IEICE TRANS. INF. & SYST., VOL.E101–D, NO.8 AUGUST 2018

LETTER

Transform Electric Power Curve into Dynamometer Diagram Image Using Deep Recurrent Neural Network

Junfeng SHI[†a)], **Wenming MA**[††b)], ***Nonmembers***, ***and*** **Peng SONG**[††], ***Member***

SUMMARY To learn the working situation of rod-pumped wells under ground, we always need to analyze dynamometer diagrams, which are generated by the load sensor and displacement sensor. Rod-pumped wells are usually located in the places with extreme weather, and these sensors are installed on some special oil equipments in the open air. As time goes by, sensors are prone to generating unstable and incorrect data. Unfortunately, load sensors are too expensive to frequently reinstall. Therefore, the resulting dynamometer diagrams sometimes cannot make an accurate diagnosis. Instead, as an absolutely necessary equipment of the rod-pumped well, the electric motor has much longer life and cannot be easily impacted by the weather. The electric power curve during a swabbing period can also reflect the working situation under ground, but is much harder to explain than the dynamometer diagram. This letter presented a novel deep learning architecture, which can transform the electric power curve into the dimensionless dynamometer diagram image. We conduct our experiments on a real-world dataset, and the results show that our method can get an impressive transformation accuracy.

key words: *deep learning, embedding, recurrent neural network, computer vision*

1. Introduction

The equipments of a rod-pumped well are always divided into two parts: some are on the ground, and the others are under the ground. We cannot know the working condition under ground directly, because we cannot frequently take the equipments out [1]. Due to the bad physical and chemical environments, we cannot install sensors to monitor the working condition under ground either. Usually, we make diagnosis indirectly by analyzing the dynamometer diagrams [2].

To generate the dynamometer diagrams, we need to install load sensor and displacement sensor on some special equipments of the rod-pumped well. These two kinds of sensors simultaneously collect data with the same sample rate. The load and displacement within the same swabbing period form a dynamometer diagrams. The X-aixs is the displacement and the Y-aixs is the load. The raw dynamometer diagram generated by sensors is known as *surface dynamometer diagram (SDD)*. However, this kind of dynamometer diagram cannot directly be used for diagnosis. We must transform the SDD into another kind of dynamometer diagram, which is known as *pump dynamometer diagram (PDD)*, through a series of mechanical calculations.

The shape of a PDD is closely related to at least one type of working conditions. Making diagnosis using a PDD is a typical process of image recognition. We don't really care about the real coordinate values of a PDD. Therefore, we actually use the dimensionless PDD (DPDD) to make diagnosis.

If we cannot collect correct data from sensors, we would get the incorrect SDDs, which would impact the shape of DPDD images. If we cannot get correct shape of DPDD images, we might get a totally different conclusion on the working condition under ground. Load sensors and displacement sensors are installed in the open air. The sensor accuracy can be impacted by the extreme weather, such as too cold or too hot. As time goes by, sensors are prone to generating unstable and incorrect data. One might expect that we could install new sensors every once in a while, but sensors, especially load sensors, are too expensive to frequently be replaced with new ones.

The electric motor, which is an essential equipment for a rod-pumped well, usually has a longer life than sensors, and is not easily impacted by the weather. Interestingly, the electric power curve (EPC) within a swabbing period can also reflect the working condition under the ground. In fact, the information in EPC (Fig. 1 left) is as much as DPDD (Fig. 1 right), but the shape of the former is too hard to explain, although the former can be directly used to make diagnosis. Therefore, as long as we transform a EPC into DPDD image, we can not only make diagnosis but also give an reasonable explanation. To the best of our knowledge, there is no literature that has studied this problem. Recently, Deep learning technologies have yield remarkable success on several domains [3]. Neural networks can almost approximate any non-linear function. The aim of this letter is using deep learning methods to do such transformation.

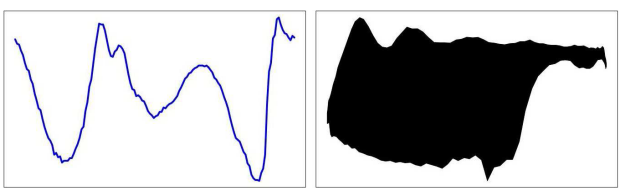

Fig. 1 Illustration of ECP (left) and DPDD (right)

Manuscript received February 6, 2018.
Manuscript revised April 12, 2018.
Manuscript publicized May 9, 2018.

[†]The author is with the Research Institute of Petroleum Exploration and Development, Petro China, Beijing 100083, P.R. China.
[††]The authors are with the School of Computer and Control Engineering, Yantai University, Yantai 264005, P.R. China.
a) E-mail: sjf824@petrochina.com.cn (Corresponding author)
b) E-mail: mwmytu@126.com (Corresponding author)

DOI: 10.1587/transinf.2018EDL8027

The main contributions of our work are as follows:

- We presented a novel deep recurrent neural network architecture to transform EPCs to a DPDD images, which is termed as DeepE2D, as well as some special methods of feature engineering.
- We conducted the experiment on a real-world datasets to demonstrate the performance of our model. The experimental results show that our model can get encouraging transformation accuracy.

The rest of this paper is organized as follows: In Sect. 2, we elaborate on our deep learning method of EPC-to-DPDD transformation. Section 3 describes our experiments and Sect. 4 concludes this paper.

2. Proposed Methodology

2.1 Architecture

Our deep recurrent neural network for EPC-to-DPDD transformation (DeepE2D) is shown in Fig. 2. There are totally four parts of this network: *Input Layers*, a *Recurrent Layer*, a *Dense Hidden Layer* and an *Output Layer*.

The role of *Input Layers* is transforming the raw data x into a embedding feature $\mathbf{x}_{em}$, which has more general information than the original feature. It is worth noting that each EPC has its own magnitude, and we must standardize x before embedding. The standardized feature is denoted as x_{sd}.

One might expect that x_{em} could be used as the input of the *Dense Layer* directly, just like many typical cases. However, for this case, there is no fixed meaning for each data point in the EPC. The motor power keeps running with a variable speed. When collecting the EPC data, we cannot know the exact locations of the data points. What we obtain

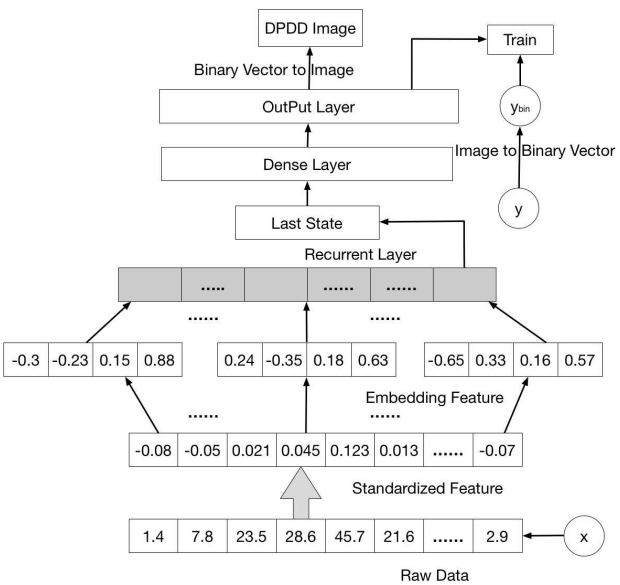

Fig. 2 Deep recurrent neural network for EPC-to-DPDD transformation (DeepE2D model)

is only a sequence of electric power curve data. Therefore, recurrent neural layers are suitable for handling such data. In this paper, we used gated recurrent unit (GRU) as the *Recurrent Layer* [4], [5].

Dense Layers are used to learn more complex abstract features. We used the rectified linear unit ReLU as the activation function of each dense layer. The major benefits of ReLU are sparsity and a reduced likelihood of vanishing gradient. The definition of ReLU is as follows [6], [7]:

$$\mathbf{h} = max(0, \mathbf{W}\mathbf{x} + \mathbf{b}) \tag{1}$$

What we care about is how to lean the shape of a DPDD image from a EPC. A DPDD image is always a closed curve. For convenience of learning, we filled in these closed curves with black color, and each DPDD became a binary image, the shape of which is $m \times n$. We then flattened them as vectors of dimension r, where r equals $m \times n$.

We treated our task as a *multi-label classification* problem [8], [9], so the activation function of the *Ouput Layer* is *sigmoid*, and the number of neurons in the *Ouput Layer* is also $m \times n$.

$$\hat{\mathbf{y}}_{bin} = \sigma(\mathbf{W}\mathbf{h}_{\mathbf{n}} + \mathbf{b}) \tag{2}$$

When training, the flatten vector of each DPDD image, denoted as $\mathbf{y}$, is need to be transformed into $\mathbf{y}_{bin}$, which has only $0s$ and $1s$. The $0s$ in $\mathbf{y}_{bin}$ stands for the pixels with white color, and the $1s$ for the pixels with black color in the original DPDD binary image. However, when predicting, we also need to transform $\hat{\mathbf{y}}_{bin}$ into $\hat{\mathbf{y}}$, the values of which stand for the pixels of the image. Finally, we must reshape the vector $\hat{\mathbf{y}}$ to a matrix, the size of which is $m \times n$. This matrix is just the DPPD image transformed from the EPC.

2.2 Feature Engineering

The original feature $\mathbf{x}$ from the raw data cannot be used to train the network directly. Different electric motors have different electric powers. We need to standardize $\mathbf{x}$ before feeding it to the learning model:

$$\mathbf{x}_{sd} = \frac{\mathbf{x} - \frac{1}{|\mathbf{x}|}\sum x_i}{var(\mathbf{x})} = \frac{\mathbf{x} - \bar{x}}{\sqrt{\frac{1}{|\mathbf{x}|}\sum (\mathbf{x} - \bar{x})^2}}, x_i \in \mathbf{x} \tag{3}$$

where $\bar{x}$ is the mean of vector $\mathbf{x}$, and $var(\mathbf{x})$ is the standard deviation of $\mathbf{x}$.

However, the length of $\mathbf{x}_{sd}$ is very small, which is 144 in our case, but the DPDD we need to learn has a large size. To obtain more complex and structural information from the original feature, we adopted *embedding techniques* [10]–[12] in this letter. We can discretize $\mathbf{x}_{sd}$ as follows:

$$\mathbf{x}_{dsd} = round(\frac{\mathbf{x}_{sd} - \min(\mathbf{x}_{sd})}{\max(\mathbf{x}_{sd})} \times C) \tag{4}$$

Now, each item in $\mathbf{x}_{dsd}$ is an integer, the value of which ranges from 0 to $C - 1$. We then transform each item, denoted as x^i_{dsd}, into a one-hot feature, the length of which is

C:

$$x^i_{dsd} \Rightarrow \mathbf{x}^i_{oh} \tag{5}$$

All the elements in $\mathbf{x}^i_{oh}$ are $0s$, except that the c_{th} element is 1.

For each element one-hot vector $\mathbf{x}^i_{oh}$, we want to lean a hash function that can transform it into a dense real-valued vector:

$$\mathbf{x}^i_{em} = \phi(\mathbf{x}^i_{oh}) = W\mathbf{x}^i_{oh} + b \tag{6}$$

The parameters are learned during procedure of training the whole network.

Similarly, the flattened vector $\mathbf{y}$ also need to be standardized. Because a DPDD image is binary, the value of each element in $\mathbf{y}$ is around 255 or 0. For a classification problem, we need to limit the value of labels in [0,1]. Therefore, we have:

$$\mathbf{y}_{sd} = \frac{255 - \mathbf{y}}{255} \tag{7}$$

When predicting, instead, we must transform the output into a binary image:

$$\mathcal{D} = reshape(255 - 255 * \hat{\mathbf{y}}, [m, n]) \tag{8}$$

2.3 Algorithms

We used a grammar like Keras framework to describe our algorithms here [13]. To speed up training time, we adopted ADAM as the optimizer [14].

Algorithm 1 describes the procedure how to build our deep neural model. x_{dim} is the length of discretized and standardized vector $\mathbf{x}_{dsd}$, while y_{dim} is the length of standardized and flattened vecotr y_{sd}. C is the number of integer values that the elements of $\mathbf{X}_{dsd}$ can be assigned to. L is the size of embedding feature $\mathbf{x}^i_{em}$. R is the dimensionality of the GRU output space.

Algorithm 2 shows the procedure of training our model. Here, we used the binary cross-entropy as the loss function. The number of data points in each electric power record is 144 in this letter, and the size of each DPDD image is 80×60, which is a appropriate size that can make the model fit better. To get statistical significant results, we shuffled the training dataset at each epoch.

3. Experiments

The key contribution of our work is on designing deep recurrent neural models for transforming EPCs to DPDD images. We conduct experiments with the aim of answering the following two questions:

Q1: Since we treat the transforming process as a multi-label classification problem, do our proposed method outperforms the existing classification algorithms?

Q2: Would different settings of the network have different impact on the performance?

Algorithm 1: Build Model

Function `buildModel`(x_{dim},y_{dim},C,L,R)**:**
  $\mathbf{X}_{dsd}$ = Input(shape=(x_{dim},), dtype='float32');
  $\mathbf{X}_{em}$ = Embedding(input_dim = C, output_dim = L, input_length=x_{dim})($\mathbf{X}_{dsd}$);
  GRU=GRU(units=R,return_sequences=False)($\mathbf{X}_{em}$);
  $\mathbf{H}_D$ = Dense(units=N_{h1}, activation='relu',name = 'H_D')($\mathbf{X}_{em}$);
  prediction = Dense(units=y_{dim}, activation='sigmoid')($\mathbf{H}_D$);
  model= = Model($\mathbf{X}_{sd}$, output=**prediction**);
  return model;
end

Algorithm 2: Train the Model

Function `trainModel`(**trainX**,**trainY**,t, b)**:**
  Transform each row of **trainX** into a discretized and standardized vector according to (7) and (8), and then we get matrix **trainX**$_{dsd}$;
  Transform each row of **trainY** into a standardized vector according to (11), and then we get matrix **trainY**$_{dsd}$;
  model = buildModel(144, 4800);
  model.compile(optimizer=Adam(lr=learning_rate), loss='binary_crossentropy',metrics=['accuracy']);
  model.fit(**trainX**$_{dsd}$ **trainY**$_{sd}$ batch_size=b, epochs=t, shuffle=True);
  SaveModel(**model**,$Path$);
end

We first present the experimental settings, followed by answering the above research questions one by one.

We collected about 2295 ECP-PDD samples of 10 rod-pumped wells in the same oil field. We then reprocessed these data and made a ECP-DPDD dataset. The length of a ECP vector is 144, and the size of a DPDD image is 80×60.

By default, the embedding size is 16, the output size of recurrent layer is 64, and we have only one dense layer here, which has 1024 neurons. We used the GNU recurrent layer by default in our model. We used emphbinary_crossentropy loss function and ADAM optimizer to train our model, and the learning rate is 0.01. The epoch is 80 and the batch size is 128. We used a 5-fold cross validation to evaluate the performance of models.

3.1 Performance Comparison (Q1)

To show the prediction accuracy of our model, we compared our proposed method with several other state-of-the-art approaches: Logistic Regression (LR), SVM (with linear kernel), and Multilayer Perceptron (MLP). The MLP has two hidden layers, the size of which are 1024 and 128 respectively.

We can see from Table 1 that our DeepE2D method got highest train accuracy and test accuracy, evaluated by a 5-fold cross validation. It is worth noting that MLP outperforms LR and SVM, which illustrates that deep models can fit more complex data. In our case, mapping 144 points to

Table 1 Performance comparison

Methods	CV Train Accuracy	CV Test Accuracy
LR	87.36%	87.17%
SVM	89.25%	89.08%
MLP	91.43%	90.71%
DeepE2D	93.29%	91.84%

Table 2 Impact of dense layer size

DL Size	CV Train Accuracy	CV Test Accuracy
32	84.80%	84.73%
64	84.76%	84.69%
128	92.47%	90.80%
256	91.50%	91.27%
512	92.72%	91.33%
1024	93.29%	91.84%
2048	94.29%	92.57%
4096	96.12%	93.17%

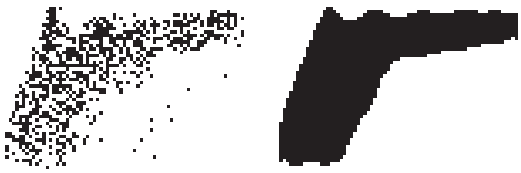

Fig. 3 DPDD images transformed by using SVM (left) and DeepE2D (right)

4800 points is so difficult that shallow architectures can not capture the relationship between the input and output well. However, the MLP doesn't treat the EPC as sequential data, which would cause incorrect mapping if similar DPDD images have very different EPC sequences in the dataset.

Although it seems that MLP and DeepE2D have only a little change in terms of accuracy metric, but this metric is borrowed from the multi-label classification tasks, which is actually not suitable for our task. However, until now we haven't found a suitable metric to reflect the transformation performance exactly. Fortunately, we found that a little improvement of such accuracy can significantly improve the ECP-to-DPDD quality. As shown in Fig. 3, the DPDD image transformed by using DeepE2D looks like more natural than the one transformed by using SVM. An important work of our future study would focus on the finding more suitable transformation accuracy metrics.

3.2 Impact of Model Settings (Q2)

We changed the DeepE2D model settings to see which factors might have significant impacts on the prediction accuracy.

Firstly, we changed the size of *Dense Layer* from 2^5 to 2^{12}, and the other settings were set as default. It seems that, from Table 2, larger *Dense Layer* could obtain higher accuracy. This is because the output of our task is too large, and a smaller dense hidden layer below the output layer would capture too general rules, which could transform all the EPC vectors to very similar DPDD images.

Table 3 Impact of recurret layer size

RL Size	CV Train Accuracy	CV Test Accuracy
8	89.68%	89.71%
16	90.98%	90.41%
32	92.78%	91.31%
64	93.29%	91.84%
128	96.03%	92.96%
256	89.29%	89.19%

Similarly, we changed the output size of last state of *Recurrent Layer*, and the other settings were set as default. Interestingly, larger size doesn't always get a better result. Setting the size as 128 got the highest accuracy, but there was a sudden decrease when we set the size as 256. It seems that large state size of *Recurrent Layer* would try to capture more local rules, but we have so much data to train the model.

4. Conclusion

In this letter, we presented a novel deep recurrent network to transform EPCs to DPDD images, which includes a *Embedding Layer*, a *Recurrent Layer*, a *Dense Layer* and a Sigmoid *Output Layer*. The *Embedding Layer* can learn more complex feature, which can capture the relationship between each element in original input. The *Recurrent Layer* treats the an EPC vector as sequential input, whereas the *Dense Layer* learns distributed representations of raw inputs. We adopted the GNU as the recurrent layer. Experimental results show that, by choosing appropriate settings, our DeepE2D outperforms the other existed classification algorithms.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 61602399).

References

[1] M. Xing, "Response analysis of longitudinal vibration of sucker rod string considering rod buckling," Advances in Engineering Software, vol.99, pp.49–58, 2016.

[2] K. Li, X.-W. Gao, H.-B. Zhou, and Y. Han, "Fault diagnosis for down-hole conditions of sucker rod pumping systems based on the FBH-SC method," Petroleum Science, vol.22, no.1, pp.135–147, 2015.

[3] M. Li, K.M. de Beurs, A. Stein, and W. Bijker, "Incorporating Open Source Data for Bayesian Classification of Urban Land Use From VHR Stereo Images," IEEE J. Selected Topics in Applied Earth Observations and Remote Sensing, vol.10, no.11, pp.4930–4943, 2017.

[4] X.Y. Zhang, F. Yin, Y.M. Zhang, C.L. Liu, and Y. Bengio, "Drawing and Recognizing Chinese Characters with Recurrent Neural Network," IEEE Trans. Pattern Anal. Mach. Intell., vol.40, no.4, pp.849–862, 2018.

[5] R. Zhao, D. Wang, R. Yan, K. Mao, F. Shen, and J. Wang, "Machine Health Monitoring Using Local Feature-Based Gated Recurrent Unit Networks," IEEE Trans. Ind. Electron., vol.65, no.2, pp.1539–1548, 2018.

[6] C. Zhang and P.C. Woodland, "DNN speaker adaptation using parameterised sigmoid and ReLU hidden activation functions," IEEE

International Conference on Acoustics, Speech Signal Process. (ICASSP), pp.5300–5304, Shanghai, 2016.

[7] H. Ide and T. Kurita, "Improvement of learning for CNN with ReLU activation by sparse regularization," International Joint Conference on Neural Networks (IJCNN), pp.2684–2691, Anchorage, AK, 2017.

[8] N. Inoue, E. Simo-Serra, T. Yamasaki, and H. Ishikawa, "Multi-label Fashion Image Classification with Minimal Human Supervision," IEEE International Conference on Computer Vision Workshops (ICCVW), Venice, Italy, pp.2261–2267, 2017.

[9] L. Jing, C. Shen, L. Yang, J. Yu, and M.K. Ng, "Multi-Label Classification by Semi-Supervised Singular Value Decomposition," IEEE Trans. Image Process., vol.26, no.10, pp.4612–4625, 2017.

[10] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," Proc. 22nd ACM international conference on Multimedia (MM '14), ACM, pp.675–678, New York, NY, USA, 2015.

[11] P. Wang, Y. Qian, F.K. Soong, L. He, and H. Zhao, "Word embedding for recurrent neural network based TTS synthesis," IEEE International Conference on Acoustics, Speech Signal Process. (ICASSP), pp.4879–4883, South Brisbane, QLD, 2015.

[12] K. Audhkhasi, A. Sethy, and B. Ramabhadran, "Semantic word embedding neural network language models for automatic speech recognition," IEEE International Conference on Acoustics, Speech Signal Process. (ICASSP), pp.5995–5999, Shanghai, 2016.

[13] F. Chollet, 2015. Keras: Deep learning library for theano and tensorflow, 2015. https://keras.io/ Accessed: 2018-01-26.

[14] D.P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, arXiv:1412.6980, Jan. 2017.