PROCEEDINGS OF
INTERNATIONAL CONFERENCE ON ADVANCED TECHNOLOGIES
https://proceedings.icatsconf.org/
11th International Conference on Advanced Technologies (ICAT'23), Istanbul-Turkiye, August 17-19, 2023.

Development of Resolver Circuit with Long Short Term Memory and Reinforcement Learning Algorithms

Yusuf ÇAĞLAYAN[1]

[1]*ASELSAN A.S. Ankara, Turkey*
ycaglayan@aselsan.com.tr

***Abstract*— In our age, the usage areas of artificial intelligence have increased considerably. These areas were particularly concerned with the correct predictability of future data using available data. It has become necessary to work on various machine learning algorithms to be used in the calculations of the resolver circuit, which is a feedback element used for tracking the position and position information of the electric motor unit used in various vehicles. The use of machine learning algorithms in the design and implementation of the resolver circuit, which is one of the most important elements of electric motor designs, will shed light on future studies. In this study, it is focused on the use of machine learning algorithms in the calculation of the resolver circuit, position and position information and the performance differences between each other. In this study, LSTM (Long Short Term Memory) and Reinforcement Learning (RL) algorithms were compared. While comparing these algorithms, the types of LSTM and RL algorithms were also studied and compared. As a result of the results obtained, it was aimed that the motor designs would be less costly, and the results obtained in terms of more reliable motor position and position information to be used were promising. In addition, with this study, a basis was created for working on machine learning algorithms in the calculation of different parameters. With this study, a great way has been achieved in integrating algorithms used in electric vehicles, which are quite obsolete today, into AI-based algorithms.**

***Keywords*— Resolver Circuit, Long-Short-Term Memory (LSTM) Networks, Reinforcement Learning Algorithms**

I. INTRODUCTION

This article presents a research on combining solver circuits with long-short-term memory (LSTM) and reinforcement learning algorithms. Resolver circuits are one of the important components used to determine the angular position of a rotating shaft. Long short-term memory (LSTM), on the other hand, is a type of neural network designed for sequential data, while reinforcement learning is a machine learning approach that allows an agent to learn the environment through rewards and punishments from its environment. This article examines how LSTM and reinforcement learning algorithms can be integrated to improve the performance of solver circuits and how this combined method can achieve better results. The development of solver circuits plays an important role in many fields such as automation, robotics and industrial applications. Therefore, the findings of the article can make valuable contributions to the relevant industries and academia.

This research demonstrates the potential of solver circuits to harness the power of artificial intelligence and machine learning techniques without being limited to traditional methods. The results of the article can inspire researchers, engineers and those interested in this field and pave the way for new discoveries.

It is hoped that the methods presented in the article will be effective in improving the performance of solver circuits. The results are successful, there may be wider use of such techniques in industrial applications and other related fields.

The organizational structure of the remainder of this article is as follows. In the following section we conclude the summary of the relevant studies, our approach and working in Chapter 3, the comparison of results industrially and the demonstration results are in Chapter 4, and finally the article in Chapter 5.

II. RELATED WORK

Short-term memory is one of the mechanisms of iterative neural networks, which is one of the deep learning algorithms. Typically, this algorithm validates datasets and snapshot data. The LSTM algorithm has been used in various fields as an advantage of the memory and data analysis engine. The LSTM algorithm ensures that all possible negative scenarios can be predicted and precautions can be taken against this. For example, the LSTM algorithm is used to predict sewer overflows in problem areas [1] of municipalities. In addition, the LSTM algorithm is used to protect automatic systems from external interference, which are necessary for electrical networks [2]. With all these results, the LSTM algorithm is used to estimate wind energy production [3], estimate solar energy production [4], estimate traffic and estimate the trajectory of medical services [5] [6], answer rating [7]. Given

the good results in other areas, we hope that using the LSTM algorithm to generate test cases is an effective strategy.

Machine learning can be used to detect and prevent software vulnerabilities and even create negative test cases. To this end, a number of studies are being carried out. For example, deep learning algorithms are used to detect and prevent [8] software security vulnerabilities.

Nuclear reactors are quite serious constructions. For these structures, all safety measures must be taken. At this stage, possible failure scenarios are very important. Possible accident scenarios created using deep learning methods can be generated automatically [9].

III. METHODOLOGY

Recursive Neural Network (RNN), Long Short-Term Memory (LSTM) and Reinforcement Learning Algorithm are important constructs in deep learning. While RNN handles sequential data processing and time dependency, LSTM has the ability to learn long-term dependencies effectively. The reinforcement learning algorithm aims to learn the best actions by interacting with the environment. It should be considered that these three main players together contribute to the effective performance of deep learning in various fields.

A. Recursive Neural Networks (RNN)

Recursive Neural Networks (RNN) is a deep learning model that can recognize time or sequential connections between input and output. These connections are very important for cases where the data is in a sequential structure. For example, cases where each word is related to the previous word in texts and the outputs change depending on the input order, such as language translation, show the power of RNN. RNN has a cyclical nature, meaning they feed their output back to them via loop so they can remember previous states. However, RNNs have difficulties in learning long- term dependencies effectively. An issue called gradient loss occurs when gradient values are too small or too large during backpropagation and can negatively impact their ability to handle long-term dependencies.

B. Long Short Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a deep learning model developed in response to the learning difficulties of long-term dependencies of RNNs. LSTM is particularly effective for handling texts, time series, and other data series with long-term dependencies. LSTM cells are equipped with special structures that can remember previous states more effectively. LSTM consists of four main components called forget, input, output and memory gate. These structures regulate the storage, updating and use of the information inside the cell as output. By reducing gradient loss, they make long-term addictions more learnable. Therefore, in the field of natural language processing such as language translation, language modeling and language analysis, and in time series estimation, LSTM can achieve more effective results than RNNs. However, due to their more complex structure, they may require more computational power and the training process may take longer because they contain more parameters. Also, the risk of over-learning may increase due to the model complexity of LSTM.

C. Reinforcement Learning Algorithm

Reinforcement learning is a learning approach in which an agent tries to learn best actions through their experiences by interacting with their environment. One of the most known algorithms in this field is Q-learning. Reinforcement learning can achieve successful results in complex and uncertain environments. It can be used especially in application areas such as gaming, robotics, finance and traffic management. The basic structures of the algorithm are state-space, action- space, reward function and policy function. The state-space is the set of states that describe the agent's environment; action-space is the set of actions that the agent can perform. The reward function provides the mechanism by which the agent is rewarded for successful actions and punished for unsuccessful actions. The policy function is the mechanism by which the agent chooses what action to take in a given situation. The advantages of the reinforcement learning algorithm include the ability to quickly adapt to environmental conditions and optimize certain tasks. The algorithm can collect data on its own to solve certain tasks and learn from its experience the actions that will best perform these tasks. However, it can be difficult to design a reward function for a reinforcement learning algorithm, and incorrect rewards can adversely affect the performance of the algorithm. The training process can take a long time and require high computational power. Also, the agent's learning through experience can sometimes be long and complex due to the uncertainty of the environment.

IV. INDUSTRIAL CASE STUDY

Electric motors are critical components widely used in many industrial applications and vehicle systems. Accurate and precise operation of these motors is critical to performance and efficiency. Resolver circuits are used to precisely detect and control the angle of rotation in electric motors. Traditional resolver circuits have limitations in handling long-term dependencies and complex environmental conditions in certain situations. Therefore, taking a more advanced and adaptive approach is an important step towards improving the performance of electric motors and making them run more reliably.

Long Short-Term Memory (LSTM) is a powerful artificial neural network model that offers superior ability in processing sequential data and recalling previous states. The use of LSTM in the development of resolver circuits provides the opportunity to learn longer-term dependencies more effectively. This allows for more precise and stable monitoring of the angle of rotation of the engine under complex operating conditions. The LSTM-based resolver circuit can adapt faster to the fast changing workloads and external factors of the motor, thus increasing the efficiency and response time of the motor.

The reinforcement learning algorithm can be used to provide adaptive control in the resolver circuit. The algorithm can generate optimal control strategies based on the real-time operating conditions of the engine. This allows the engine to

react quickly to unpredictable environmental changes and optimize its performance. At the same time, reinforcement learning can also reduce energy consumption by learning the correct and optimized operating conditions to increase the energy efficiency of the motor. In this way, the resolver circuit developed with LSTM and reinforcement learning algorithm has the potential to increase the efficiency of electric motors and offer a more adaptive control mechanism.

A. Experimental Setup

This study was conducted using LENOVO 20YTS0GS00 computer with 11th Gen Intel(R) Core(TM) i7-11850H @ 2.50GHz 2.50 GHz operating system. While using the programs, Tensorflow and stable-baselines3 libraries were used. In this project, all existing angle values from 0 to 360 degrees will be used to calculate the sine and cosine values for the input values. And when testing, these fixed values will be taken as a basis.

B. Application Management

The development phase was primarily tested with the LSTM algorithm. the code first generated sine and cosine values for angles between 0 and 360 degrees using NumPy. Then, the input data (angles) and target data (sine and cosine values) were prepared to train the LSTM model. Normalized to scale the input data from 0 to 1. Then, their data was split into training and test sets.

Next, we create an LSTM model using TensorFlow's Keras API. The model consists of a 64-unit LSTM layer and a 2-output Dense layer to estimate the sine and cosine values. The mean square error (MSE) was used as the loss function and the Adam optimizer for training. The model is then trained on the training data and evaluated on the test data. Finally, the trained LSTM model is used to estimate the sine and cosine values for a new angle (in this case, 45 degrees) and print the predicted results.

A summary of the LSTM model used is shown in Table II.

TABLE I
THE CALCULATED MATHEMATICAL VALUE OF THE SINE AND COSINE VALUES AT THE SELECTED ANGLE DEGREE (45 DEGREES)

sin(45)	0.85090352453
cos(45)	0.52532198881

TABLE II
SUMMARY OF THE USED FIRST LSTM MODEL

Model: sequential		
Layer (type)	Output Shape	Param #
lstm 63 (LSTM)	(None, 64)	16896
dense 54 (Dense)	(None, 2)	130
Total params: 17,026		
Trainable params: 17,026		
Non-trainable params: 0		
Predicted Sine: 0.6343855857849121		
Predicted Cosine: 0.68389892578125		

When the values in Table 1 and the values estimated by the LSTM algorithm for the sine and cosine values in Table 2 are compared, the need for further development of the LSTM model has emerged. If this development is gathered under 10 headings,

1) Increasing Model Complexity: Consider increasing the number of LSTM units or adding more LSTM layers to the model. A more complex model may have a higher capacity to catch complex patterns in the data, potentially leading to better predictions and less loss.

2) Adjusting the Learning Rate: Experiment with different learning rates for the optimizer. Too high a learning rate may cause the model to exceed the optimal solution, while too low a learning rate may result in slow convergence. Finding an appropriate learning rate can significantly affect the training performance of the model.

3) Batch Size: Try different batch sizes during training. Larger batch sizes can help achieve stable updates of model weights, potentially resulting in faster convergence and lower loss. However, very large heap sizes may require more memory during training.

4) Array Length: Set the string length used for training. In current code, each angle is treated as a separate array with array length 1. You can experiment with longer sequences to provide more context to the LSTM and potentially improve predictions.

5) Editing Techniques: Apply regularization techniques such as L1 or L2 editing to the LSTM layer or Dense layer. Regularization can prevent overfitting and improve the generalization performance of the model.

6) Using a Different Activation Function: Try using different activation functions for the LSTM layer, such as hyperbolic tangent (tanh) or sigmoid. Different activation functions can affect the model's ability to learn patterns in the data and contribute to loss reduction.

7) Data Augmentation: If you have a limited amount of data, consider applying data augmentation techniques to create additional training examples. For example, you can add random noise to the angle data to create variations and increase the variety of training samples.

8) Gradient Clipping: Apply gradient clipping to limit the size of gradients while training. This can help prevent bursting gradients, especially in deep LSTM networks.

9) Hyperparameter Tuning: Experiment with different hyperparameter values such as group size, number of periods, and LSTM layer units. A systematic hyperparameter search can help find the optimal configuration to reduce loss.

10) Stop Early: Practice stopping early during exercise. Monitor the loss on the validation set and stop training when the loss starts to increase or reaches a plateau. This can prevent overfitting and help maintain the best model.

The new LSTM model can be changed as follows:

- To increase the number of LSTM units to 128.
- To increase the batch size to 32.
- To set the sequence length to 5.
- To add L2 regularization to the LSTM layer with a regularization strength of 0.01.
- To shuffle the training data before each epoch to introduce data augmentation.

- To add gradient clipping to the optimizer with a clip value of 1.0.
- To use the 'tanh' activation function for the LSTM layer.
- To set the learning rate of the optimizer to 0.001.
- to implement early stopping with a patience of 10 epochs.
- To use the Adam optimizer with the default learning rate.

TABLE III
SUMMARY OF THE USED SECOND LSTM MODEL

Model: sequential		
Layer (type)	Output Shape	Param #
lstm 13 (LSTM)	(None, 128)	66560
dense 11 (Dense)	(None, 2)	258
Total params: 66,818		
Trainable params: 66,818		
Non-trainable params: 0		
Predicted Sine: 0.8304228782653809		
Predicted Cosine: 0.52862938165664673		

The new LSTM model formed when 10 solutions are applied to the 1st LSTM model is shown in Table 3. When the sine and cosine values in Table 3 and Table 1 are examined, it is seen that the new implementations have reduced the loss value and are close to the real value.

Now, if a model is to be built with Reinforcement Learning Algorithm, the feature of the model is as follows:

The code defines a special environment called "SineCosineEnv" to simulate the predicted sine and cosine values for certain angles. It uses a 64 element LSTM model and a dense layer to predict values. The reinforcement learning algorithm used is Deep Q-Networks (DQN) by Stable-Baselines3. The DQN agent is trained for 100,000 time steps using "MlpPolicy" with MLP architecture. The agent takes the current normalized angle as input and learns to estimate the corresponding sine and cosine values in an effort to minimize the standard error between the predictions and the actual values.

Enhanced learning features:

- Environment: A special environment presents the problem of estimating the sine and cosine values for angles.
- DQN: The Deep Q-Network algorithm is used to train an agent to make optimal predictions based on actions (angles) and rewards (prediction accuracy).
- Principle: "MlpPolicy" is chosen, which uses a multi-layer perceptron (MLP) architecture to select actions in discrete areas.
- Discovery: DQN uses epsilon-greedy discovery where an agent looks for a certain probability (epsilon) and otherwise uses its current knowledge.
- Target Network: DQN uses a target network to stabilize learning, using separate networks for action selection and value estimation.

Thus, the code trains the DQN agent to evaluate sine and cosine values using an LSTM model in the user environment.

The trial and error tool learn to make accurate predictions for different angles and combines reinforcement learning methods such as DQN algorithm, experience repetition, and target networks to provide effective learning.

TABLE IV
SUMMARY OF THE USED FIRST REINFORCEMENT LEARNING ALGORITHM

Model: sequential		
Layer (type)	Output Shape	Param #
lstm(LSTM)	(None, 64)	18896
dense(Dense)	(None, 1)	65
Total params: 16,961		
Trainable params: 16,961		
Non-trainable params: 0		
Predicted Sine: 0.016603474277024376		
Predicted Cosine: 0.9998621528200435		

Comparing the results obtained in Table 1 with Table 4, the estimated values were far from the true values. For this, various solution methods should be tried and a result closer to the real results should be obtained. To further improve these results, the following methods can be tried:

- Normalizing Target Values: Normalizing target sine and cosine values to the range [0, 1] to match the output of the model.
- Normalizing Observations: Shrinking the observations (input data) to a smaller range to improve convergence and stability during training.
- Increasing Training Time Steps: Increasing the number of training timesteps to enable the agent to learn better impressions and approach a more appropriate policy
- Adjusting the Learning Rate: Experiment with different learning rates should be done for the optimizer.
- Exploring Different Architectures: Try different LSTM model architectures, such as increasing the number of LSTM units, adding more layers, or using different enable functions, to see if the model can capture more complex patterns.

In the next step, a new solution was applied as follows and the results are shown in Table 5.

- Target sine and cosine values were normalized to the range [0, 1] by adding 1 and dividing by 2,
- Observations (input data) were normalized to the interval [0, 1] by dividing by ,360.0,
- The training time steps are increased to 5000 to allow the agent to learn better representations and approach a more appropriate policy.
- The learning rate was set to 0.001 to prevent overshoot and improve convergence.

The results obtained with the changes made are shown in Table 5. When the values in Table 5 and Table 1 are compared, the implementations are quite close to the real value.

C. *Results and Discussion*

When all the results are examined, it has been determined that deep learning and machine learning algorithms are very advantageous in the predictive development of the sine and cosine values used in the calculations of the resolver circuit.

TABLE V
SUMMARY OF THE USED SECOND REINFORCEMENT LEARNING ALGORITHM

Model: sequential		
Layer (type)	Output Shape	Param #
lstm(LSTM)	(None, 64)	18896
dense(Dense)	(None, 1)	65
Total params: 16,961		
Trainable params: 16,961		
Non-trainable params: 0		
Predicted Sine: 0.8367458		
Predicted Cosine: 0.519999		

These determinations have emerged by developing LSTM and Reinforcement algorithms. These algorithms can be further developed according to the related problem and the inputs used, and their architecture can become a very different role. The important point here is the nature and number of the problem and the data used. In the studies, a system specific to our problem and specific to the data type we use has been developed and development processes have been carried out with LSTM and Reinforcement learning algorithm architectures. When all the results are examined, the absolute result is that the algorithm can be improved.

V. CONCLUSIONS

In conclusion, the application of LSTM and Reinforcement Learning algorithms to the resolver circuit used in electric motors has shown promising results in accurately predicting sine and cosine values. By optimizing the LSTM model with more units (128), larger batch sizes (32), and longer sequence length (5), we were able to improve model accuracy and convergence. The addition of L2 regularization (0.01) and gradient clipping (1.0) further stabilized the learning process, preventing overfitting and improving generalization. Additionally, using the "tanh" trigger function and using an early shutdown with a 10-epoch delay improved the model's performance.

For the reinforcement learning algorithm, normalizing the target sine and cosine values along with the input data allowed the agent to make more accurate predictions. With an increase in the number of learning time steps (5000) and a learning rate of 0.001, the agent learned to better represent the environment and moved closer to a more appropriate policy. Taken together, these results demonstrate the effectiveness of LSTM and reinforcement learning approaches to accurately predict motor parameters. As for future research, researchers could explore the integration of more complex architectures and advanced optimization techniques to further enhance the predictive capabilities of these algorithms. Furthermore, the inclusion of real engine data and the study of various reward functions can provide valuable insights into optimizing engine performance in practical applications. The combination of these approaches promises to improve motor control and efficiency in a variety of industrial and automotive applications.

ACKNOWLEDGMENT

We would like to thank the Aselsan A.S. test groups and anonymous comments for supporting our study. I am also grateful to Cem Çiğdemoğlu and Mehmet Kubilay.

REFERENCES

[1] Gudaparthi, H., Johnson, R., Challa, H., and Niu, N., "Deep learning for smart sewer systems: Assessing nonfunctional requirements," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society*, Jun, 2020, paper, p. 35-38.

[2] Musleh, A. S., Chen, G., Dong, Z. Y., Wang, C., and Chen, S. "Attack Detection in Automatic Generation Control Systems using LSTM-based Stacked Autoencoders," *IEEE Transactions on Industrial Informatics.*, 2022.

[3] Tao, Y., Chen, H., and Qiu, C., "Wind power prediction and pattern feature based on deep learning method," in *2014 IEEE PES Asia- Pacific Power and Energy Engineering Conference (APPEEC)*, 2014, paper, p. 1-4.

[4] Gensler, A., Henze, J., Sick, B., and Raabe, N., "Deep Learning for solar power forecasting—An approach using AutoEncoder and LSTM Neural Networks," in *2016 IEEE international conference on systems, man, and cybernetics (SMC)*, 2016, paper, p. 002858-002865.

[5] Zhang, W., Yu, Y., Qi, Y., Shu, F., and Wang, Y., "Short-term traffic flow prediction based on spatio-temporal analysis and CNN deep learning," *Transportmetrica A: Transport Science*, vol. 15(2), pp. 1688- 1711 , 2019.

[6] Pham, T., Tran, T., Phung, D., and Venkatesh, S., "Predicting health-care trajectories from medical records: A deep learning approach," *Journal of biomedical informatics*, vol. 69, pp. 218-229, 2017.

[7] Li, Z., Huang, J., Zhou, Z., Zhang, H., Chang, S., and Huang, Z., "LSTM-based deep learning models for answer ranking," in *2016 IEEE First International Conference on Data Science in Cyberspace (DSC)*, 2016, paper, p. 90-97.

[8] Singh, S. K., and Chaturvedi, A., "Applying deep learning for dis-covery and analysis of software vulnerabilities: A brief survey," *Soft Computing: Theories and Applications*, pp. 649-658, 2020.

[9] Kim, H., Cho, J., and Park, J., "Application of a deep learning technique to the development of a fast accident scenario identifier," *IEEE Access*, vol. 8, pp. 177363-177373, 2020.