Recurrent Neural Networks for Oil Well Event Prediction

Lars Vidar Magnusson and J. Roland Olsson, *Østfold University College, Department of Computer Science and Communication, NO-1757, Halden, Norway*

Chau Thi Thuy Tran, *Institute for Energy Technology, NO-1757, Halden, Norway*

We have conducted a comparison between three types of recurrent neural networks and their ability to predict anomalies occurring in oil wells using a publicly available dataset. We have included two types of well-known state-of-the-art recurrent neural networks and a new type with neurons evolved specifically for the dataset using automatic programming. We show that the new type of recurrent neuron offers a massive improvement over the state of the art. The overall test accuracy of the new network type is 94.6%, which is an improvement by 18.3%, or 14.6 percentage points. We also show that a network with the new neuron performs better than any other solution proposed for the dataset.

INTRODUCTION

There are thousands of oil wells in maintenance in the world, and the number is expected to remain stable in the foreseeable future. The safe operation of these oil wells is critical to both our way of life and the environment.

Early prediction of anomalies and undesirable events would facilitate safe operation and could prevent potentially catastrophic events, but the complexity of the problem makes it difficult to do this manually. Petrobras, a Brazilian state-owned multinational company, has released a dataset suitable for training machine learning algorithms[1] for this purpose. The dataset is called *3W* since it contains data from three types of oil wells. Most of the data were captured from real oil wells, but they have been augmented with data from simulated oil wells and handcrafted data to reduce the imbalance between the different event types.

Attempts to utilize machine learning on the 3W dataset have so far been limited. One study[2] investigated three possible approaches and concluded that the random forest (RF) algorithm coupled with advanced preprocessing of the data and Bayesian optimization of the hyperparameters is the best solution. The researchers report an overall accuracy of 94%, which is the best published result for the dataset to our knowledge. Another study[3] compared the performance of several well-known classification algorithms, and the investigators concluded that conventional decision trees together with clever feature engineering offers the best performance. Both studies were limited to real-world and simulated instances, they exclude one of the event types due to a limited amount of data, and they perform comprehensive data processing and data augmentation to optimize the results. These are reasonable techniques to improve results, but they the dataset less representative of actual live data. A recent study with autoencoders for feature extraction[4] offers a better solution in terms of being able to handle live data, but it, too, reduces complexity by not considering transient states and removing handcrafted instances.

In this article, we investigate the performance of three deep recurrent neural networks (RNNs) on the 3W dataset. We did preliminary experiments with feed forward, convolutional, and RNNs and found the last of these to be best suited for the dataset. In general, time series is the natural application domain for RNNs due to their ability to remember and extract essential information from series. The most common RNN is long short-term memory (LSTM),[5] which we use as a baseline for our experiments. There are several derivatives of LSTM, such as gated recurrent units,[6] that we have not employed since they typically give similar results.

Digital Object Identifier 10.1109/MIS.2023.3252446
Date of publication 3 March 2023; date of current version 10 April 2023.

We chose, instead, to include a more recent and quite different architecture compared to LSTM, namely, Independently RNNs (IndRNNs)[7] that, for some datasets, beat LSTM by a wide margin. We have also included ADATE recurrent neural (ARN) networks, which are based on a new recurrent neuron that has been evolved specifically for the 3W dataset. We also included two types of ensemble classification algorithms similar to that of Marins et al.[2] for reference.

This is a comparative study, so we chose to make the problem as challenging as possible and the comparison fair rather than trying to optimize for the best results. We have used the entire 3W dataset (all well and event types) and no data processing or feature engineering, and we have used a time window of only 64 s. The problem we attempt to solve is, therefore, more challenging than that in previous works.[2–4] By using the entire dataset and no data processing, we also ensure that the trained models will be more likely to handle live data in an actual well, which likely would contain similar imperfections.

We present and compare the results of the LSTM, IndRNN, and ARN networks on the 3W dataset. We show that the ARN network outperforms the other recurrent networks by 18.3% or more in terms of overall accuracy and that similar improvements have been achieved in the other performance metrics we have employed. We also show that the ARN network performs better than the ensemble classification algorithms and that it performs as well as or better than the best known solution,[2] despite being trained and tested on a more challenging dataset and with less information available.

The remainder of the article is structured as follows. We start with a description of the 3W dataset and then present the RNN technology used. Then, we move on to describing how we compared the different models before presenting and discussing the results. The article is concluded with a brief discussion of what our results imply for the future of oil well event prediction and artificial neural networks.

OIL WELL EVENT DATA

The 3W dataset[1] is the first publicly available dataset for event prediction with multivariate time series data. It contains observations from three types of oil wells. The primary source of data has been real oil wells, but, to overcome challenges with sparse observations for certain types of events, they have also included observations from simulated wells and handcrafted data. There are 1984 instances in total: 1025 real wells, 939 simulated wells, and 20 handcrafted wells.

Each well has a variable number of observations, and there are eight sensor values in each observation. The values stem from sensors measuring the pressure, temperature, and flow rate at critical locations in the well such as the downhole safety valve (DHSV) and production choke (PCK), and their measurements are in pascals, degrees Celsius, and cubic meters, respectively. There is one observation per second for all of the well types.

The dataset contains eight different event types, which are listed in Table 1 together with the number of instances of each type. *Abrupt basic sediment and water (BSW) increase* signifies that there is a sudden increase in the amount of sediment and water in the flow. *Spurious DHSV closure* is when the DHSV valve closes without proper cause. *Severe slugging* is a critical event that occurs when there are extreme fluctuations in the flow due to large liquid slugs. *Flow instability* is a significant but tolerable instability in the flow that potentially could lead to severe slugging. *Rapid productivity loss* is a sudden decrease in the productivity of the oil well. *Quick PCK restriction* occurs when there is a restriction in the control valve over a short

TABLE 1. Event types in the 3W dataset and their corresponding number of instances.*

Description	Real	Simulated	Drawn	Total
Normal	597	0	0	597
Abrupt BSW increase	5	114	10	129
Spurious DHSV closure	22	16	0	38
Severe slugging	32	74	0	106
Flow instability	344	0	0	344
Rapid productivity loss	12	439	0	451
Quick PCK restriction	6	215	0	221
Scaling in the PCK	4	0	10	14
Hydrate in the production line	3	81	0	84
Total	1025	939	20	1984
Percentage of total	51%	47%	10%	100%

*Basic sediment and water (BSW) is the amount of water and sediment in the flow. The downhole safety valve (DHSV) is a valve mounted in the drill hole as a fail-safe mechanism. The production choke (PCK) is a device that controls the amount of fluids produced.

period of time. *Scaling in the PCK* is an accumulation of inorganic deposits in the control valve. *Hydrate in the production line* is when water and natural gas combine under high pressure into a substance that can cause blockage.

RNNs

RNNs are artificial neural networks in which we have one or more layers of recurrent neurons. These neurons differ from conventional artificial neurons in that they have cyclic connections to themselves, and they are invoked recurrently on a sequence of inputs.

LSTM Networks

LSTM networks were introduced as a solution to allow recurrent neurons to successfully store information over time.[5] In this study, LSTM serves as the starting point for the synthesis of the ARN neuron presented in the next section.

There are several versions of the LSTM neuron. In our experiments, we have chosen the *vanilla* LSTM neuron with peephole connections as described in Greff et al.[8] A full description is beyond the scope of this article. We have, however, included a version of an LSTM without peepholes in the diagram in Figure 1 to give an overview of the most important concepts.

The diagram presents a single neuron, but the formulas are expressed in vector and matrix notation. This is done to account for having multiple neurons in each layer and makes it possible to perform all of the

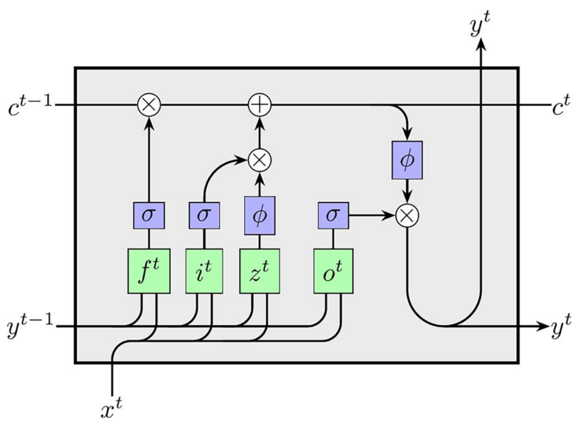

FIGURE 1. A circuit diagram of the long short-term memory (LSTM) neuron. Here, x^t, c^t, and y^t are the input, the internal state, and output of the neuron at time t, respectively. The blue boxes σ and ϕ are the sigmoid and hyperbolic tangent activation functions. The white circles are operators. The green equation boxes are defined in (1–3), and (4).

calculations for the entire layer at once. Each neuron has an internal state c^t. If a network has n neurons in a hidden layer, then the collected internal states for all of the neurons are represented by a vector $\mathbf{c}^t = [c_0^t, c_1^t, ..., c_n^t]$. The equation boxes in the diagram are defined as

$$\mathbf{f}^t = U_f\mathbf{y}^{t-1} + W_f\mathbf{x}^t + b_f \tag{1}$$
$$\mathbf{i}^t = U_i\mathbf{y}^{t-1} + W_i\mathbf{x}^t + b_i \tag{2}$$
$$\mathbf{z}^t = U_z\mathbf{y}^{t-1} + W_z\mathbf{x}^t + b_z \tag{3}$$
$$\mathbf{o}^t = U_o\mathbf{y}^{t-1} + W_o\mathbf{x}^t + b_o \tag{4}$$

where the dimensions of the weight matrices are defined by the dimensionality of the input and the number of LSTM neurons.

The diagram can be written in equation form as follows:

$$\mathbf{c}^t = \sigma(\mathbf{i}^t) * \phi(\mathbf{z}^t) + \mathbf{c}^{t-1} * \sigma(\mathbf{f}^t) \tag{5}$$
$$\mathbf{y}^t = \sigma(\mathbf{o}^t) * \phi(\mathbf{c}^t) \tag{6}$$

where σ is the sigmoid function, and ϕ is hyperbolic tangent. The operator $*$ denotes elementwise multiplication.

The internal state c^t of the neuron can store information. The input x^t as well as the output y^{t-1} and internal state c^{t-1} from the previous time step are passed through three separate *gates* that together control what gets stored and deleted in the internal state.

ARNs

ARNs are recurrent neurons that have been synthesized by the ADATE system,[9,10] which is a general system for the automatic inference of algorithms using evolutionary principles. ADATE uses a functional programming language to synthesize recursive programs. In this case, the programs represent recurrent neurons. ARNs are evolved by inferring and testing millions of possible neurons, as described by Olsson et al.[11]

The ARN presented in this article has been evolved specifically for the 3W dataset. A circuit diagram of the neuron can be seen in Figure 2. The ARN neuron introduces an extra bias:

$$b_0 = 0.18 \times 10^{-4}. \tag{7}$$

The calculations start with multiplying the input $\mathbf{x}^{(t)}$ with four weight matrices W_0, W_1, W_2, and W_3, the dimensions of which are defined by the input size and the number neurons in the layer. The new bias b_0 is also weighted and added to the weighted inputs:

$$\mathbf{f}^t = W_0\mathbf{x}^t + \mathbf{a}_0 b_0 \tag{8}$$
$$\mathbf{g}^t = W_1\mathbf{x}^t + \mathbf{a}_1 b_0 \tag{9}$$
$$\mathbf{h}^t = W_2\mathbf{x}^t + \mathbf{a}_2 b_0 \tag{10}$$
$$\mathbf{i}^t = W_3\mathbf{x}^t + \mathbf{a}_3 b_0. \tag{11}$$

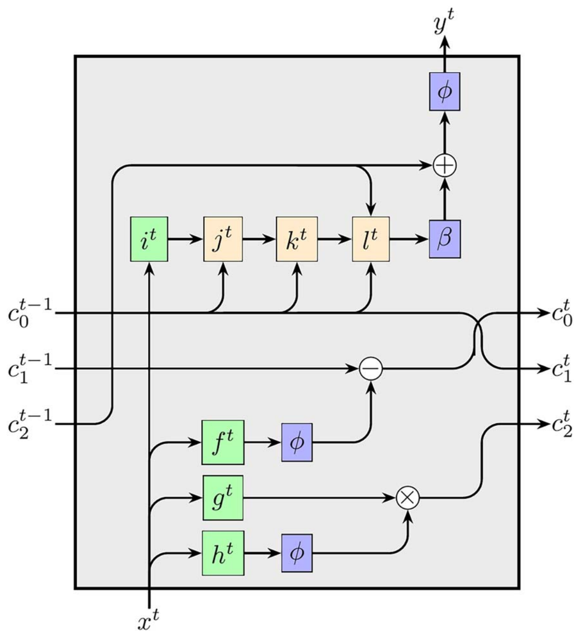

FIGURE 2. A circuit diagram of the ADATE recurrent neural (ARN) neuron. Here, x^t is the input; c_0^t, c_1^t, and c_2^t are internal states; and y^t is the output at time t. The equation boxes are defined in (8–11,17,18), and (19).

Here, $\mathbf{a}_{i\in\{0,1,2,3\}}$ are auxiliary weights. The neuron uses a set of activation functions defined as

$$\mathrm{relu}(x) = \alpha(x) = \max(0, x) \tag{12}$$
$$\mathrm{srelu}(x) = \beta(x) = \max(-1, \min(1, x)) \tag{13}$$
$$\phi(x) = \tanh(x) \tag{14}$$
$$\phi\phi(x) = \tanh(\tanh(x)) \tag{15}$$
$$\mathrm{rrelu}(x) = \delta(x) = \mathrm{relu}(\mathrm{srelu}(x)) \tag{16}$$

where relu is a widely used activation function for deep learning,[12] and srelu is a saturating relu function that has been used in automatic programming.[11] The rrelu function has been synthesized by ADATE for this neuron.

We have abstracted some of the transformations into separate equations to simplify the notation and highlight how they operate:

$$\mathbf{j}^t = -\delta(\mathbf{i}^t) * \mathbf{c}_0^{t-1} \tag{17}$$
$$\mathbf{k}^t = \mathbf{c}_0^{t-1} - \phi\phi\left[\frac{\mathbf{j}^t}{\mathbf{c}_0^{t-1}}\right] \tag{18}$$
$$\mathbf{l}^t = \mathbf{c}_2^{t-1} + \alpha(\mathbf{c}_0^{t-1}) - \mathbf{k}^t. \tag{19}$$

The final equations are defined as follows:

$$\mathbf{c}_0^t = \phi(\mathbf{f}^t) - \mathbf{c}_1^{t-1} \tag{20}$$
$$\mathbf{c}_1^t = \mathbf{c}_0^{t-1} \tag{21}$$
$$\mathbf{c}_2^t = \phi(\mathbf{h}^t) * \mathbf{g}^t \tag{22}$$
$$y^t = \phi[\mathbf{c}_2^{t-1} + \beta(\mathbf{l}^t)]. \tag{23}$$

The neuron has three internal states c_0^t, c_1^t, and c_2^t and produces an output y^t. It operates totally disconnected from the other neurons in the layer. This makes the neuron similar to IndRNNs.[7]

Another noteworthy characteristic is that the neuron does not consider any outputs. It primarily operates by maintaining its own internal states. It has been optimized to work ideally in a single layer, so each neuron would operate in complete isolation even with multiple layers of neurons.

The neuron also features a temporal skip connection for the internal state $\mathbf{c}_0^{(t-1)}$, which is forwarded to $\mathbf{c}_1^{(t)}$ without any alteration. This skip connection can be used to accumulate information more easily over time than would otherwise be possible.

HOW WE COMPARED PERFORMANCE

This section describes how we conducted our experiments to compare the performance of the recurrent networks and the classification algorithms.

Dataset Preparation

The 3W dataset contains eight event types,[1] as seen in Table 1. The dataset instances are organized in such a way that only one event can occur per oil well. Each instance is labeled with a single code representing either normal operation or that some type of event has occurred within the instance. All observations within an instance are labeled either as *normal*, *in transition*, or in a *stable* event state.

Not every instance has all types of observations. There are three possible situations: In the *normal* instances, all of the observations are labeled *normal*. In the *severe slugging* and *flow instability* instances, there is no transition, so all of the observations are labeled as *stable*. The remaining instances start out in normal operation before transitioning to an event.

We included all states in our experiments, so we ended up with 15 different ground-truth labels: two for the six event types that transition from normal operation to an event state, and one for normal operation and the two event types that have only stable observations. Table 2 contains a list of all event types and their labels.

The observations are stored as a time series with a frequency of 1 Hz. We tried sampling sequences with lengths in a power-of-two series $(2, 4, 8, \ldots, 256)$, and we settled on using a length of 64 since this gave the best result in our preliminary experiments. For *normal*, *severe slugging*, and *flow instability* instances, we

TABLE 2. Event types and their class labels.

Transitional	Stable	Event Description
	0	Normal
1	2	Abrupt BSW increase
3	4	Spurious DHSV closure
	5	Severe slugging
	6	Flow instability
7	8	Rapid productivity loss
9	10	Quick PCK restriction
11	12	Scaling in the PCK
13	14	Hydrate in the production line

extracted the observations directly from the first observations. For the remaining instance types, we found the location of the two event transitions with a linear search with a step size equal to the sample size.

The dataset was split into three parts using stratified sampling to ensure an equal distribution of the event types: 50% were used for training, 25% were used for validation, and 25% were used for the final testing. This partitioning was used for training and evaluating the performance of the RNNs.

The training of the tree ensemble algorithms was done on the concatenation of the training and validation datasets with threefold cross validation repeated 10 times. This gives the ensemble algorithms validation folds of the same size as the validation data for the neural networks (25% of the total), but it also gives the ensemble methods more training data.

The Neural Networks

We used the same simple neural network architecture in both our experiments with LSTM and ARN. The layers in the network are as follows:

- 8 input neurons.
- 128 LSTM/ARN neurons.
- 15 hyperbolic tangent activation neurons.
- 15 linear activation neurons.

The number of hidden recurrent neurons was chosen by doing preliminary tests with $2^n : n \in \{2, 3, 4, 5, 6, 7, 8\}$ neurons and 128 gave the best result. We also tested with more than one hidden layer, but without any increase in performance. The experiments with IndRNN were done with a slightly different architecture. The first and last layers are the same, but two middle layers are replaced by two layers with 256 recurrent neurons. This configuration was found in a manner similar to how we found the number of recurrent neurons for ARN/LSTM.

We tested both the ADAM[13] and NADAM[14] algorithms for training the networks. We found no significant difference between the two, so we decided to use the ADAM optimizer.

The Ensemble Classifiers

We included two ensemble classification techniques in our experiments to allow us to compare our results to techniques similar to the ones used by Marins et al.[2] and Turan and Jäschke.[3] We used the classifium GB (CGB) algorithm,[15] a version of gradient boosting, and RF.[16]

Hyperparameter Tuning

We found the data window size and the number of hidden recurrent neurons by doing preliminary tests as described earlier. We tuned all of the hyperparameters in the ADAM optimizer as suggested by Choi et al.[17] to allow for the best possible conditions for finding optimal weights when training the networks. The optimal values of the hyperparameters were found when using LSTM neurons, and then we used the same parameter values to evolve the ARNs.

The hyperparameters for the CGB algorithm are automatically tuned with a complex pipeline as described by Olsson and Acharya.[15] The hyperparameters for the RF algorithm and IndRNN were optimized using a grid search.

Target Functions and Other Metrics

We used categorical cross entropy (CCE) as the target loss function when training both the recurrent networks and the tree ensemble classification algorithms. CCE is defined as

$$\mathrm{CCE} = -\ln\left(\frac{e^{y_t}}{\sum_{j=1}^{C} e^{y_j}}\right) \qquad (24)$$

where t is the target class, y_i is the ith output of model y, and C is the number of target classes. We used the CCE and the accuracy metrics to evaluate the final performance on test data. Accuracy is defined as n_c/n, where n_c is the number of instances correctly classified, and n is the number of instances in total.

We also calculated the information retrieval characteristics *precision* (P), *recall* (R), and *F-measure* (F). These measurements can be defined using the true positive count p_t, the false positive count p_f, and the false negative count n_f:

$$P = \frac{p_t}{p_t + p_f} \quad R = \frac{p_t}{p_t + n_f} \quad F = \frac{2 * P * R}{P + R}.$$

These metrics are calculated per event type and averaged to find the final scores.

RESULTS AND DISCUSSION

The results of the LSTM and ARN networks on both validation data and test data can be seen in Table 3. The ARN network is far superior in terms of both accuracy and CCE. The CCE has been reduced to 49.0% of the LSTM CCE result on test data—a reduction of more than 50%. The accuracy of the ARN network is 18.3%, or 14.6 percentage points, better than the LSTM network on test data. We conducted a McNemar test[18] to ensure the statistical significance of this result, and the result was a p value of 1.14×10^{-21}—well below any reasonable limit.

These results show that the ARN network performs significantly better than LSTM and IndRNN on the 3W dataset. The accuracy result is also better than that reported by Marins et al.,[2] Turan and Jäschke,[3] and Gatta et al.[4] These results are not directly comparable, but the comparison should be beneficial for Marins et al.,[2] Turan and Jäschke,[3] and Gatta et al.[4] since they used a simplified dataset and significantly bigger window sizes to train their models. In the cases of Turan and Jäschke[3] and Gatta et al.,[4] the data were also processed extensively.

The results of CGB and RF are better than the results of LSTM and IndRNN, but both perform worse than the ARN network. In both cases, the amount is significant enough to suggest that the ordering is correct. This is strengthened by the fact that the CGB and RF models were trained using more data (75% of the total instead of 50%).

The average precision, recall, and F-measure for the two network types can be seen in Table 3. The ARN network has an average F-measure and recall above 0.84 and an average precision above 0.88, which are far better than the corresponding scores for LSTM and IndRNN. The ensemble methods CGB and RF perform better than LSTM and IndRNN but worse than ARN.

The per-event prediction performance can be seen in the confusion matrices in Figure 3. The precision and recall characteristics for both LSTM and IndRNN are quite good for many of the individual classes, but the scores are considerably worse for several others. It is apparent that the results for the ARN network are much more stable across all event types. The only clearly visible issue remaining is *flow instability*, which continues to be confused with normal operation for some instances, but there is a visible improvement over both LSTM and IndRNN.

A BRIEF LOOK AHEAD

In this article, we demonstrate that a new type of neuron that has been evolved specifically to predict rare and undesirable events in oil wells is able to achieve far better results than two state-of-the-art general-purpose recurrent neurons. The amount by which it has improved suggests that similar results can be achieved on other datasets with multivariate time series. This coincides with the earlier findings.[11] These findings are by no means conclusive in saying that similar results can be found for multivariate time series in general, but they suggest a very interesting possibility and certainly warrant more attention.

We also demonstrate that a simple deep neural network with specialized recurrent neurons is able to outperform the best proposed machine learning solutions for the dataset. These methods have required advanced feature processing and feature engineering, and they have been trained and tested with considerably larger time windows. The network proposed here is able to predict with a significantly higher degree of certainty and within, at most, 64 s what state the well is in. The

TABLE 3. The collected results of all of the networks and ensemble methods.*

	Validation Accuracy	Validation CCE	Test Accuracy	Test CCE	Precision	Recall	F-Measure
ARN	0.955	0.187	0.946	0.215	0.882	0.841	0.850
LSTM	0.845	0.385	0.800	0.437	0.784	0.728	0.733
IndRNN	0.844	0.340	0.795	0.488	0.776	0.710	0.718
CGB	0.912	—	0.915	—	—	—	—
RF	0.898	—	0.902	—	0.795	0.760	0.777

*The first columns contain the accuracy and CCE for the validation and test data, and the last three columns are the precision, recall, and F-measure on the test data. ARN: ADATE recurrent neural; CCE: categorical cross entropy; CGB: classifium GB; IndRNN: independently recurrent neural network; LSTM: long short-term memory; RF: random forest.

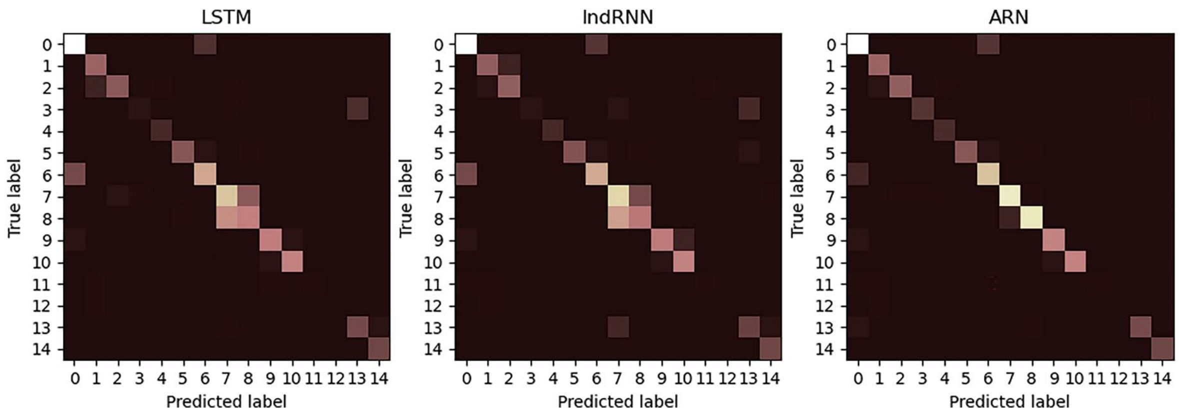

FIGURE 3. The confusion matrices for LSTM, IndRNN, and ARN on test data. Correct classes are in the rows, and predictions are in the columns. Brighter elements have higher values.

novel neural net that we present in this article could, therefore, likely be used as a subsystem in a real-time management system.

We are currently working on solutions for extending the work done with ARNs[11] to automatically evolve neural network architectures as well as the neurons, which could lead to a solution for synthesizing optimized neural networks for any problem.

ACKNOWLEDGMENTS

We would like to thank the anonymous referees who provided useful and detailed comments on an earlier version of the manuscript.

REFERENCES

1. R. E. V. Vargas et al., “A realistic and public dataset with rare undesirable real events in oil wells,” *J. Petroleum Sci. Eng.*, vol. 181, Oct. 2019, Art. no. 106223, doi: 10.1016/j.petrol.2019.106223.
2. M. A. Marins et al., “Fault detection and classification in oil wells and production/service lines using random forest,” *J. Petroleum Sci. Eng.*, vol. 197, Feb. 2021, Art. no. 107879, doi: 10.1016/j.petrol.2020.107879.
3. E. M. Turan and J. Jäschke, “Classification of undesirable events in oil well operation,” in *Proc. 23rd IEEE Int. Conf. Process Control (PC)*, 2021, pp. 157–162, doi: 10.1109/PC52310.2021.9447527.
4. F. Gatta, F. Giampaolo, D. Chiaro, and F. Piccialli, “Predictive maintenance for offshore oil wells by means of deep learning features extraction,” *Expert Syst.*, early access, Aug. 2022, doi: 10.1111/exsy.13128.
5. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
6. K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” 2014, *arXiv:1409.1259*.
7. S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, “Independently recurrent neural network (IndRNN): Building a longer and deeper RNN,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5457–5466, doi: 10.1109/CVPR.2018.00572.
8. K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A search space odyssey,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, doi: 10.1109/TNNLS.2016.2582924.
9. R. Olsson, “Inductive functional programming using incremental program transformation,” *Artif. Intell.*, vol. 74, no. 1, pp. 55–81, Mar. 1995, doi: 10.1016/0004-3702(94)00042-Y.
10. R. Olsson, “Population management for automatic design of algorithms through evolution,” in *Proc. IEEE Int. Conf. Evol. Comput.*, 1998, pp. 592–597, doi: 10.1109/ICEC.1998.700095.
11. R. Olsson, C. Tran, and L. Magnusson, “Automatic synthesis of neurons for recurrent neural nets,” 2022, *arXiv:2207.03577*.
12. P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for activation functions,” 2017, *arXiv:1710.05941*.
13. D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, *arXiv:1412.6980*.
14. T. Dozat, “Incorporating Nesterov momentum into ADAM,” in *Proc. Int. Conf. Learn. Representations*, 2016, pp. 1–6.
15. R. Olsson and S. Acharya, “Using automatic programming to improve gradient boosting for

classification," in *Proc. Int. Conf. Artif. Intell. Soft Comput.*, Springer, 2022, pp. 242–253.

16. T. K. Ho, "Random decision forests," in *Proc. 3rd IEEE Int. Conf. Document Anal. Recognit.*, 1995, vol. 1, pp. 278–282, doi: 10.1109/ICDAR.1995.598994.
17. D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On empirical comparisons of optimizers for deep learning," 2019, *arXiv:1910.05446.*
18. Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, Jun. 1947, doi: 10.1007/BF02295996.

LARS VIDAR MAGNUSSON is an associate professor at Østfold University College, NO-1757, Halden, Norway. His research interests include image analysis, natural language processing, and time series analysis. Magnusson received his Ph.D. degree in automatic programming and image analysis from the University of Oslo. Contact him at lars.v.magnusson@hiof.no.

J. ROLAND OLSSON is an associate professor at Østfold University College, NO-1757, Halden, Norway. His research interests include recurrent neural nets, automatic programming, and neural program synthesis. Olsson received his Dr. Scient. degree in automatic programming from the University of Oslo. Contact him at roland.olsson@hiof.no.

CHAU THI THUY TRAN is a researcher at the Institute of Energy Technology, NO-1757, Halden, Norway. Her research interests include machine learning, optimization, and computer vision. Tran received her master's degree in machine learning from Østfold University College. Contact her at chau.tran@ife.no.