Acta Tecnología - International Scientific Journal about Technologies
Volume: 7 2021 Issue: 3 Pages: 71-77 ISSN 2453-675X

doi:10.22306/atec.v7i3.111 Received: 30 July 2021; Revised: 17 Sep. 2021; Accepted: 28 Sep. 2021

PROBLEM PREDICTION DURING TRIP IN AND TRIP OUT PROCEDURES WITH ARTIFICIAL NEURAL NETWORKS

Ádám Viktor Pásztor
EPAM Systems, Nagy Imre str. 7 III/16 Szolnok Hungary, EU,
padamv91@gmail.com (corresponding author)
Richárd Ürmös
MOL Plc. Horváth Zoltán str. 17 I/7 Kiskunfélegyháza, Hungary, EU,
rurmos@mol.hu

Keywords: artificial neural networks, deep learning, drilling operation, trip in, trip out
Abstract: In recent times, the adaptation of artificial intelligence (AI) technologies has been spread in the petroleum industry. Such methods as Artificial Neural Networks (ANN), Fuzzy Logic, or Evolutionary Computing have the potential to improve the currently applied methods in every sector of the industry. They provide an advanced encroachment of the complex physics of downhole parameters, which directly add to their modeling ability compared to the traditional empirical and analytical methods. In this study, the development of a feed-forward neural network is presented. The purpose of the development is to predict the possible problems in case of a drilling operation, during running in and pulling out of the hole (RIH & POOH), based on the data acquired during the drilling of the hole.

1 Introduction

1.1 Artificial Neural Networks

Neural networks are a set of algorithms modelled loosely after the human brain that is designed for pattern recognition. They interpret sensory data through a kind of machine perception, labelling, or raw clustering input. The pattern recognition is done on numerical data, which is contained in vectors. To make this possible, real-world data must be translated, be it images, sound, text, or time series.

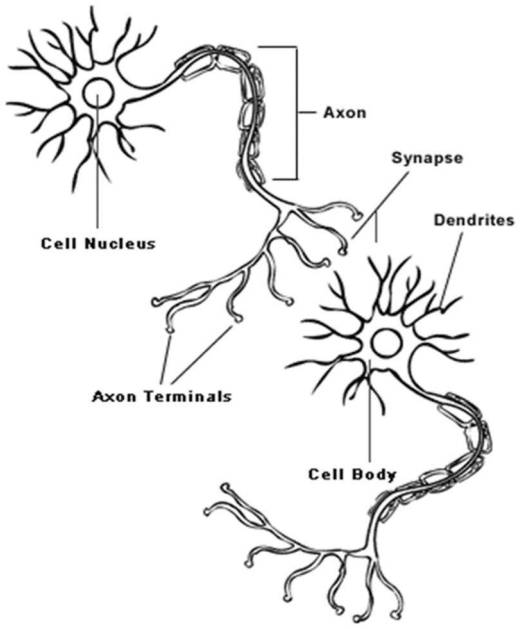

Figure 1 Structure of biological neurons [1]

Each neuron (Figure 1) is a processing tool of our brain. Each neuron will try to stimulate other neurons via its axon terminals and tells whether a terminal should be active or remain inactive. By doing that repeatedly across multiple neurons (the human brain has around 100 billion neurons), our brain can process complex things and solving problems.

Deep learning is the name that is used for networks that are composed of several layers ("stacked neural networks") [2] (Figure 2). The layers are made of nodes, where computation happens. The nodes are loosely patterned on the human brain's neurons, which fire when sufficient stimuli have encountered A node, combines input from the data with a set of coefficients, or weights, that either amplify or dampen that input. With this, significance can be assigned to the input concerning the task the algorithm is trying to learn, e.g., which input is the most helpful in classifying data without error. The sum of the input-weight products is passed through the so-called activation function of a node. The activation function modifies the signal to determine whether and to what extent it should progress further through the network to affect the outcome, e.g., an act of classification. If the signals pass through, the neuron has been "activated" (Figure 3).

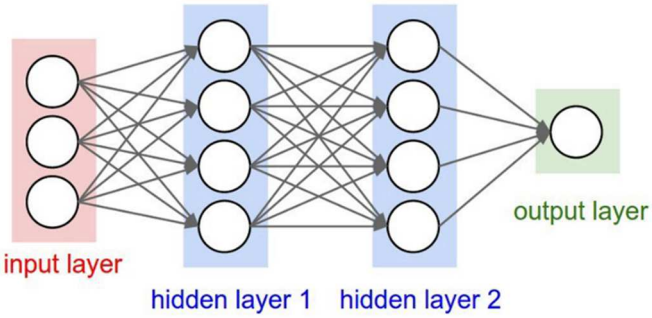

Figure 2 Model of a neural network [3]

PROBLEM PREDICTION DURING TRIP IN AND TRIP OUT PROCEDURES WITH ARTIFICIAL NEURAL NETWORKS

Ádám Viktor Pásztor; Richárd Ürmös

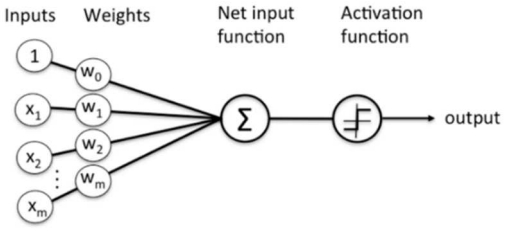

Figure 3 Working model of a node [4]

A node layer is a row of those neuron-like switches that turn on or off as the input is fed through the net. Each output of a layer is simultaneously the input of the subsequent layer, starting from an initial input layer receiving the data.

1.1.1 Working mechanics of neural networks

At the highest and simplest representation, a supervised neural network can be presented as a black box with two methods, learn and predict, as follows (Figure 4).

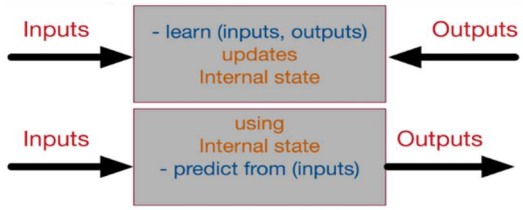

Figure 4 Neural network as a black box

The "learning" process takes the inputs and the desired outputs and updates its internal state accordingly, so the calculated output gets as close as possible to the desired output. The "predict" process takes an input and generates, using the internal state, the most likely output according to its past "training experience". That is the reason why machine learning is sometimes called model fitting. The training procedure of a feed-forward neural network is presented in detail in the following flowchart (Figure 5):

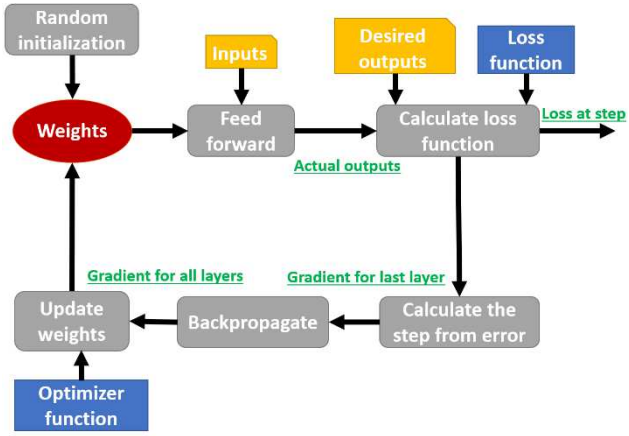

Figure 5 Training procedure of feedforward neural networks

The input parameters which are given to the neural network to process are called features. The nodes have a weight for each feature and a bias term as well. The weights and the bias term together make up the regression parameters. These regression parameters are then passed to an activation function, which decides whether the result is significant enough to "fire" the node, producing the output. The first step of training is to give an initial value for each regression parameter. It is very likely to perform poorly with these random initial values, but the training will essentially punish the network for poor performance.
After the first guess of the network (output calculated with the initial weights), the decision to make is how to modify the weights to reach a better result. In order to do this, first, the level of error needs to be measured. This is done by the application of a so-called loss function, which indicates the severity of error for the current parameters. As a result, the actual goal is to find the minimum of the loss function (Figure 6). The way how this minimum is reached is a question of the optimization method, but to apply any of them, the gradient of the error is needed at the given point corresponding to actual regression parameters. Based on the chosen method, a step is calculated (1), which will improve the result.

$$w_{new} = w_{old} + step \quad (1)$$

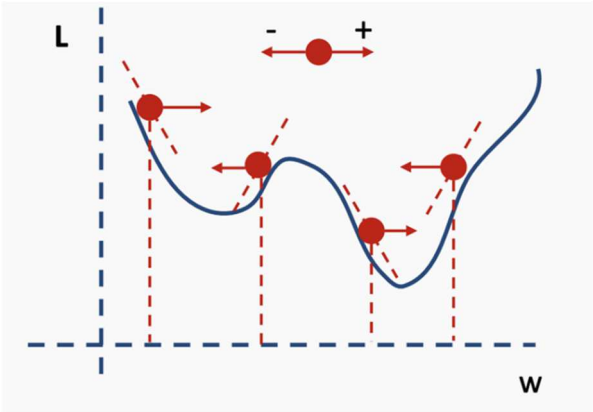

Figure 6: Finding minimum of loss function L(w)

Backpropagation is the learning mechanism of a neural network. It can be considered as the messenger, which tells the network whether a mistake was made or not during the prediction. The development of backpropagation was one of the most important milestones in the field of artificial neural networks.

During prediction, a signal is propagated through the nodes of the artificial neural network to the output layer, where the "decision is made". After the generation of the output, its error is propagated back through the network in such a way, that the parameters of the network can be altered accordingly.

During the backpropagation process, the derivatives for the different parameters in the network are determined, which is needed for the optimization. So backpropagation is the prerequisite of optimization.

1.1.2 Configuration parameters of ANNs

There are several configuration parameters that need to be set for a feedforward network to ensure successful results.

Activation functions

The activation function is analogous to the build-up of electrical potential in biological neurons, which is "fired" when the so-called activation potential is reached. This behaviour is mimicked by the artificial neural network with the application of probability. So a neural network without any activation function is actually a linear regression model, which is limited in the set of functions it can approximate. The selection of the activation function can greatly alter how the firing occurs in the network. The activation function should do two things:

- ensure non-linearity,
- ensure that gradients remain large through the hidden unit.

To perform backpropagation on the network, the activation function is required to be differentiable, so the gradients of error (loss) can be calculated with respect to the weight,s which are then updated using gradient descent.

Loss functions

Several functions can be used to estimate the error of a set of weights in a neural network. However, a function where the space of candidate solutions maps onto a smooth (but high-dimensional) landscape, on which the optimization algorithm can reasonably navigate via iterative updates to the model weights, is preferred [5].

Optimization methods

During the training of the artificial neural network, the goal is to decrease the loss with each epoch. This can be achieved by finding the minimum of the loss function with the optimization of the weights in the network. To achieve this, different optimization methods can be utilized.

Initialization methods

The initialization of network weights is an important and often overlooked characteristic of developing neural networks. Poor initialization of network weights can be the source of many issues which can deteriorate the performance. Because of the inherent way the gradient updates are calculated, a model initialized with all zeroes would learn nothing, as the weights would stay zeroes.

Feature normalization

Feature normalization involves normalizing features before applying the learning algorithm. This is the rescaling of the feature generally done during the preprocessing. According to Ioffe and Szegedy [6], gradient descent converges much faster with feature scaling than without it.

Feature standardization

With feature standardization, the values of each feature in the data will have zero-mean (when subtracting the mean in the numerator) and unit-variance. This method is widely used for normalization in many machine learning algorithms (mostly those that involve distance-based methods). The general method of calculation is to subtract the mean of each feature from the actual value and divide the result by the standard deviation of the given feature.

1.2 Application of ANNs in drilling problems prediction

It is drilling a well that accounts for most of the investments in the oil and gas industry. Thus, it is crucial to avoid any complications, accidents during the construction of a well. Predicting these problems some time ahead they would occur may save a lot of money and reduce non-productive time substantially, as it allows a proactive reaction rather than remediating the occurred problem, which is more than often not successful. The prevailing trend in the century is using ANNs to predict such problems. Borozdin et al. [7] summarized the drilling problems and assigned a value to them based on the possibility of using a neural network to predict them.

Most of the drilling problems that occur during drilling are stuck pipe, lost circulation, and gas, water, or oil kicks. Thus, this work focuses on these problems regarding the applicability of ANNs.

In this decade, several works were devoted to predicting stuck pipes. Ferreira et al. [8] developed an automated decision support algorithm to avoid drilling operations. This earlier work compared real-time data with historical data and required an engineer to detect the cases when action needed to be taken. Naraghi et al. [9] used an active learning method (ALM) to predict the probability of the drillstring being stuck using the surface mechanical parameters of 150 drilled wells. Salminen et al. [10] developed a model that compared the real-time data with the expected trends calculated using torque and drag software and trend analysis. This way, they were able to predict stuck pipe events with sufficient time ahead to prevent them.

Murillo et al. [11] used adaptive fuzzy logic and ANNs to predict stuck pipe incidents. One hundred eighty-five data sets were generated from drilling and mud reports that consisted of the measured and vertical depths, GPM, WOB, RPM, BS, drillcollar length, ROP, torque and drag, chloride filtrate, PV, YP, MW, and gel strength. To reduce the number of variables, dimensionless groups were introduced. Discriminant analysis was used to produce discriminant functions as output curves. 75% of the data was used to train the ANN, while 25% was used to test it. Using the ANN introduced less error in predicting the occurrence of the sticking than the fuzzy logic model.

Jahanbakshi et al. [12] used a dataset of 214 samples that were divided into a 70:30 ratio randomly to train and test their ANN. The data included mud properties, BHA

length, seconds the pipe was still, and different hole sizes. In their work, a feed-forward backpropagation model was used with one hidden layer. After optimization, it was found that the ANN provided the best results with the transit transfer function and 18 neurons in the hidden layer. The ANN showed 82.15% accuracy in predicting sticking cases.

Alshaikh et al. [13] compared three machine learning models to predict stuck pipe, namely decision trees, support vector machines, and ANNs. In their work, a dataset of 9 historical stuck pipe occurrences was used. The parameters contained were the surface drilling parameters, their average of the previous 12 timesteps, and the rate of change of the given parameter. The output was the probability of being stuck. According to their results, which were validated using nested cross-validation, the ANN was 96.88% accurate in predicting stuck pipe incidents with a precision of 94.28%, which means that in 6 out of 100 cases, there was a false alarm.

2 Methods

There are multiple reasons to pull the drillstring out of the hole and to run a new assembly, namely:

- reaching the total planned depth,
- changing BHA (running new rotary assembly, changing malfunctioning downhole tools, bits called bittrip or roundtrip),
- hole conditioning before cementing and open-hole geophysical measurements (it involves pulling out of the hole and running in again, then circulating mud).

Tripping in and out of the hole may pose several risks, downhole problems that can be traced back to the condition of the well, mud, and mudcake as well as the well geometry and well stability. The most common issue is the pipe becoming stuck in the well. The sticking mechanisms include the following:

- Loose or unconsolidated formations collapsing into the borehole and packing off the drillstring.
- Differential sticking due to high-pressure difference and/or thick mudcake.
- Mobile formations behaving in a plastic manner, squeezing into the wellbore.
- Reactive formations swelling into the wellbore.
- Drillstring vibration causing caving that packs off the drillstring.
- Keyseating occurs when the rotating drillpipe wears a groove into the borehole wall making tripping of the larger diameter tools out of the hole challenging.
- Under-gauge holes develop when the bit starts to wear. Running a new bit poses the risk of jamming in the under gauge section.
- Hole cleaning problems preventing the removal of cuttings from the borehole, packing off the drillstring.

All these sticking mechanisms have their early warning signs. This can be avoided or successfully mitigated. To do so, the drilling and formation parameters must be closely monitored. As there is a connection between the formation and the surface mechanical drilling parameters the latter can be used to approximate the formation parameters. Several authors worked hard to develop mathematical models to describe the drilled formations based on the surface drilling parameters with limited success. This is where the power of the ANNs proves very useful.

The aim of this section is to develop a feed-forward artificial neural network that can recognize critical points of an open-hole section, where some kind of sticking may occur during tripping in and out of the hole. The neural network should also be able to provide an early warning sign to prolong the time window for any reaction to make. To develop the network, the surface mechanical drilling parameters are used exclusively which can be monitored at all times at the rig site:

- Measured depth (MD)
- Rate of penetration (ROP),
- Weight on bit (WOB),
- Revolution per Minute (RPM),
- Flow rate (FR),
- Torque (TQ),
- Standpipe pressure (SPP),
- Mud weight in and out (MW),
- Mud temperature in and out (TMP),
- Total gas content (TGAS).

Using these surface drilling parameters, the artificial neural network is trained and tested on the data of 3 drilled wells. 65% of the data is used to train and 35% is used to test the neural network. As a result, the ANN will be able to point out critical points along the trajectory of the well, where the stuck pipe may occur during tripping in and out, also providing an early warning sign for the driller to react faster.

2.1 Development environment

The presented development is done in Java language, with the use of the deeplearning4j library. When considering large-scale server-side applications, Java is the most favored, the deeplearning4j library makes it possible to develop artificial neural networks in Java or to import (even retrain) models from Pytorch, Tensorflow, or Keras and deploy them in JVM Microservice environments, mobile devices, IoT, and Apache Spark.

2.2 ANN development

During the development process of the neural networks, an iterative approach was applied, which included the generation and the training of several networks with a static, common base configuration and changing “case” configuration. The changing configuration contained the number of hidden layers (1, 2

,3), the overall number of hidden neurons (12 and 15), the number of training epoch (200, 400, 600, 800), the size of data batches(30, 50, 100, 250, 500), the learning rate (1e-4, 5e-4, 1e-3, 5e-3) and weight decay (1e-7, 5e-7, 1e-6, 5e-6, 1e-5, 5e-5, 1e-4) applied for the optimization function. This resulted in 3360 models for both trip-out and trip-in procedures.

The applied activation function was ReLU in the hidden layers, sigmoid in the output layer, and hyperbolic tangent in the input layer.

ReLU is the simplest non-linear activation function, which performs well in most applications, as it avoids and rectifies the vanishing gradient problem. Most of the deep learning models use ReLU nowadays. However, ReLU should only be used within hidden layers of a neural network. For the output layer, the activation function should be sigmoid for binary classification, hyperbolic tangent for multiclass classification, and linear for a regression problem.

The zero-centeredness issue of the sigmoid function can be resolved by using the hyperbolic tangent function. Because of this, the hyperbolic tangent function is always preferred to the sigmoid function within hidden layers. However, the hyperbolic tangent still suffers from the other problems plaguing the sigmoid function, such as the vanishing gradient problem.

Sigmoids suffer from the vanishing gradient problem. They are not zero-centered; gradient updates go too far in different directions, making optimization more difficult. Sigmoids saturate and kill gradients and also have slow convergence. Sigmoids are still used as output functions for binary classification but are generally not used within hidden layers. A multidimensional version of the sigmoid is known as the softmax function and is used for multiclass classification.

The applied loss function was mean squared logarithmic error. The adaptive moment estimation method was applied as the optimization method. Adaptive Moment Estimation (Adam) [14] is a method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients. Adam also keeps an exponentially decaying average of past gradients similar to momentum. Whereas momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface.

The applied initialization method was the Xavier method, which is a simple heuristic for assigning network weights. With each passing layer, the variance should remain the same. This keeps the signal from exploding to high values or vanishing to zero. So the weights should be initialized in such a way that the variance remains the same for both the input and the output. The weights are drawn from a distribution with zero mean and a specific variance. The normalization and standardization are done by the default data initialization module of deeplearing4j.

2.3 Model evaluation

During the training process, only the first well's data was used, as it had the most problems during the trip in and trip out processes. The data of the two other wells were used as a validation check to give feedback on how well a network can be utilized for new wells.

For the evaluation of the generated and trained models, so-called confusion matrices were used. Each row in the matrix represents the instances in an actual class while each column represents the instances in a predicted class (Figure 7):

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Figure 7 Confusion matrix

With the parameters of the confusion matrix the following indicators (2), (3), (4), (5) are calculated:

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

$$F1\ score = \frac{2TP}{2TP + FP + FN} \quad (4)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

The F1 score is the harmonic mean of precision and sensitivity.

The evaluation process consisted of the following steps:

1. Eliminate the models which gave a worse than 0.5 value for any of the indicator parameters on the training dataset.
2. Search for the models which resulted in the best indicator parameter values on the training dataset.
3. Search for the models which resulted in the best indicator parameter values for the validation datasets.
4. Eliminate models, where overfitting is indicated (indicator parameters have high values in case of training dataset and low values in case of training dataset).
5. For each generated model calculate the ratio of the calculated and best values for each indicator parameter.
6. Select the model with the best calculated / best ratio values.

PROBLEM PREDICTION DURING TRIP IN AND TRIP OUT PROCEDURES WITH ARTIFICIAL NEURAL NETWORKS

Ádám Viktor Pásztor; Richárd Ürmös

Unfortunately, the amount of data for the third well in case of build-in problems was not sufficient for the evaluation.

3 Results

The best model for the trip-out procedure is summarized in Table 1. The high ratio values in the case of the two validation datasets suggest, that the model gave exceptionally good results compared to all the built models. On top of that, the actual values are also high (averaging above 0.8), which means, that the model performs well outside of the training dataset. The somewhat low recall ratio in the case of the training dataset is because of an overfitted model, which resulted in a recall value of 1 for the training set but very low values for the validation sets. The network has the highest hidden layer number and hidden neuron number among the provided options, which indicates high nonlinearity between the input and output parameters. Both the batch size and the epoch number are just the second highest within the given range. This is an indication of the optimal setup because neither the increase nor the decreasing of training cycles or training batches would improve the performance of the model.

Table 1 Best model for trip-out procedure

Trip out procedure		
Layer number	3	
Hidden neurons	5	
Epochs	600	
Batch Size	250	
Learning rate	1,E-03	
Weight decay	1,E-06	
Training Dataset		
Indicator	Actual	Ratio
Accuracy	0,9112	0,980312
Precision	0,8579	0,895886
Recall	0,7253	0,7253
F1 Score	0,786	0,941881
First Validation Dataset		
Indicator	Actual	Ratio
Accuracy	0,8502	0,980312
Precision	0,8147	0,895886
Recall	0,8293	0,7253
F1 Score	0,8219	0,941881
Second Validation Dataset		
Indicator	Actual	Ratio
Accuracy	0,9122	0,939463
Precision	0,908	0,909606
Recall	0,6781	0,78921
F1 Score	0,7764	1

The best model for the trip-in procedure is summarized in Table 2. As it was stated before, in the case of the third well, there were not enough positive occurrences (which is fortunate from the perspective of the drilling procedure) for the evaluation. The presented model provided the best average ratio values for both the training and validation datasets. Unfortunately, the actual values are low especially in the case of the precision and F1 score, which means that the model overpredicted the number of positive occurrences for the validation set. The reason for this could be the relatively low occurrence of failure during the build-in procedures of the second well. The batch size, the learning rate, and the weight decay are all one of the extreme values, which indicates, that the optimal model was not found.

Table 2 Best model for trip-in procedure

Trip out procedure		
Layer number	2	
Hidden neurons	6	
Epochs	600	
Batch Size	30	
Learning rate	5,E-03	
Weight decay	1,E-04	
Training Dataset		
Indicator	Actual	Ratio
Accuracy	0,9739	0,993066
Precision	0,931	0,931
Recall	0,8882	1
F1 Score	0,9091	0,977948
Validation Dataset		
Indicator	Actual	Ratio
Accuracy	0,6945	0,729364
Precision	0,1047	0,985876
Recall	0,7143	0,7143
F1 Score	0,1826	1

4 Conclusion

The presented results show that the described methodology is sufficient to develop such neural networks which can predict the possible problems during the trip-in and trip-out procedure at an acceptable level.

The indicator parameters reached high values in the case of both procedures considering the training dataset. The significance of this result lies in the fact that the drilling procedure consists of a series of build-in and out sessions. With every session, the accuracy of a neural network can be improved, so each session will be safer than the previous.

With the evaluation of the validation dataset, the applicability of a pre-trained model is tested. For the trip-out procedure, the results were really promising, the

indicator parameters showed accuracy levels way above the acceptable level.

Acknowledgments
SUPPORTED BY THE ÚNKP-20-4 NEW NATIONAL EXCELLENCE PROGRAM OF THE MINISTRY FOR INNOVATION AND TECHNOLOGY FROM THE SOURCE OF THE NATIONAL RESEARCH, DEVELOPMENT AND INNOVATION FUND.

References

[1] Structure of biological neurons, [Online], Available: https://i.pinimg.com/474x/05/0d/29/050d29215939e961f2aa2808f5a7432f--neurons-nervous-system.jpg [28 Aug 2021], 2021.
[2] LECUN, Y., BENGIO, Y., HINTON, G.: Deep learning, *Nature*, Vol. 521, No. 7553, pp. 436-444, 2015.
[3] Model of a neural network, [Online], Available: https://i0.wp.com/thaddeus-segura.com/wp-content/uploads/2020/08/neural-ent.jpeg?resize=493%2C241&ssl=1 [28 Aug 2021], 2021.
[4] Working model of a node, [Online], Available: https://wiki.pathmind.com/images/wiki/perceptron_node.png [28 Aug 2021], 2021.
[5] STEWART, M.: Intermediate Topics in Neural Networks, [Online], Available: https://towardsdatascience.com/comprehensive-introduction-to-neural-network-architecture-c08c6d8e5d98 [28 Aug 2021], 2019.
[6] IOFFE, S., SZEGEDY, C.: *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shif*, arXiv, 2015.
[7] BOROZDIN, S., DMITRIEVSKY, A., EREMIN, N., ARKHIPOV, A., SBOEV, A., CHASHCHINA-SEMENOVA, O., FITZNER, L., SAFAROVA, E.: *Drilling Problems Forecast System Based on Neural Network*, SPE Annual Caspian Technical Conference, 2020.
[8] FERREIRA, A.P.L.A., CARVALHO, D.J.L., RODRIGUES, R.M.: *Automated Decision Support and Expert Collaboration Avoid Stuck Pipe and Improve Drilling Operations in Offshore Brazil Subsalt Well*, Houston, 2015.
[9] NARAGHI, M.E., EZZATYAR, P., JAMSHIDI, S.: Prediction of Drilling Pipe Sticking by Active Learning Method (ALM), *Journal of Petroleum and Gas Engineering*, Vol. 4, No. 07, pp. 173-183, 2013.
[10] SALMINEN, K., CHEATHAM, C., SMITH, M., VALIULLIN, K.: Stuck-Pipe Prediction by Use of Automated Real-Time Modeling and Data Analysis, *SPE Drilling & Completion*, Vol. 32, No. 03, pp. 184-193, 2017.
[11] MURILLO, A., NEUMAN, J., SAMUEL, R.: *Pipe Sticking Prediction and Avoidance Using Adaptive Fuzzy Logic Modeling*, Oklahoma City, USA, 2009.
[12] JAHANBAKHSHI, R., KESHAVARZI, R., ALIYARI SHOOREHDELI, M., EMAMZADEH, A.: Intelligent Prediction of Differential Pipe Sticking by Support Vector Machine Compared With Conventional Artificial Neural Networks: An Example of Iranian Offshore Oil Fields, *SPE Drilling & Completion*, Vol. 27, No. 04, pp. 586-595, 2012.
[13] ALSHAIKH, A., MAGANA-MORA, A., GHARBI, S.A., AL-YAMI, A.: *Machine Learning for Detecting Stuck Pipe Incidents: Data Analytics and Models Evaluation*, Beijing, 2019.
[14] KINGMA, D.P., LEI BA, J.: *Adam: A Method for Stochastic Optimization*, International Conference on Learning Representations, pp. 1-15, 2015.

Review process
Single-blind peer review process.