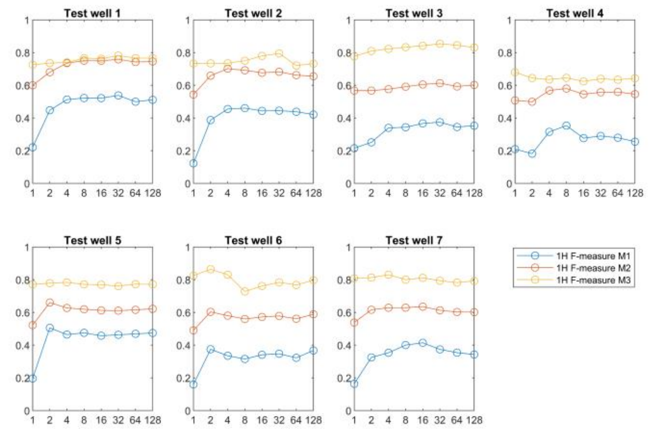

Fig. 8 F-measure for 1 hidden layer architecture in the original dataset

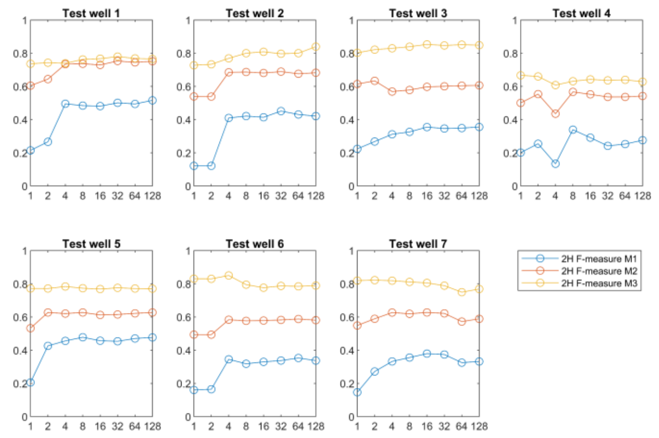

Fig. 9 F-measure for 2 hidden layer architecture in the original dataset

Table 6 Wells codes

Code	Well name
1	SHRIMPLIN
2	SHANKLE
3	LUKEGU
4	CROSSH
5	NOLAN
6	NEWBY
7	BIBLE

problem which the model is trying to solve in fact, for what already said the metrics M2 and M3 do not perform a classification of the data samples on all the nine facies thus the classification problem for both these metrics is simpler than the one for metric M1. The higher complexity in the facies classification for metric M1 requires a more complex network architecture - which in this case is represented by 8 hidden neurons - to reach good performance instead, metrics M2 and M3 are already satisfied with one or two neurons. Furthermore, beside the similar patterns almost all the test wells in the mentioned figures also show very similar F-measure performance in fact, the M1 metric curves reaches a maximum of around 0.4, 0.6 for M2 and 0.8 for M3. These highest values are achieved with 8 hidden neurons number for all the test wells and hidden layers number in fact, after 8 hidden neurons the network start overfitting thus it is not capable to increase its performance anymore. The proof of the overfitting trend is given examining Fig. 10 and Fig. 11 where the F-measure of the test set reported in Fig. 8 and Fig. 9 are plotted together with the F-measure of the training data ("train" in figures). As a matter of fact, while in both the figures until 8 hidden neurons the train curves follow the same trend of the test curves, after 8 hidden neurons the F-measure of the test curve start decreasing in all the three metrics while the training curves are still increasing. This is the evidence of an overfitting model because while the network can classify the data used to train the model, it cannot do the same for the test data. One solution to this problem could be taking the training and the test samples spread randomly in the dataset reducing the risk of finding test samples which are not included in the training data however, as already said in section 2.1 this is not the purpose of this work that aims to simulate a real facies classification case study where the test wells represent the unknown facies that the ANN model tries to classify based only on the well logging data. Another solution could be adding a new artificial well made with a mixture of the data in each of the seven wells, this would help to keep the test data inside the training region but even this solution is not acceptable for the same reason.

To have a cleaner point of view, in Fig. 12 the average F-measure of all the seven test wells divided by the three metrics M1, M2, M3 are shown. As already said, the number of neurons in the hidden layers hardly influences the network performance in metric M1 while this effect is softer for metric M2 and completely absent on metric M3 which shows a flat pattern. The hidden layers number seems to have a lower weight on the network performance while the one hidden layer (1H in figure) achieves slightly higher results than two hidden layers (2H in figure) in all the metrics. In sum, the most performing architecture in all the three metrics is the one hidden layer and 8 hidden neurons per layer which achieves 0.4099 in M1, 0.6274 in M2, and 0.7731 in M3. The results on the pay zone metric M3 are the

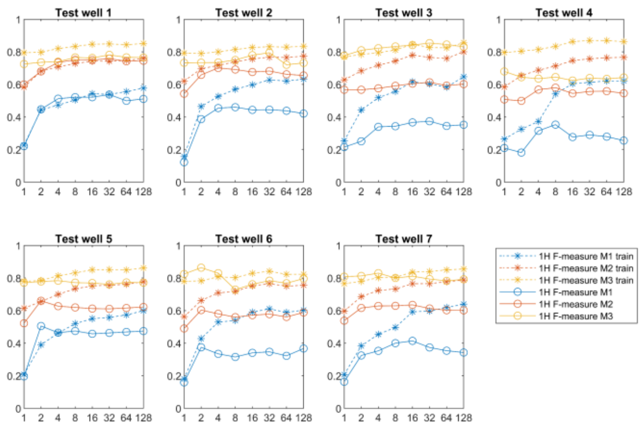

Fig. 10 Testing oversampling in 1 hidden layer architecture in the original dataset

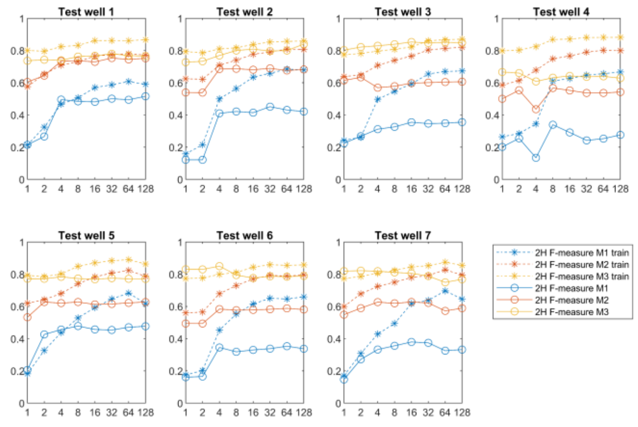

Fig. 11 Testing oversampling in 2 hidden layers architecture in the original dataset

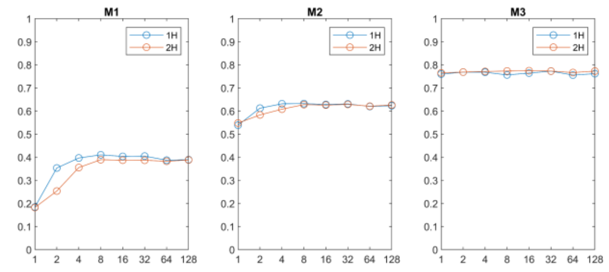

Fig. 12 Average F-measure of the seven test wells on the three metrics in the original dataset

most interesting because effectively the model shown high capability in recognizing whether a facies is included in the pay zone or not and this is in the main interest in a formation evaluation problem.

These results should not be compared with the results of the SEG contest (Hall and Hall, 2017) because although the database used in this work comes from the same source, it differs on the number of samples and input data used. In fact, as already mentioned this database do not include any artificial well and geological features which can be both classified as high reliable data. However, the methodology used in this work can be considered analogous to the one proposed in the mentioned contest where the first ranked team achieved an accuracy of 0.6388 in absolute which coincide with the absolute metric M1 (Hall and Hall, 2017). As shown in Fig. 13 the higher average accuracy of the seven test wells on the metric M1 performed in this case study is 0.4312, achieved with one hidden layer (1H) and 8 neurons architecture. This result is an optimum score due to the consideration made for the differences in the dataset which are related with the aim of the project. Another comparative parameter to evaluate the results achieved is represented by the accuracy scored by a random classifier which is equal to 0.16

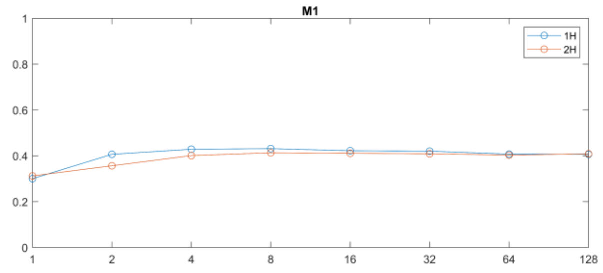

Fig. 13 Average accuracy of the seven test wells on the absolute metric M1 in the original dataset

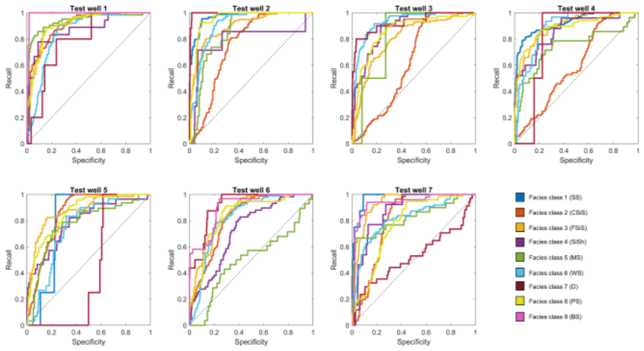

Fig. 14 ROC curve for one hidden layer and 8 neurons architecture in the original dataset

thus, this supports the validity of our results (Hall and Hall, 2017). To better understand how the model performs on each facies class the ROC curves of the architecture who achieved the highest F-measure (one hidden layer and 8 neurons) is displayed in Fig. 14 for all the seven test wells. Firstly, the facies show good results in terms of classification, in fact the curves are far beyond the dashed curve - which represents a random classifier performance - in almost all the wells, and this is shown also in Table 7 where the area under the curve (AUC) is higher than 0.8 for almost all the facies and test wells. However, there are some exceptions such as

- Facies class 2 scored an AUC of 0.7522 in test well 2, 0.6219 in test well 3 and 0.5976 in test well 4.
- Facies class 5 scored an AUC of 0.7375 in test well 4 and 0.5021 in test well 6.
- Facies class 7 scored an AUC of 0.4222 in test well 5 and 0.5289 in test well 7.

Additionally, the misclassification of these facies outlined with the ROC curves and AUC values could not be included in the statistical error of the classifier because by considering Table 8 which shows the average AUC of each facies class in the all the wells weighted on the facies count of Table 3, effectively facies class 2 (nonmarine coarse Siltstone), class 5 (Mudstone) and class 7 (Dolomite) score an AUC of respectively 0.7557, 0.7941 and 0.7689 which are the lowest results in all the nine facies. Therefore, these results show that the model biases on misclassifying facies class 2, 5, and 7 meaning that its associated well logs

Table 7 AUC of the nine facies classes in all the test wells in the original dataset

Well code	AUC								
	1	2	3	4	5	6	7	8	9
1	NA	0.9086	0.9104	0.8696	0.9366	0.8508	0.7876	0.9093	0.9909
2	0.9690	0.7523	0.8952	0.7886	0.8508	0.9057	0.9956	0.9518	NA
3	NA	0.6219	0.8033	0.8865	0.8235	0.9323	0.9256	0.8596	NA
4	0.9227	0.5976	0.8988	0.8599	0.7375	0.8877	0.8086	0.8860	NA
5	0.8029	0.8520	0.8754	0.7888	0.7697	0.7531	0.4221	0.8606	NA
6	NA	0.8199	0.8499	0.7269	0.5021	0.8188	0.9181	0.8402	0.9034
7	0.9770	0.8037	0.9349	0.8739	0.8203	0.8166	0.5289	0.7713	0.9381

Table 8 Average AUC in all the test wells weighted on the facies count in the original dataset

Weighted AUC								
1	2	3	4	5	6	7	8	9
0.9384	0.7557	0.8744	0.8114	0.7941	0.8571	0.7689	0.8623	0.9333

are not easily distinguished by the model. Probably the cause of this effect is that the wells where facies class 2, 5, and 7 score a low AUC include unique data samples of those facies classes which cannot be found in other wells thus when these wells are used for testing instead of training the network cannot performs a good classification on the mentioned facies because it did not learn those information in the training process. As a matter of fact, the ANN is a good tool in data interpolation while it is not in data extrapolation therefore, when test data are outside the training data region the network overfits decreasing its performance (Hagan et al., 2014d).

3.2 Balanced dataset results

As it was for the results of the original dataset, in Fig. 15 and Fig. 16 the F-measure of the models trained with the balanced dataset as a function of the tested architectures and the three metrics are shown. The first noticeable thing is that these results are not higher than those from the original dataset (Fig. 8, Fig. 9), meaning that balancing the facies have not improved the performance of the classifier. Oppositely, although the maximum F-measure values reached do not severely differ, the general trend is that these results are lower than those achieved with the original dataset and this is valid mainly for the metric M2 and M3 which are substantially lower in test well 4 and 5. Actually, the random oversampling technique used to balance the facies introduces further overfitting in the models as is shown in Fig. 17 and Fig. 18 where the training set results were plotted together with the test data results. The overfitting pattern with the balanced dataset is even more visible than the original dataset because the F-measure training results ("train" in figures) are decisively higher than those in the original dataset. This is evident by looking at the trends of the training and test data which begins to deviate already at 2 hidden neurons, and this is valid for all the wells and the hidden layers number. In effect, the random oversampling has increased the network performance in the training data, meaning that the model good classifies the data samples used to train it however, the results on the test data did not increase as well. The increasing of overfitting in the network performance is a well know drawback of the oversampling techniques (Hi and Garcia, 2009) but usually it also brings an increase of the test data performance. Nevertheless, in this case it did not happen probably due to the data division applied in fact, even duplicating the data samples the test well samples still cannot be included in the training data thus the classifier did not learn any information from those data. Certainly, spreading the test data along all the dataset would increase the efficiency of the oversampling technique but as already repeated different times this do not match the aim of this work. In Fig. 19 the average results of the F-measure of the three metrics for the balanced dataset are shown as a function of the hidden neurons and hidden layers number. Differently, from the original dataset, the higher results - compared with the original dataset results in Table 9 - are achieved with 4 hidden neurons number for all the three metrics while the hidden layers number do not affect the results because the two curves (1H, 2H) are practically overlapping.

3.3 Sample complexity study

In Fig. 20 and Fig. 21 the accuracy results of the sample complexity study for the three metrics M1, M2, M3 on both one hidden and two

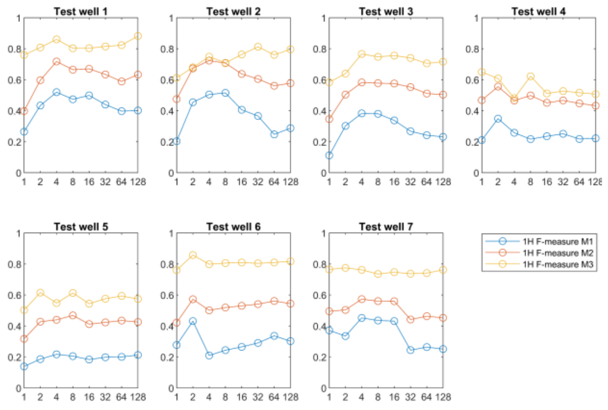

Fig. 15 F-measure for 1 hidden layer architecture in the balanced dataset

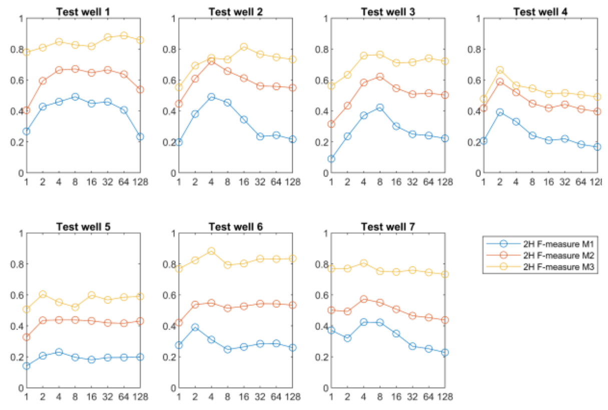

Fig. 16 F-measure for 2 hidden layers architecture in the balanced dataset

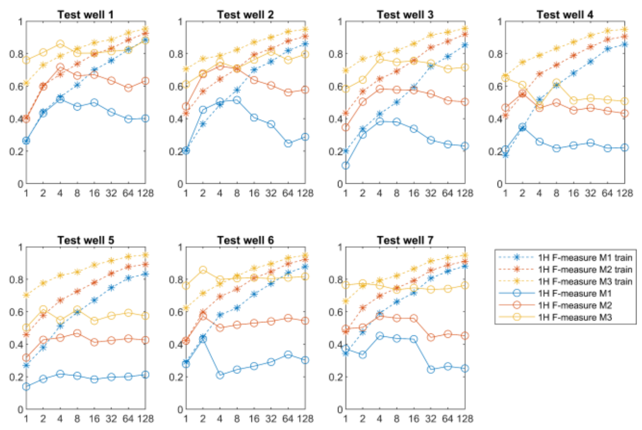

Fig. 17 Testing oversampling in 1 hidden layer architecture in the balanced dataset

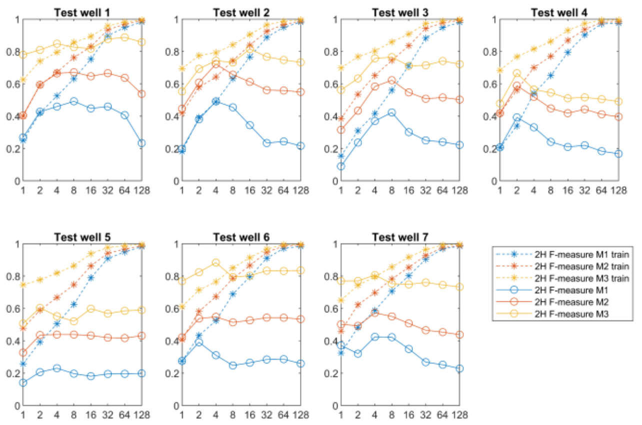

Fig. 18 Testing oversampling in 2 hidden layers architecture in the balanced dataset

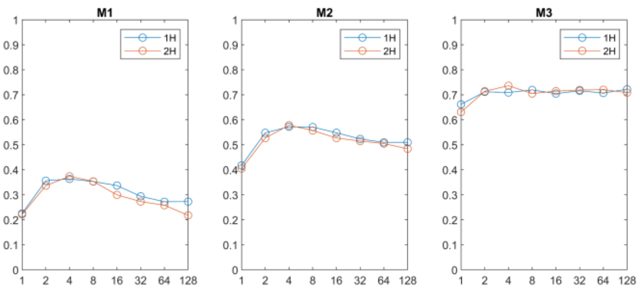

Fig. 19 Average F-measure of the seven test wells on the three metrics for the balanced dataset

Table 9 Summary of the F-measure results averaged in all the wells obtained with the two different dataset for the tree metrics

Dataset	M1	M2	M3
Original	0.4099	0.6274	0.7731
Balanced	0.3739	0.5783	0.7363

hidden layers architectures are shown. In the figures the accuracy is reported as a function of both the number of wells used (horizontal axis) and the hidden neurons number (vertical axis) in a colour map. Firstly, the two figures show almost identical results in all the three metrics thus this supports what already outlined about the independence of the results against the hidden layers number. The most evident result is the decrease of the accuracy with the reducing of the number of wells used, and this is valid for both the architectures and all the three metrics. Obviously, the metric M2 and M3 less suffer the decreasing of the wells number than the metric M1 because as described in section 3.1 these metrics effectively sorts the data samples in less than nine facies. This is an expected results because as already said in section 2.8 in general the accuracy of a classifier increases with the dataset size with a power law (Mitsa, 2019). This is confirmed also in this case study in particular, in Fig 22 the accuracy on the metric M1 of one hidden layer with 8 neurons architecture- which is the architecture who achieved the higher score in the M1 - is shown. The curve in black is a fitting (R-square = 0.9954) of the data point (blue crosses in figures), and its power law equation is expressed as

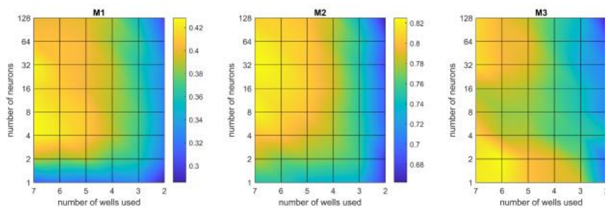

Fig. 20 Accuracy for 1 hidden layer architecture

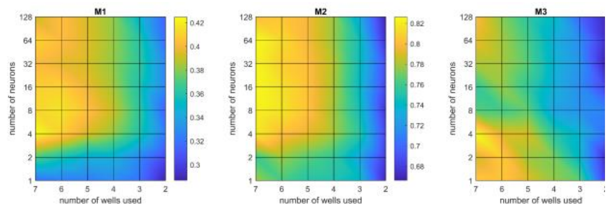

Fig. 21 Accuracy for 2 hidden layers architecture

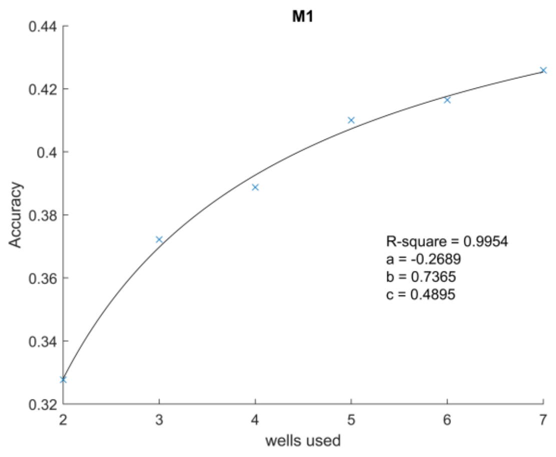

Fig. 22 Accuracy in the metric M1 of the one hidden layer and 8 hidden neurons architecture as a function of the wells used

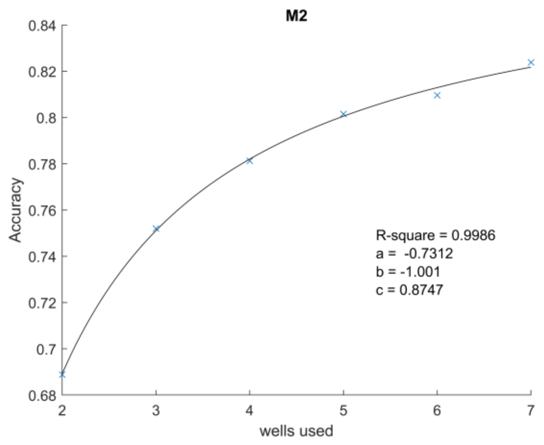

Fig. 23 Accuracy in the metric M2 of the one hidden layer and 8 hidden neurons architecture as a function of the wells used

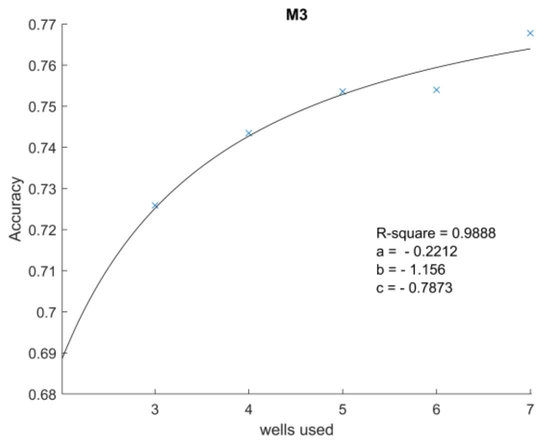

Fig. 24 Accuracy in the metric M3 of the one hidden layer and 8 hidden neurons architecture as a function of the wells used

$$f(x) = ax^b + c,\quad a = -0.2689,\quad b = 0.7365,\quad c = 0.4895, \tag{12}$$

where x represents the number of well used and a, b, c are the power law coefficients. The M2 and M3 metrics also follow a power law trend as shown in Fig. 23 and Fig. 24 thus this can be considered valid in general.

4 Conclusion

The aim of this work is to study the feasibility of the Artificial Neural Network (ANN) technique on a facies classification problem. Different ANN models were developed and trained using supervised learning techniques based on a dataset which includes a series of well logging data and associated recognized facies taken from the Hugoton and Panoma fields in Kansas US. The performance of the ANN models build was evaluated based on three metrics M1, M2, M3 which judge the classifier in terms of how it classifies the nine facies classes, the adjacent facies and the facies constituents the pay zone. The accuracy, the F-measure, and the ROC curve as well as the AUC values were used to quantify the model's performance on each metric. Subsequently, the dataset was changed to balance the facies classes oversampling the data trying to enhance the model performance. Finally, the influence of the dataset size on the model performance was analysed by applying a sample complexity study.

The higher F-measure achieved by the model trained with the original dataset on the three metrics M1, M2, M3 are respectively 0.4099, 0.6274, 0.7731. These results were achieved with one hidden layer and eight hidden neurons architecture that has been proven to be the optimum network layout. In fact, by using less than eight hidden neurons the performance on the three metrics is lower and by using more than eight hidden neurons the network shows an overfitting trend. Actually, the results show that the number of hidden layers do not influence the network performance. The 0.7731 scored on the F-measure for the metric M3 shows that the model is a good tool to distinguish the pay zone facies, this is an important result considering the data division and that only the well logging data were used as inputs.

The ROC and AUC values have shown that the model has similar performance in almost all the test wells and facies classes however, facies class 2 (nonmarine coarse Siltstone), class 5 (Mudstone) and class 7 (Dolomite) scored lower values in some test wells. As a matter of fact, the model biases on misclassify those facies probably to the uniqueness of well logging data included in the wells used as test well.

Balancing the facies samples turned out to decrease the network performance instead of increasing them. In fact, the results show that although the F-measure on the training set were enhanced, the same did not happen for the test set consequently the model severely overfits the results prematurely. The cause of this effect could be the data division applied to the model which selected a whole well as a test well instead of spreading the test samples along the dataset.

Finally, the sample complexity study has shown that the accuracy of the model effectively decreases as a power law by reducing the wells in the dataset. This result is valid for all the three metrics, both the hidden layers number and hidden neurons number.

5 References

Bartlett, P., (1998). The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. IEEETrans. Inf. Theory 44.

Bestagini, P., Lipari, V., Tubaro, S., et al., (2017). A machine learning approach to facies classification using well logs, in: 2017 SEG International

Exposition and Annual Meeting, Society of Exploration Geophysicists.

Dubois, M.K., Bohling, G.C., Chakrabarti, S., (2007). Comparison of four approaches to a rock facies classification problem. Computers & Geosciences 33, 599–617.

Fanchi, J.R., 2002. Shared Earth Modeling: Methodologies for Integrated Reservoir Simulations. Gulf Professional Publishing. chapter 4. pp. 52–68.

Hagan, M., Demuth, H., Beale, M., De Jesús, O., (2014a). Neural Network Design. 2nd Edition. Martin Hagan. chapter 22. pp. 1–31.

Hagan, M., Demuth, H., Beale, M., De Jesús, O., (2014b). Neural Network Design. 2nd Edition. Martin Hagan. chapter 25. pp. 1–12.

Hagan, M., Demuth, H., Beale, M., De Jesús, O., (2014c). Neural Network Design. 2nd Edition. Martin Hagan. chapter 12. pp. 1–55.

Hagan, M., Demuth, H., Beale, M., De Jesús, O., (2014d). Neural Network Design. 2nd Edition. Martin Hagan. chapter 13. pp. 1–52.

Hall, B., (2016). Facies classification using machine learning. The Leading Edge 35, 906–909.

Hall, M., Hall, B., (2017). Distributed collaborative prediction: Results of the machine learning contest. The Leading Edge 36, 267–269.

Halotel, J., Demyanov, V., Gardiner, A., (2020). Value of geologically derived features in machine learning facies classification. Mathematical Geosciences 52, 5–29.

He, H., Garcia, E.A., (2009). Learning from imbalanced data. IEEE Transactions on knowledge and data engineering 21, 1263–1284.

Hill, D.G., (2017). Formation evaluation, in: Springer Handbook of Petroleum Technology. Springer, pp. 433–500.

Hyne, N., (1991). Dictionary of Petroleum Exploration, Drilling and Production: Tulsa. volume 1.

Ioffe, S., Szegedy, C., (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167.

Krawczyk, B., (2016). Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence 5, 221–232.

Ling, C.X., Li, C., (1998). Data mining for direct marketing: Problems and solutions., in: Kdd, pp. 73–79.

Ma, Y., (2019). Facies and lithofacies classifications from well logs, in: Quantitative Geosciences: Data Analytics, Geostatistics, Reservoir Characterization and Modeling. Springer. chapter 10, pp. 231–254.

Masters, T., (1993). Practical neural network recipes in C++. Morgan Kaufmann. chapter 10. pp. 174–187.

MathWorks, (2020). Deep Learning Toolbox: User's Guide (R2020a).

McPhee, C., Reed, J., Zubizarreta, I., (2015). Best practice in coring and core analysis, in: Developments in Petroleum Science. Elsevier. volume 64, pp. 1–15.

Mitsa, T., (2019). How do you know you have enough training data?, URL: https://towardsdatascience.com/how-do-you-know-you-have-enough-training-data-ad9b1fd679ee, accessed 07/07/2020.

Nielsen, M., (2015). Neural Networks and Deep Learning. Determination Press. chapter 3.

Rao, V., Knight, R., (2017). Sustainable Shale Oil and Gas: Analytical Chemistry, Geochemistry, and Biochemistry Methods. Elsevier. chapter 7. pp. 95–114.