-RESEARCH PAPER-

International Journal of Mining and Geo-Engineering

IJMGE

IJMGE 57-4 (2023) 389-396

DOI: 10.22059/IJMGE.2023.356428.595045

Hydrocarbon reservoir potential mapping through permeability estimation by a CUDNNLSTM deep learning algorithm

Behnia Azizzadeh mehmandost olya [a], Reza Mohebian [a, *]

[a] *School of Mining Engineering, College of Engineering, University of Tehran, Tehran, Iran.*

Article History:
Received: 06 March 2023.
Revised: 19 May 2023.
Accepted: 08 June 2023.

ABSTRACT

Potential mapping of Permeability is a crucial factor in determining the productivity of an oil and gas reservoirs. Accurately estimating permeability is essential for optimizing production and reducing operational costs. In this study, we utilized the CUDNNLSTM algorithm to estimate reservoir permeability. The drilling core data were divided into a training pool and a validation pool, with 80% of the data used for training and 20% for validation. Based on the high variation permeability along the formation, we developed the CUDNNLSTM algorithm for estimating permeability. First, due to the highly dispersed signals from the sonic, density, and neutron logs, which are related to permeability, we adjusted the algorithm to train for 1000 epochs. However, once the validation loss value reached 0.0158, the algorithm automatically stopped the training process at epoch number 500. Within 500 epochs of the algorithm, we achieved an impressive accuracy of 98.42%. Using the algorithm, we estimated the permeabilities of the entire set of wells, and the results were highly satisfactory. The CUDNNLSTM algorithm due to the large number of neurons and the ability to solve high-order equations on the GPU is a powerful tool for accurately estimating permeability in oil and gas reservoirs. Its ability to handle highly dispersed signals from various logs makes it a valuable asset in optimizing production and reducing operational costs, because it is much cheaper than the cost of core extraction and has very high accuracy.

Keywords: Potential mapping, Permeability estimation, Deep Learning, CUDNNLSTM, Oil and gas reservoirs.

1. Introduction

Potential maps are an invaluable tool in mineral exploration and geoenergy. These maps possess the ability to highlight points with economic potential, making them highly valuable in the adoption of extractive or production plans. One of the most significant features of these maps is their integration with the geographic information system. In general, potential maps serve as a tool for distinguishing important areas from less important ones. In the field of geoenergy, potential maps can reveal the location of hydrocarbon reserves in two or three dimensions. It is important to note that not all geological zones are capable of production when it comes to the extraction of hydrocarbon reserves from hydrocarbon fields. This issue depends on various factors, including permeability [1, 2].

Permeability estimation is a crucial part of the characterization of porous media, which has applications in a number of fields, including oil and gas reservoirs, geothermal systems, groundwater management, and environmental remediation [3, 4]. It is traditional to conduct laboratory experiments to estimate permeability, but these experiments can be time-consuming, expensive, and not always accurate. As a potential alternative to traditional methods of estimating permeability, the use of machine learning (ML) and deep learning (DL) has been proposed as a potential tool in recent years. As a result of ML and DL, it is possible to significantly reduce the time and cost of estimating permeability and improve the accuracy with which predictions can be made.

The use of ML techniques for estimating permeability has been explored in several studies. An ML model based on neural networks was developed in a study by Wang et al in 2021 in order to predict the permeability of sandstones using a machine learning approach. In training the model, a dataset of petrophysical properties was used, and after training, the model was able to predict permeability values with a high degree of accuracy [5]. There is strong evidence to suggest that ML-based tools can be used to reduce the time and cost associated with estimating permeability, as indicated by the study [6]. Among other methods that rely on neural networks and fuzzy logic is the investigation of thin sections, in which the ability of the neural network instead of the human interpreter indicates the direct effect of semi-automatic methods on determining the permeability value [7].

Furthermore, Feng et al. in 2008 proposed a machine learning model based on support vector regression (SVR) that can be utilized to predict the permeability of sandstones [8]. A model was trained using a dataset of petrophysical properties and proved to be highly accurate when it came to predicting permeability values based on the data. There was a demonstration in the study that the use of machine learning techniques can be used to accurately estimate permeability in geological formations [9]. Using an ML model based on decision trees, Ahmadi and Chen developed a model in 2019 to predict the fractured rock's permeability. They used a decision tree framework to create the ML model. Based on a dataset of micro seismic data, the model was trained and was able to predict permeability values with a high level of accuracy. There was considerable evidence in the study that ML techniques can be used to estimate permeability in complex geological formations in an accurate manner [10].

* Corresponding author. *E-mail address:* mohebian@ut.ac.ir (R. Mohebian).

In addition to DL techniques, permeability estimation has also been explored using DL techniques. According to Singh et al. in 2019, there was a study that proposed a method for estimating the permeability of soils utilizing DL. This method involved training a deep belief network (DBN) based on the well logs and seismic data. In comparison to traditional ML methods, the deep neural network was able to predict permeability values with high accuracy. DL methods demonstrated to be capable of accurately estimating the permeability of geological formations by the study, demonstrating the potential of DL techniques [11].

Furthermore, AL Qahtani et al. in 2018 developed a DL model by integrating a convolutional neural network (CNN) into their DL model in order to predict permeability in shale formations using a DL model based on a convolutional neural network (CNN). In an experiment based on a dataset of digital rock images, the model was trained on a dataset of permeability values and it was able to simulate permeability values with high accuracy. In the study, DL techniques have demonstrated to be capable of accurately estimating permeability in complex geological formations using a variety of approaches [12].

In a follow-up study, a DL model based on a CNN was also proposed by Wang et al. in 2021 for predicting permeability in shale formations. They used a dataset of digital rock images to train the model, which achieved high accuracy in predicting permeability values. In this study, DL techniques were shown to be useful in accurately estimating permeability in geological formations, demonstrating their potential for practical applications [13].

This study aims to identify hydrocarbon-producing zones by estimating permeability using a deep learning algorithm. The algorithm was trained on well log data to predict permeability values at different depths. The resulting permeability estimates were then used to create a hydrocarbon production potential map. The deep learning algorithm employed in this study is a powerful tool for predicting permeability values. It is capable of learning complex relationships between input data and output values, making it well-suited for predicting permeability from well log data. To create the hydrocarbon production potential map, we used the permeability estimates to identify zones with high hydrocarbon production potential. These zones were then mapped in the depth of the well, providing a visual representation of the hydrocarbon potential at different depths.

2. Material

2.1. The principles of permeability calculation in the laboratory on the core

Permeability is defined as the ability of a material to allow fluids or gases to pass through it. It is usually expressed in units of Darcy (D). Darcy's law states that the flow rate (Q) of a fluid through a porous medium is proportional to the pressure difference (ΔP) between the two ends of the medium and the permeability (k) of the medium. The equation can be expressed as below, where A is the cross-sectional area of the medium, and L is its length (Figure 1) [14, 15].

$$Q = kA\left(\frac{\Delta p}{L}\right) \tag{1}$$

There are several methods for measuring permeability in the laboratory, including the constant-head method, the falling-head method, and the pulse decay method. The choice of method depends on the type of material being tested, the level of precision required, and the available equipment. For example, the constant-head method is used for testing highly permeable materials, such as sands and gravels. In this method, a constant pressure difference is maintained across the sample, and the flow rate is measured over time. The permeability is then calculated using Darcy's law. The equation for permeability in the constant-head method is:

$$k = \frac{QL}{A\Delta P} \tag{2}$$

Where Q is the flow rate, L is the length of the sample, A is the cross-sectional area of the sample, and ΔP is the pressure difference [16].

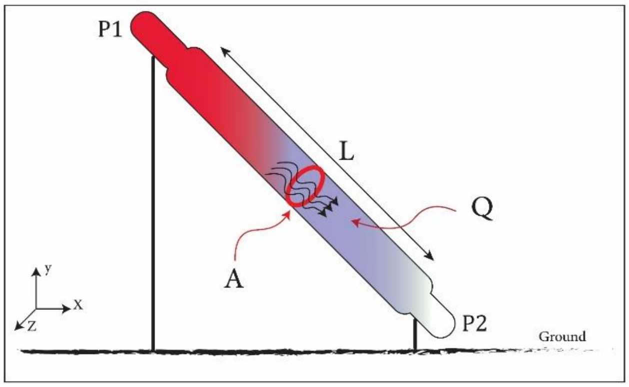

Figure 1. The Darcy law schematic.

2.2. Deep learning Algorithm mathematics and architecture

The recurrent neural network (RNN) is a type of deep learning algorithm that has gained popularity over the years for its use in a wide range of applications, such as natural language processing, speech processing, and time-series analysis. It is widely known that the Long Short-Term Memory (LSTM) algorithm is a type of RNN, which is widely used for modelling sequential data due to its ability to capture long-term dependencies between data points [17, 18]. As a result, LSTM models can be computationally expensive to train when used on a large dataset, which limits their scalability. The CUDNNLSTM algorithm is a parallelized version of the LSTM algorithm that has been designed to run efficiently on graphics processing units (GPUs) and is designed to run quickly. The Long Short-Term Memory (LSTM) algorithm was first introduced in 1997 by Sepp Hochreiter and Jürgen Schmidhuber. The LSTM algorithm was developed as a solution to the vanishing gradient problem, which is a common issue with traditional recurrent neural networks (RNNs) that hinders their ability to learn long-term dependencies in sequential data [19].

The original LSTM architecture had three types of gates: input, forget, and output. These gates allowed the network to selectively retain or discard information at each time step, depending on its relevance to the current task. The LSTM algorithm quickly gained popularity for its ability to learn long-term dependencies in sequential data and was widely used in various fields, including speech recognition, handwriting recognition, and natural language processing.

Over the years, researchers proposed several improvements and variations to the original LSTM architecture. One of the most significant improvements was the Gated Recurrent Unit (GRU) proposed by Kyunghyun and et al in 2014 [20].The GRU architecture had two gates: update and reset, which simplified the LSTM architecture and achieved comparable performance on various tasks. In recent years, there has been a growing interest in using deep learning algorithms for various applications, including natural language processing, computer vision, and speech recognition.

The vanilla RNN architecture (figure 2) was used in the initial implementation of CUDNN; however, vanilla RNNs have been known to have difficulty with long-term dependencies. The Long Short-Term Memory (LSTM) architecture, developed by Google researchers, addresses many of these issues by selectively updating hidden states with gated cells [21].

The development of the CUDNNLSTM algorithm is closely tied to the rise of deep learning and the increasing demand for efficient implementations of recurrent neural networks (RNNs) on graphics processing units (GPUs). A set of optimized primitives for deep learning on GPUs was made available in 2014 by NVIDIA as part of their CUDA Deep Neural Network library (CUDNN) [22]. In addition to optimizing the convolution and pooling layers, RNNs were also implemented as primitives. Speech recognition and machine translation are popular tasks for the LSTM architecture, but its computational requirements make GPU implementation difficult. As a result, NVIDIA released an updated version of CUDNN in 2016 that included an optimized LSTM implementation, known as CUDNNLSTM. By optimizing data layouts, kernel fusions, and dynamic parallelism, CUDNNLSTM takes full advantage of the parallelism offered by GPUs to minimize computation times. By using GPUs instead of earlier implementations, deep LSTM models could be trained and deployed with significantly improved performance.

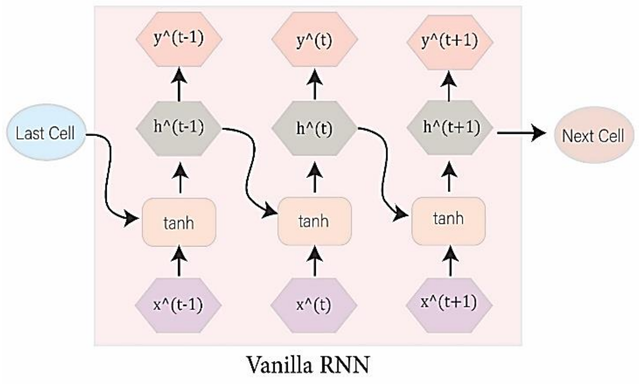

Figure 2. The Vanilla RNN architecture.

CUDNNLSTM uses the Long Short-Term Memory (LSTM) network as a basis for its architecture, a type of RNN that is able to recognize long-term dependencies in sequential data by utilizing memory cells and gates. CUDNNLSTM networks consist of several layers of LSTM cells that are interconnected to each other and to inputs and outputs.

There are several components in each LSTM cell (figure 3):

1. Input gate: Determines which information from the current input should be stored in the cell state.
2. Forget gate: Determines which information from the previous cell state should be forgotten.
3. Cell state: The memory of the LSTM cell that stores information from previous time steps and current input.
4. Output gate: Determines which information from the current cell state should be output.

CUDNNLSTM algorithm in greater detail. Figure 3 shows the state of the gates, but Figure 4 reveals the performance of the CUDNNLSTM algorithm and the state of the gates in more detail.

The input to the LSTM layer at time step t is a vector $X_t \in R^D$, where D is the dimensionality of the input. The LSTM layer maintains an internal state $h_t \in R^H$, where H is the dimensionality of the hidden state, and a cell state $C_t \in R^H$, which stores the long-term memory (table 1). The internal state and the cell state are updated at each time step based on the input and the previous states.

The input gate i_t takes the input X_t and the previous hidden state h_{t-1} and produces a vector of values between 0 and 1 that determines how much of the input should be added to the cell state. The forget gate f_t takes the input X_t and the previous hidden state h_{t-1}, produces a vector of values between 0 and 1 that determines how much of the cell state should be forgotten. The output gate O_t takes the input X_t and the current hidden state h_t, produces a vector of values between 0 and 1 that determines how much of the cell state should be output. The cell gate C_t takes the input X_t, the previous hidden state h_{t-1}, and the previous cell state C_{t-1}, produces a vector of values between -1 and 1 that determines the new cell state.

The updated cell state C_t and hidden state h_t at time step t are calculated as follows:

1. In forger gate:

$$f_t = \sigma\,.(W_f.X_t + U_f.h_{t-1} + b_f) \quad (3)$$

2. In input gate:

$$i_t = \sigma.(W_i.X_t + U_i.h_{t-1} + b_i) \quad (4)$$

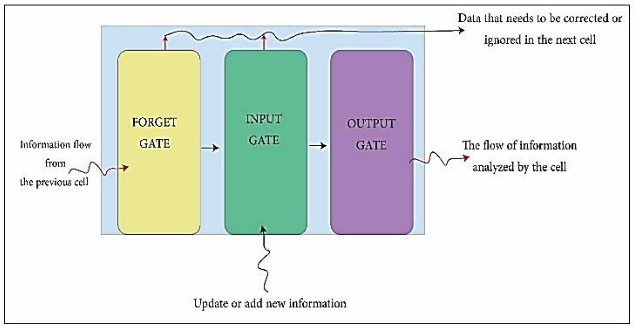
Data that needs to be corrected or ignored in the next cell
Information flow from the previous cell
FORGET GATE
INPUT GATE
OUTPUT GATE
The flow of information analyzed by the cell
Update or add new information

Figure 3. A schematic of the state of the gates in one cell of LSTM/CUDNNLSTM

Table 1. Abbreviations used in expressing equations used in algorithm gates

Decision maker (N-1)	C_{t-1}
Result (N-1)	h_{t-1}
Result (N)	h_t
Decision maker (N)	C_t
Result (N)	h_t
Input data	X_t

3. In cell or LSTM gate

$$C_{t'} = \tanh(W_c.X_t + U_c.h_{t-1} + b_c) \quad (5)$$

4. Update cell state

$$C_t = f_t \times C_{t-1} + i_t \times C_{t'} \quad (6)$$

5. In output gate

$$O_t = \sigma.(W_o.X_t + U_o.h_{t-1} + b_o) \quad (7)$$

6. Update hidden state

$$h_t = O_t \times \tanh(C_t) \quad (8)$$

where σ is the sigmoid function, tanh is the hyperbolic tangent function, and W_f, W_i, W_c, W_o, U_f, U_i, U_c, and U_o are weight matrices, and b_f, b_i, b_c, and b_o are bias vectors. The variables with primes ($C_{t'}$) denote the intermediate cell state before applying the forget and input gates.

In summary, the CUDNNLSTM algorithm is a type of recurrent neural network (RNN) optimized for use on NVIDIA GPUs. It is designed to efficiently process sequential data, such as time series or natural language text, by learning patterns and relationships between the input data and the output predictions. The CUDNNLSTM algorithm is based on the Long Short-Term Memory (LSTM) architecture, which is a type of RNN capable of capturing long-term dependencies in the input data. The CUDNNLSTM algorithm uses a set of learnable parameters, including weights and biases, to transform the input data into a hidden state representation. This hidden state is then used to make predictions about the output data. The algorithm is optimized for use on NVIDIA GPUs by leveraging the parallel processing capabilities of these devices. This allows for faster training and inference times, making it well-suited for use in large-scale machine learning applications. Overall, the CUDNNLSTM algorithm is a powerful tool for processing sequential data and has been used in a wide range of applications, including speech recognition, natural language processing, and image captioning.

2.3. Feed of CUDNNLSTM

A complex interrelationship exists between the physical properties governing permeability in porous media, such as rocks and soils. Density, porosity, and wave speed are the three primary properties that affect permeability. It is possible to predict and optimize permeability under different conditions based on the mathematical relationships among these properties [23, 24].

Density can be expressed mathematically as the mass per unit volume of a porous medium:

$$\rho = (1-\varphi) \times \rho_s + \varphi\rho_f \tag{9}$$

There are three density factors in this equation: ρ is the density of the medium, ρ_s is the density of the solid phase, ρ_f is the density of the fluid phase, and φ is the porosity of the medium. According to the equation above, porosity is the percentage of void space within a medium and is related to density. As porosity increases, the available space for fluid flow also increases.

In a porous medium, wave speed is related to density and porosity through the bulk modulus by excremental equation, which is a measure of the medium's resistance to compression.

$$V = (\frac{K}{\rho})^{0.5} \tag{10}$$

In this equation, v is the speed of waves through the medium, K is the bulk modulus, and ρ is the density of the medium.

On the basis of porosity and other physical properties of the medium, empirical equations can be used to estimate permeability, such as the Kozeny-Carman equation. Porous media can be theoretically measured and validated using experimental techniques, such as CT scanning and permeameters. To understand and predict a porous medium's permeability, it is essential to understand the interrelationships between density, porosity, and the speed of waves passing through it. Based on the measured physical properties of the medium, empirical equations can be used to estimate permeability. Mathematical relationships between these properties have been extensively studied. As mentioned, the relationship between sonic, density, and neutron logs and permeability has been established through various empirical relationships. However, in this study, we sought to develop a more robust approach for estimating permeability using these logs. To achieve this, we first normalized the data by converting all logs into a range between -1 and 1. This was necessary due to variations in log changes that made them incomparable on the same scale. After normalizing the logs, we compared their new values against the core permeability data. After fitting a line to the data, we found that the sonic, density, and neutron logs exhibited the highest correlation with permeability data. Based on these findings, we used these three logs as inputs to an algorithm for estimating permeability. By this approach, we were able to develop a more accurate and reliable method for predicting permeability in subsurface reservoirs. After fitting a line to all the logs in the full set (Figure 5), the density, neutron, and sonic logs had correlation coefficients of 0.90, 0.83, and 0.73, respectively. These three coefficients had the highest values among all the other correlation coefficients.

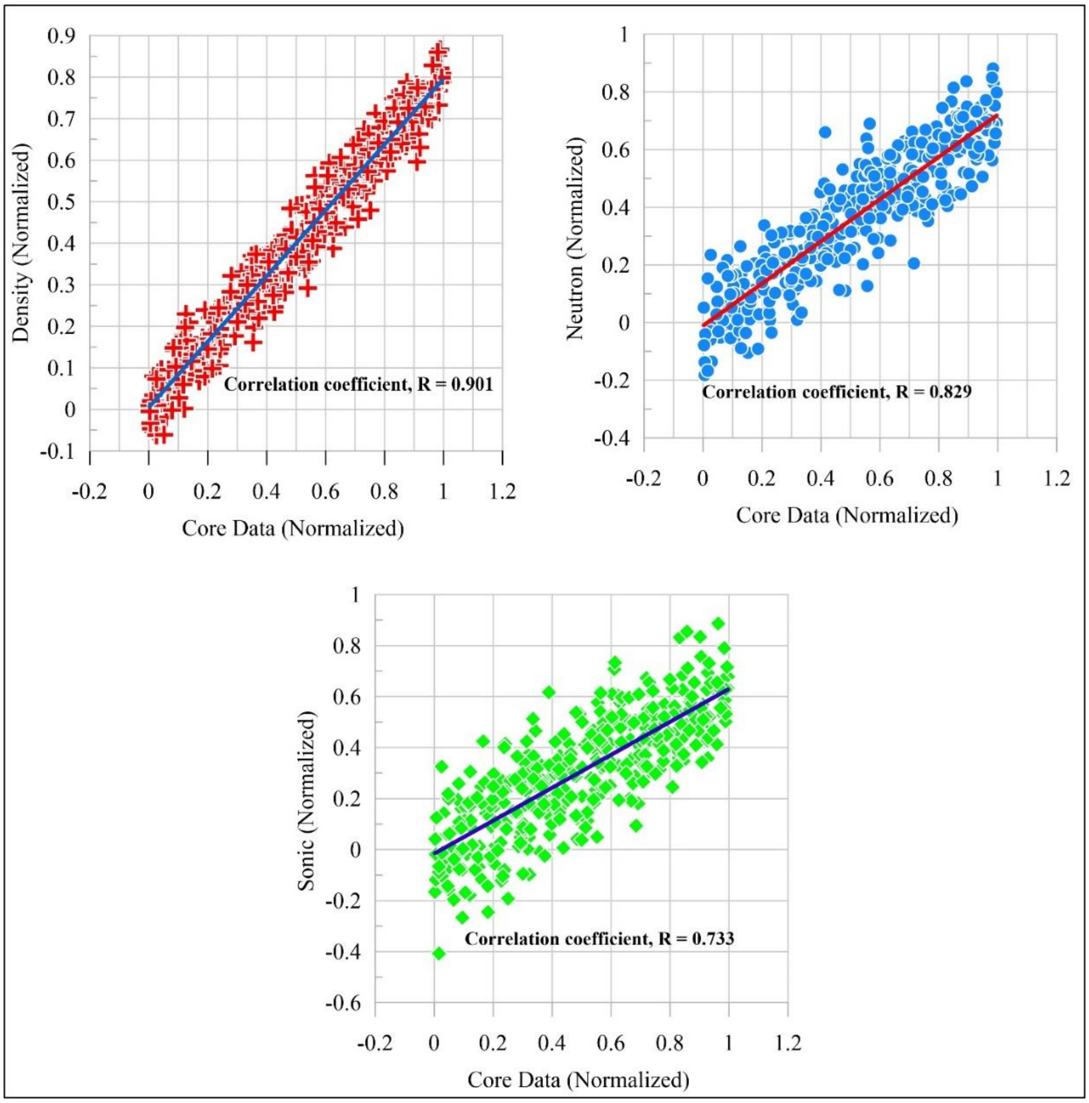

Figure 5. Correlation coefficient of neutron, sonic, and density logs with core permeability data. (All data are normalized between -1 and 1).

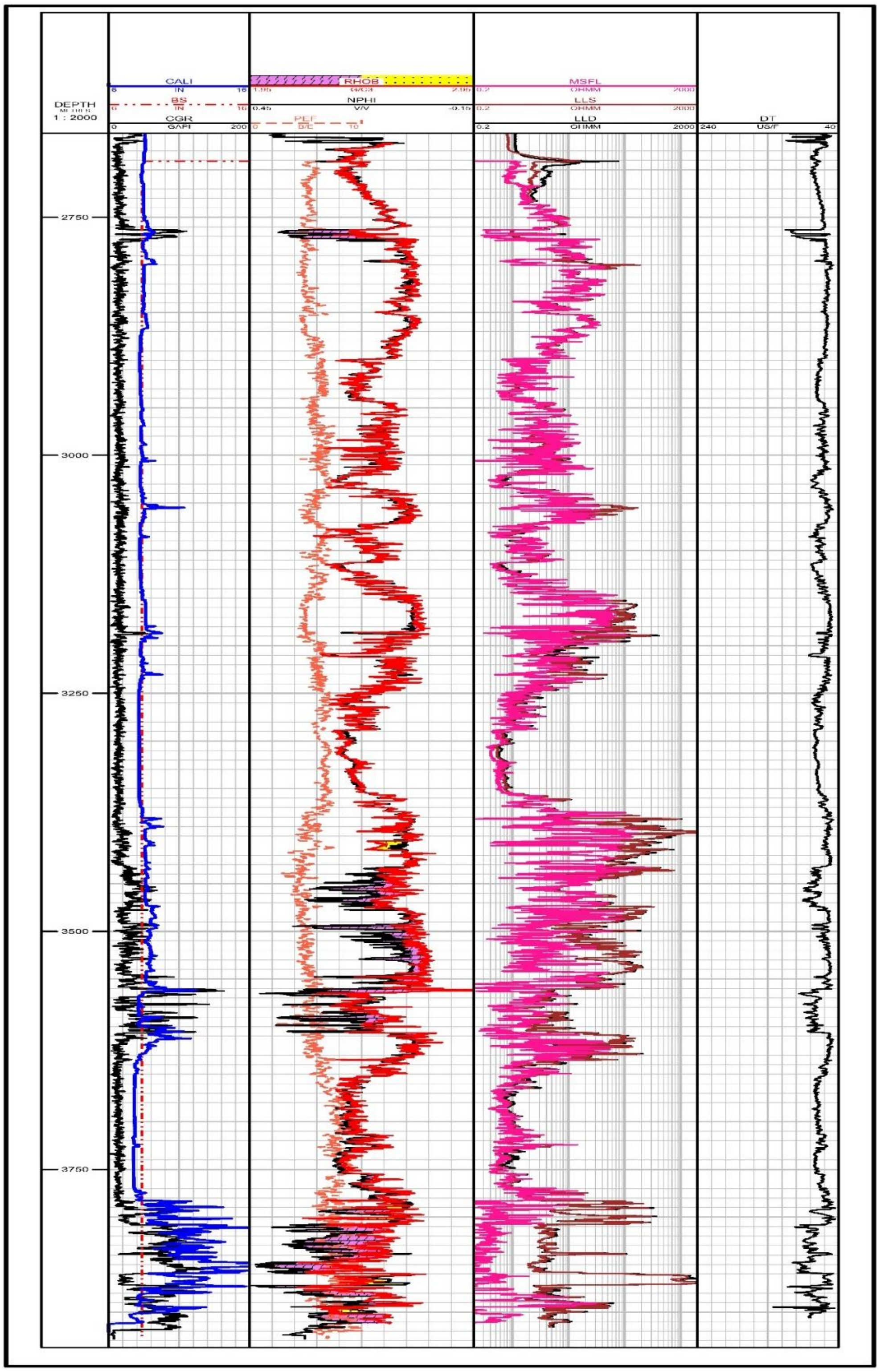

Figure 6. The Full-Set of data which sonic, density and neutron logs were used.

3. Algorithm design for permeability estimation

This algorithm (CUDNNLSTM) is designed to estimate time series data with high dispersion using several key steps (table 2). Here is a detailed explanation of the algorithm and its optimizer:

1. Data Pre- preprocessing: To ensure the accuracy of the logs, we took great care in the pre-processing stage. We carefully examined the data and removed any invalid or unreliable information. For instance, we removed the data of washout, log tails, and spike's locations since they can distort the readings and lead to inaccurate results. After

Table 2. Algorithm Code.

```
1.  Define the look-back window size n.
2.  Load the time series data x_t.
3.  Normalize the data to obtain x̂_t using a min-max scaler.
4.  Split the data into training and testing datasets. Create the training dataset (X_train, Y_train) and the testing dataset (X_test, Y_test) by sliding a window of
    size n over the normalized data x̂_t.
5.  Define the CuDNNLSTM model with LSTM units and a dense layer with a single output.
6.  Train the model by minimizing the mean squared error (MSE) between the predicted and actual output using the Adam optimizer:

●   Initialize the model weights
●   For each epoch i in the range 1, ..., N:
        ●   Shuffle the training dataset
        ●   For each mini-batch of size m in the training dataset:
                ●   Compute the gradients of the loss with respect to the model parameters using backpropagation through time (BPTT)
                ●   Update the model parameters using the Adam optimizer
●   Evaluate the model on the training and testing datasets by computing the MSE.
●   Fine-tune the model if necessary, by adding more layers, changing the number of units, or modifying the learning rate and other hyperparameters
●   . Make predictions on new data x'_t.
●   Normalize the new data to obtain x̂'_t.
●   Create the new dataset (X_new, Y_new) by sliding a window of size n over the normalized new data x̂'_t.
●   Predict the output ŷ_new using the trained model and the input X_new. Evaluate the predictions on the new data by computing the MSE.
```

removing the data with these conditions, we filled the resulting gaps using the average method to ensure that there were no missing values in the data. We also de-spike the sonic log values to eliminate any sudden, extreme changes that could have been caused by noise or other factors. Moreover, we applied environmental corrections to all three logs to account for any variations in the well's conditions. This step was crucial to ensure that the logs accurately reflected the subsurface properties of the well. We took into consideration factors such as temperature, pressure, and salinity, among others, to make the necessary adjustments. Overall, our pre-processing efforts were aimed at producing high-quality logs that could be used for further analysis and interpretation. We also corrected the depth of other logs using gamma log data. The reason for using the gamma log is that we had laboratory log gamma data from core data, which allowed us to compare the core data with the log data.

2. Data preprocessing: The input data (sonic, density, and neutron loges) are first normalized between -1 and 1 using the MinMaxScaler function from the sklearn preprocessing module. To ensure that the algorithm can properly process the input data, this step ensures that the data is within a consistent range.

3. Splitting the data: The normalized data is then split into training and testing datasets using a specified ratio. The model is then tested on unseen data to determine how accurate it is.

4. Creating training and testing datasets: A function is then created to generate training and testing datasets with a specified look-back window. The algorithm will use this window to determine how many previous time steps it will consider when making its predictions. A crucial step in the algorithm's learning process is identifying patterns and relationships between inputs and outputs. The core data is used as training data and the other part serve as test data in a ratio of 8:10.

5. Building the CUDNNLSTM model: The CUDNNLSTM model is then created with an input shape of (look_back, 3) and an output shape of (1). This model is a type of recurrent neural network that is optimized for GPU processing. This makes it an ideal choice for processing large amounts of time series data with high dispersion.

6. Compiling the model: The model is compiled with mean squared error loss and the Adam optimizer. The mean squared error loss function is used to evaluate the performance of the model's predictions. The Adam optimizer is a popular optimizer used in deep learning algorithms that uses adaptive learning rates to improve training efficiency.

7. Training the model: During training, the model uses the training dataset to update the weights in the neural network. The performance of the model is evaluated on both the training and testing datasets using mean squared error and root mean squared error metrics. These metrics are used to evaluate the accuracy of the model's predictions.

8. Optimizing with the Adam optimizer: The Adam optimizer uses a combination of momentum and gradient descent techniques to update the weights in the neural network during training. It computes individual adaptive learning rates for each weight based on the historical gradients for that weight. This helps the optimizer to converge to a better solution faster than other traditional optimization algorithms.

9. Fine-tuning the model: After training the model, the performance is evaluated on the testing dataset. If the performance is not satisfactory, the model can be fine-tuned by adjusting the hyperparameters, such as the number of neurons in the CUDNNLSTM layer or the learning rate of the optimizer.

10. Making predictions: Once the model is trained and fine-tuned, it can be used to make predictions on new data. The input data is preprocessed and fed into the model to obtain the predicted output.

11.Evaluating the predictions: The accuracy of the predictions can be evaluated using metrics, such as mean squared error or root mean squared error. These metrics help determine how well the model is able to estimate time series data with high dispersion.

The CUDNNLSTM algorithm and its optimizer are carefully designed to ensure accurate and efficient estimation of time series data with high dispersion. Using adaptive learning rates, the optimizer improves the training efficiency of the model. As a result, the CUDNNLSTM is the perfect choice for applications that require extensive time series data analysis.

4. Result

As shown in Figure 5, we can see the parts of the formation where the drilling core data is available. In our research, we separated 80% of the data and put it in a training pool, and we used the remaining 20% as a validation pool. Finally, after observing the excellent results of the total permeability along the formation, we expanded and estimated it using the CUDNNLSTM algorithm.

As the signals received from the sonic, density, and neutron logs are highly dispersed, we adjusted the algorithm that we described previously to be trained for 1000 epoch. However, when the validation loss value reached 0.0158 and after that, the amount of validation loss increased rapidly, and the algorithm automatically paused the training process because the training process had been completed. As a result, it was able to reach an accuracy of 98.42% within 500 epoch of the algorithm (Figure 6).

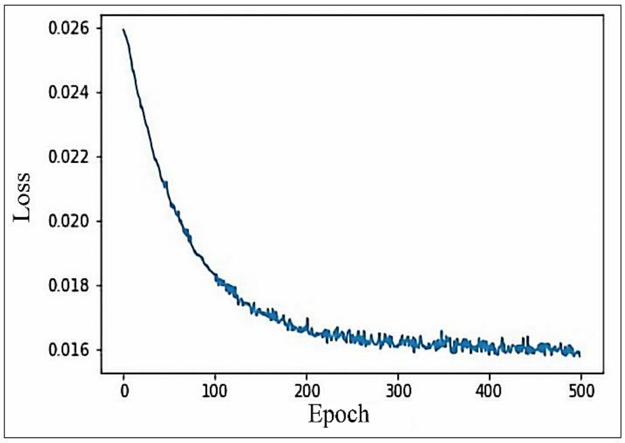

Figure 7. Amount of validation loss in each epoch.

Using the test data from the drilling cores, we estimated the permeability after training the algorithm in order to evaluate its success in estimating permeability. To measure the difference between two data, the mean squared error (MSE) criterion was used, and its value was equal to 1.203. (Figures 7 & 8).

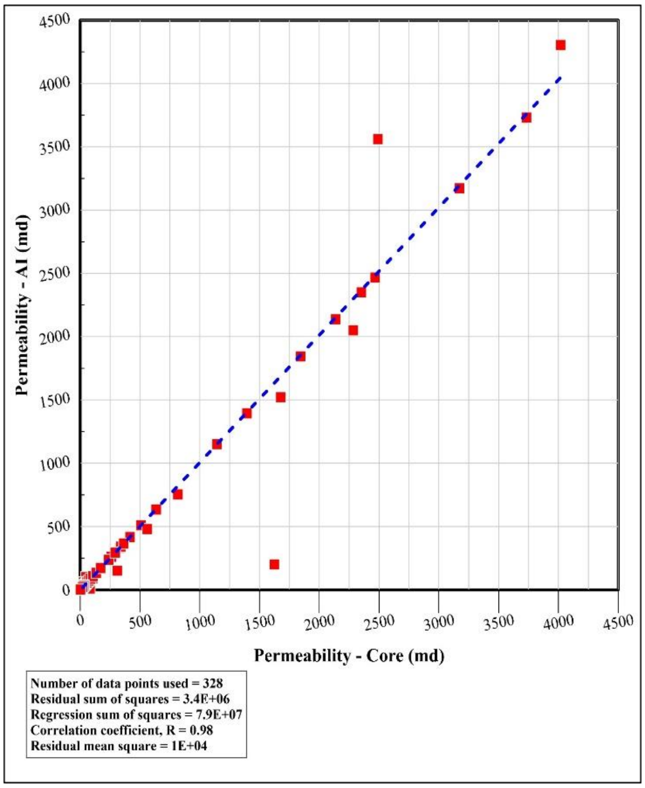

Figure 8. The result of permeability estimation by the CUDNNLSTM algorithm against Core values.

This time, we applied the algorithm to the entire set of wells (figure 9) in order to estimate their permeabilities after seeing the excellent and satisfactory results in Figures 7 and 8.

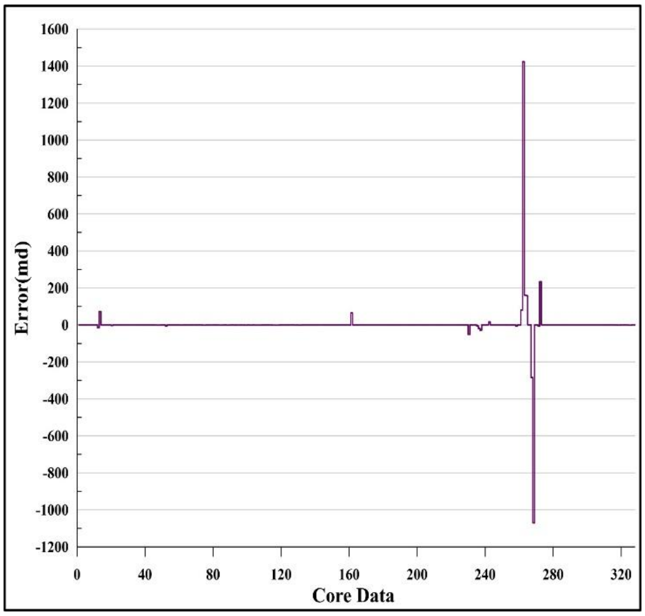

Figure 9. The error rate between the estimated data and the core data.

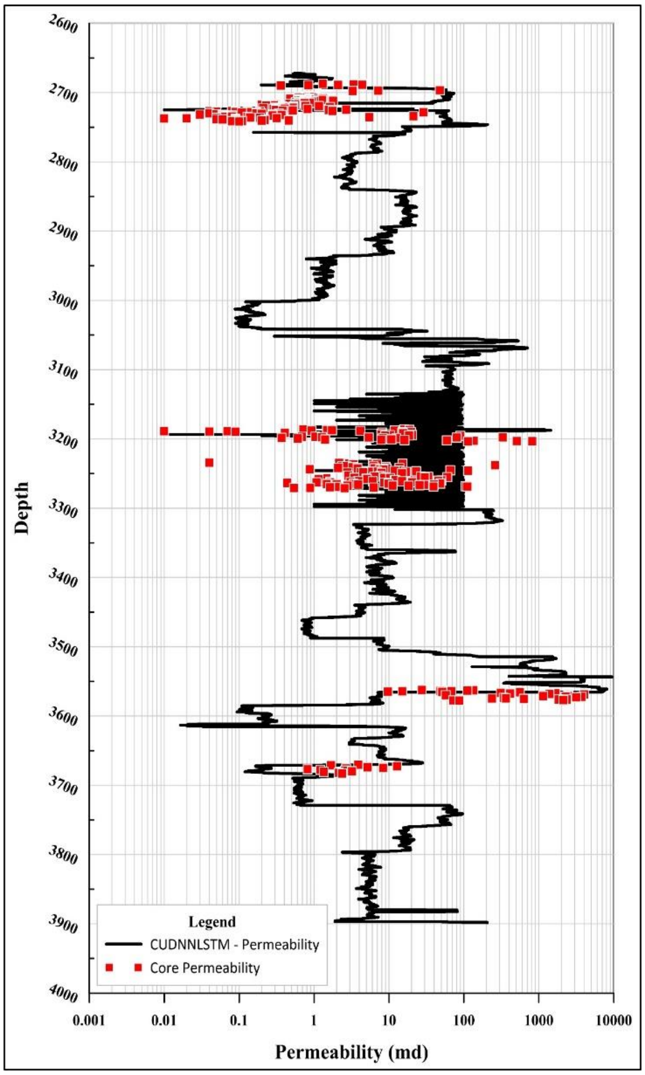

Figure 10: The result of permeability estimation by the CUDNNLSTM algorithm.

5. Conclusion

A powerful deep learning algorithm, CUDNNLSTM, has demonstrated excellent performance in a variety of applications. To predict the next numerical value in a series, we applied the CUDNNLSTM algorithm to a dataset of numerical values. According to our results, the CUDNNLSTM algorithm achieved 98.4% accuracy and 0.158 validation loss. Consequently, the algorithm accurately predicted the next number in a series of numbers. Due to its ability to learn complex patterns and relationships in the dataset, the CUDNNLSTM algorithm achieves high accuracy. By processing large amounts of data quickly and efficiently, the algorithm can identify key features and patterns that are crucial to accurate predictions.

In addition, the algorithm's low validation loss value indicates that it is not overfitting to the training data. Predictive modelling, where the goal is to predict future values accurately based on historical data, is especially vulnerable to overfitting, which can lead to poor performance when dealing with new and unseen data. As a result of our study, the CUDNNLSTM algorithm was demonstrated to be extremely effective in predicting the next value in a series of numbers. There is potential for further improvements in predictive modelling tasks based on these results for the field of deep learning.

REFERENCES

[1]. M. Arab Amiri, M. Karimi and A. Alimohammadi, "Hydrocarbon resources potential mapping using the evidential belief functions and GIS, Ahvaz/Khuzestan Province, southwest Iran," Arabian Journal of Geosciences, vol. 8, no. 6, pp. 1-13, 2014.

[2] M. Badawy, T. Abdel Fattah, S. Abou Shagar, A. Diab, M. Rashed and M. Osman, "Identifying the hydrocarbon potential from seismic, geochemical, and wireline data of the Sallum intra-basin, North Western Desert of Egypt," NRIAG Journal of Astronomy and Geophysics, vol. 12, no. 1, pp. 1-18, 2022.

[3] . C. W. Spencer, "Review of characteristics of low-permeability gas reservoirs in western United States," AAPG, vol. 73, no. 5, pp. 613-629, 1989.

[4] D. Bennion, R. Bietz, F. Thomas and M. Cimolai, "Reductions In the Productivity of Oil And Low Permeability Gas Reservoirs Due to Aqueous Phase Trapping," journal of canadian petroleum technology, vol. 33, no. 09, 1994.

[5] Y. D. Wang, M. J. Blunt, R. T. Armstrong and P. Mostaghimi, "Deep learning in pore scale imaging and modeling," Earth-Science Reviews, vol. 215, 2021.

[6] H. Al Khalifah, P. Glover and P. Lorinczi, "Permeability prediction and diagenesis in tight carbonates using machine learning techniques," Marine and Petroleum Geology, vol. 112, 2020.

[7] M. Abedini, M. Ziaii and J. Ghiasi-freez, "The application of Committee machine with particle swarm optimization to the assessment of permeability based on thin section image analysis," IJMGE, vol. 52, no. 2, pp. 177-185, 2018.

[8] F. Feng, P. Wang, Z. Wei, G. Jiang, D. Xu and J. Zhang, "A New Method for Predicting the Permeability of Sandstone in Deep Reservoirs," Geofluids, pp. 1-16, 2020.

[9] R. Rezaee and J. Ekundayo, "Permeability Prediction Using Machine Learning Methods for the CO2 Injectivity of the Precipice Sandstone in Surat Basin, Australia," Energies, vol. 6, 2022.

[10] M. A. Ahmadi and Z. Chen, "Comparison of machine learning methods for estimating permeability and porosity of oil reservoirs via petro-physical logs," Petroleum, vol. 5, no. 3, 2019.

[11] B. Singh, P. Sihag, S. M. Pandhiani and S. Gautam, "Estimation of permeability of soil using easy measured soil parameters: assessing the artificial intelligence-based models," journal of Hydraulic Engineering, 2019.

[12] N. Alqahtani, R. T. Armstrong and P. Mostaghimi, "Deep Learning Convolutional Neural Networks to Predict Porous Media Properties," in SPE Asia Pacific Oil and Gas Conference and Exhibition, 2018.

[13] Y. D. Wang, T. Chung, R. T. Armstrong and P. Mostaghimi, "ML-LBM: Predicting and Accelerating Steady State Flow Simulation in Porous Media with Convolutional Neural Networks," Transp Porous Med, pp. 49-75, 2021.

[14] S. George W, "Measurement of permeability I. Theory," Journal of Non-Crystalline Solids, vol. 113, no. 2-3, pp. 107-118, 1989.

[15] D. Sundaram, J. Tamás Svidró, A. Diószegi and J. Svidró, "Measurement of Darcian Permeability of foundry sand mixtures," international Journal of Cast Metals Research, vol. 34, no. 2, pp. 97-103, 2021.

[16] R. Baker and j. Doolittle, "Permeability measurement techniques for porous media: A review," Journal of hydrology, vol. 303, pp. 1-4, 2005.

[17] B. Azizzadeh mehmandost olya and R. Mohebian, "Q-FACTOR ESTIMATION FROM VERTICAL SEISMIC PROFILING (VSP) WITH DEEP LEARNING ALGORITHM, CUDNNLSTM," JOURNAL OF SEISMIC EXPLORATION, pp. 89-104, 2023.

[18] A. Chawla, P. Jacob, B. Lee and S. Fallon, "Bidirectional LSTM autoencoder for sequence-based anomaly detection in cyber security," International Journal of Simulation--Systems, Science & Technology, 2019.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, pp. 1735-1780, 1997.

[20] C. Kyunghyun, M. Bart van, C. Gulcehre, D. Bahdanau, B. Fethi and H. Schwenk, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," arXiv, 2014.

[21] T. Stérin, N. Farrugia and V. Gripon, "An intrinsic difference between vanilla rnns and gru models," COGNTIVE, p. 84, 2017.

[22] C. Sharan, W. Cliff, V. Philippe, C. Jonathan, T. John and C. Bryan , "cuDNN: Efficient Primitives for Deep Learning," ARXIV, 2014.

[23] j. Soete, L. Kleipool, H. Claes, S. Claes, H. Hamaekers, S. Kele and M. Özkul, "Acoustic properties in travertines and their relation to porosity and pore types," Marine and Petroleum Geology, vol. 59, pp. 320-335, 2015.

[24] G. Hamada and V. Joseph, "Developed correlations between sound wave velocity and porosity, permeability and mechanical properties of sandstone core samples," Petroleum Research, vol. 5, no. 4, pp. 326-338, 2020.

[25] C. Kyunghyun , B. v. Merrienboer, G. Caglar , B. Dzmitry , B. Fethi and S. Holger , "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," ARXIV, 2014.