A review of recurrent neural network architecture for sequence learning: Comparison between LSTM and GRU

Shiva Nosouhian
Electrical Engineering
Islamic Azad University Najafabad Branch

Fereshteh Nosouhian
Mechanical Engineering
Isfahan University of Technology

Abbas Kazemi Khoshouei
Agricultural Engineering
University of Shahrekord

Abstract

Deep neural networks (DNNs) have made a huge impact in the field of machine learning by providing unbeatable human-like performance to solve real-world problems such as image processing and natural language processing (NLP). Convolutional neural network (CNN) and recurrent neural network (RNN) are two typical architectures that are widely used to solve such problems. Time sequence-dependent problems are generally very challenging, and RNN architectures have made an enormous improvement in a wide range of machine learning problems with sequential input involved. In this paper, different types of RNN architectures are compared. Special focus is put on two well-known gated-RNN's Long Term Short Memory (LSTM) and Gated Recurrent Unit (GRU). We evaluated these models on the task of force estimation system in pouring. In this study, four different models including multi-layers LSTM, multi-layers GRU, single-layer LSTM and single-layer GRU) were created and trained. The result suggests that multi-layer GRU outperformed other three models.

Keywords—Recurrent neural network, Long-term short memory, Gated recurrent unit

1 Introduction

Nowadays, the recurrent neural network (RNN) has gained a lot of attention due to its outstanding performance in solving real-world machine learning problems, especially when it comes to dealing with sequential data and input-output data having different lengths [1–12]. Alex Graves in "Supervised Sequence Labelling with Recurrent Neural Networks" shows that RNNs are very powerful sequential learners [13]. One of the most popular areas where RNNs are usually used is in modeling natural language processing such as machine translation [14]. Other areas include handwriting recognition/generation [15], speech recognition [16], and human activities modeling [17].

In the past two decades, robots have taken charge in many areas such as medical centers [18], military [19] and industry [20]. Scientists have been trying to design and build robots that are capable of smooth and natural movements to do the tasks that are normally done by human beings. One of the most activities that can be helpful in lots of situations is pouring. We use pouring in our daily activities in an unconscious way.

To properly calculate the amount of transferring during the pouring process, humans use two factors: vision feedback and force feedback. Y. Huang et al. introduce an approach RNN model base on the LSTM to simulate the pouring activity [21].

In this work, the main focus is to compare different RNN architectures for this problem. We used a number of trainable parameters as a similarity factor to generate four different models. Models include 3-layer LSTM, 3-layer GRU, 1-layer LSTM and 1-layer GRU. The efficiency of LSTM and GRU are compared under the same conditions. Based on the experience in a very similar environment with the same features such as a number of epochs, loss function, optimizer and batch size, GRU produces better results than LSTM.

This paper is organized as follows. Section II includes a brief background on the recurrent neural network and elaborates on the difference between LSTM and GRU. Section III describes the dataset and dataset preprocessing. In section IV, architectures and the training process are discussed. Section V shows the evaluated results and concludes the paper.

2 Background

In this section, we describe two recurrent units LSTM and GRU.

2.1 Long Term-Short Memory (LSTM)

LSTM was introduced in 1997 by Hochreiter & Schmidhuber to mitigate the vanishing gradient problem which previously recurrent networks would encounter [21] [22]. Instead of vanishing or exploding backpropagated errors, in LSTM, they will flow back through unlimited numbers of layers.

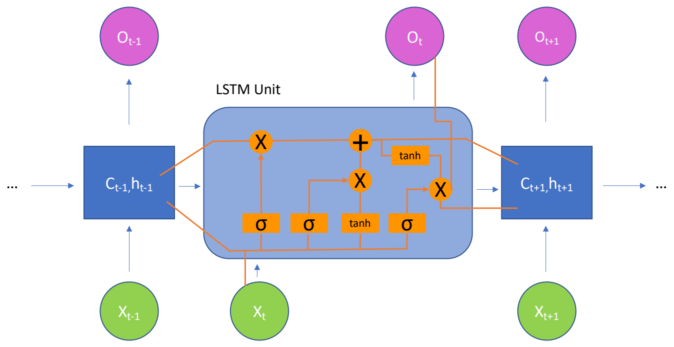

Figure 1: Diagram of a one-unit Long Short-Term Memory (LSTM)

LSTM uses a memory unit; that is why it can solve the problems requires of memories of events that happened very long-time steps ago. Figure 1, illustrates one unit of LSTM.

LSTM has a chance to choose among read, write or reset the sell at each step. Equation 1 shows the complete updating process in LSTM units [23].

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \tag{1}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$
$$h_t^l = o \odot \tanh\left(c_t^l\right)$$

2.2 Gated Recurrent Unit (GRU)

GRU was introduced in 2014 by k. Cho el al. [24] as an alternative solution to alleviate the complexity of LSTM units. GRU has fewer trainable parameters because it does not have the output layers that LSTM has. Figure 2 illustrates one unit of GRU.

Equation 2 shows the complete updating process in GRU units [23].

$$\begin{pmatrix} r \\ z \end{pmatrix} = \begin{pmatrix} sigm \\ \text{sigm} \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \tag{2}$$

$$\tilde{h}_t^l = \tanh\left(W_x^l h_t^{l-1} + W_g^l\left(r \odot h_{t-1}^l\right)\right)$$
$$h_t^l = (1-z) \odot h_{t-1}^l + z \odot \tilde{h}_t^l$$

Although LSTM and GRU are very similar to each other, they have some key differences:

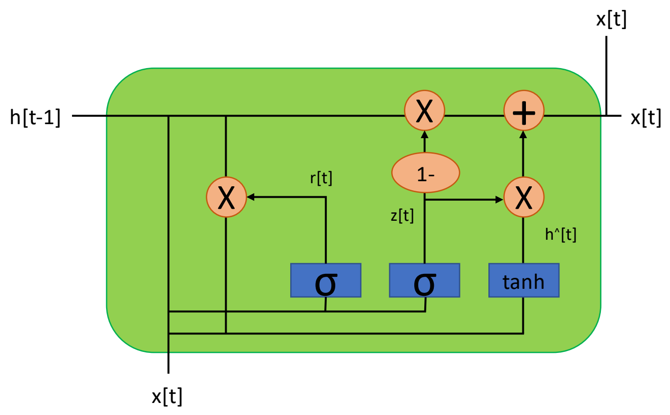

Figure 2: Diagram of a one-unit Gated Recurrent Unit (GRU)

- GRU has two gates (reset and update), while LSTM has three gates (Input, output and forget). Reset gate in GRU handles how new inputs are combined with previous memory, and update gate handles how much of the previous state need to keep. The update gate does the same work of that input and forget gates do in LSTM.
- Unlike LSTM gate, GRU gate does not have the c_t memory in each unit.

So, it is obvious that GRU is very similar to LSTM, and having less complexity and fewer parameters make it an interesting architecture to compare its performance to LSTM in the pouring scenario.

3 DATASET AND DATA PREPRATION

In this study, a dataset that includes 1307 motion sequences and their corresponding weight measurements was used. The data set has the shape of [# sequences, Max_length, # features]. The maximum length in this dataset is 1099, and any sequence that has a length less than 1099 was padded with zero at the end [25].

This dataset also has 10 features for each time steps:

[

θ_t	rotation angle at time t(degree)
f_t	weight at time t(lbf)
f_{init}	weight before pouring (lbf)
f_{ep}	weight while cup is empty (lbf)
f_{fin}	weight after pouring (lbf)
d_{rc}	diameter of the receiving cup (mm)
h_{rc}	height of the receiving cup (mm)
d_{pc}	diameter of the pouring cup (mm)
h_{pc}	height of the pouring cup (mm)
ρ	material density/water density (unitless)

]

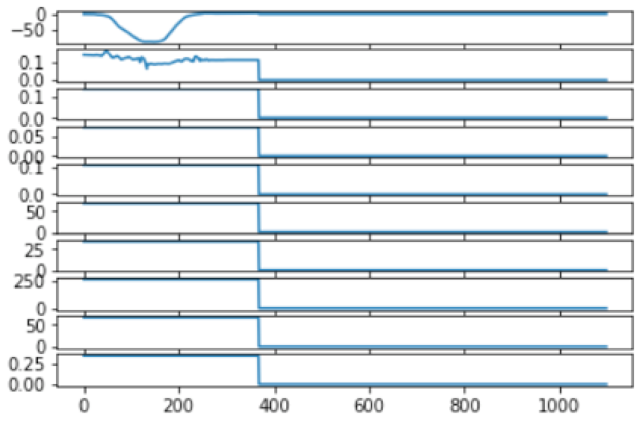

Figure 3: One-time sequence of the dataset

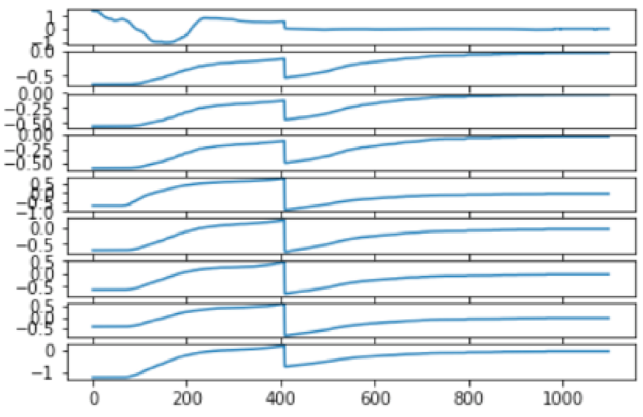

Figure 4: One-time sequence standardize of the dataset

Figure 4 illustrates a sequence of the dataset. The only rotation angle and weight change with time. Each feature has a different range size. In this work, the goal of this work is to predict the weight at time t (target/response) using 9 other features (predictors). The rest of this section talks about preprocessing features for neural network input.

3.1 Standardization

The original dataset was split into train, validation and test sets with ratios of 0.7, 0.2 and 0.1, respectively. The next step is dividing each of these three datasets into input and target. In this process, f_t is extracted from train_validation and test dataset. The final shape for these datasets became:

$Train\_Validation\_Input = (1176, 1099, 9)$
$Train\_Validation\_Output = (1176, 1099, 1)$
$Test\_Input = (131, 1099, 9)$
$Test\_Output = (131, 1099, 1)$

Standardizing a dataset means to rescale the dataset. Standardization was done in order to bring all input features into the same range. So, the mean of values becomes 0 and the standard deviation becomes 1. In this paper, Standard scale used to scale the dataset's features. Figure 4 illustrates a time sequence after applying the scaler function.
Test dataset scaled base on the data extracted from the training scaled process output.

4 Model

My goal in this paper was not to compete with previous work [21], instead I studied the performance of LSTM and GRU Network on this dataset. So, I chose 4 different models to compare their performance on this problem.

4.1 Designing Model

The Number of parameters was chosen as a key factor of similarity between models. So, all four models had the same number of parameters to become more likely to each other.
The dataset includes zero paddings, and it is important to ignore those values during the training process because they are not real values and they only there to fix the maximum length size.
Keras [26], introduced the "Masking" layer that can mask those data from going through the training process. In this paper, all of those models include one layer of Masking to ignore padding values.
Between each layer, I put a dropout layer to help model trains better.
Table 2 shows the parameter of these models.

4.1.1 Multi LSTM Model

This model includes a masking layer with a value of 0. (dataset include zero paddings) and input shape of (None, 9). The first layer should have input shape, and, in this case, nine features are the input. First LSTM has 55 hidden layers. The second has 27 and the third has 16. Between each layer, there is a dropout layer with a value of 0.2. At the end model used 3 fully connected layers with the size of 64, 29 and 1 (our output layer) with the activation function of relu.

4.1.2 Multi GRU Model

Like first model, this model also includes a masking layer at the first and then three layers of GRU with 64, 32 and 16 hidden layers. Also, it has 3 fully connected layers with dense of 64, 32 and 1 as the output with same activation function.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 July 2021 doi:10.20944/preprints202107.0252.v1

	Train Loss	Validation Loss	Test (1) Loss	Test (2) Loss	Time
Multi LSTM	0.0028	0.0010	0.0014	0.0329	65 min
Multi GRU	0.0024	0.0012	0.0013	0.0222	45 min
Single LSTM	0.0047	0.0020	0.0019	0.0310	28 min
Single GRU	0.0060	0.0033	0.0026	0.0227	18 min

Table 1: Results of training models for 20 epochs

	No. trainable parameters	No. Layer
Multi LSTM	29083	3LSTM + 3FC[1]
Multi GRU	29073	3GRU + 3FC
Single LSTM	28881	1LSTM + 1FC
Single GRU	28831	1GRU + 1FC

Table 2: Models architecture

4.1.3 Single LSTM Model

The difference between this model and multi layers LSTM is this model only include one LSTM layer with 80 hidden layers and one output with dense 1.

4.1.4 Single GRU Model

This model also includes one layer of GRU with 93 hidden layers and output dense 1.

4.2 Train parameters

Loss function is one of two mandatory parameters that need to compile model. Picking a good loss function can improve the final result of the model. The Mean-Squared error (MSE) loss function picked for this study. MSE is a measure of the quality of a predictor. It's a non-negative value and values closer to the zero are better.
The other important parameters for training is optimizer. Optimizer also plays a huge part in training process. Adam [27] chose for optimizer because Adam can generalize faster than other optimizers, so it can learn better in small size epochs. Optimizers have a learning rate parameters that can be helpful in learning process. All of the models used 0.01 as their learning rate in this paper.

5 Results and Conclusion

This section will cover the result of training and test evaluation. To be clear, 10% of the first dataset picked for testing purpose and also, Model tested with a new unseen dataset. So, we have two different test dataset, that the first one is picked randomly from the same dataset used for training.
The second test dataset which used in this study has different shape than the first one. The maximum length of new one is only 834.
New_Test_Input = (289, 834, 9)
To use the same scaler's variables which used by train_validation dataset, two datasets must have same number of columns, which means maximum length must become 1099. One way to fix this difference is to add zero to the end of each sequence time. So, after padding zero the shape becomes:
Padded_New_Test_Input = (289, 1099, 9)
StandardScaler function can now apply on the Input data and prepare the input data for evaluating with models.
Two number of epochs chose for this study (20 and 45). Purpose of this choosing was, estimate the performance of different models with small amount of training steps. First time models trained with the same train_validation dataset for 20 epochs. Results showed in table 1.

Table 1 shows that Multi GRU network perform better than the other models in small number of train steps. Increasing the amount of train steps will lead a model to learn better and produce a better result. So, for second part of test, the number of epochs increased to 45 and all the other parameters kept as the previous. Table 3 shows the results of the second test.

Table 3 shows that increasing the number of epochs mad a huge change in the result of multi layers GRU. Both GRU models (single and multi) predicted better results when I increased the number of epochs. However, LSTMs models made a less accurate prediction. Table **??** illustrate the performance increase/decrease in these two test processes.

Figure 5 to 10 illustrates 6 different results of output and prediction model for multi layers GRU. Figures 11-13 demonstrate how the other models performed after 45 epochs of training steps.

In conclusion, this study showed that GRU worked better than LSTM for this specific problems. Also, results show that multi layers of GRU performed better than using single layer. Also, the results show that GRU take less amount of time to train.
In the future, a better architecture can be used to generate a competitive result with existent architecture [21], and calculate the improvement performance. I would suggest creating the same architecture with GRU gates that introduce

	Train Loss	Validation Loss	Test (1) Loss	Test (2) Loss	Time
Multi LSTM	0.0013	0.0012	0.0021	0.0411	146 min
Multi GRU	0.0014	0.0009	0.0010	0.0007	101 min
Single LSTM	0.0006	0.0009	0.0027	0.0343	62 min
Single GRU	0.0008	0.0015	0.0018	0.0187	41 min

Table 3: Results of training models for 45 epochs

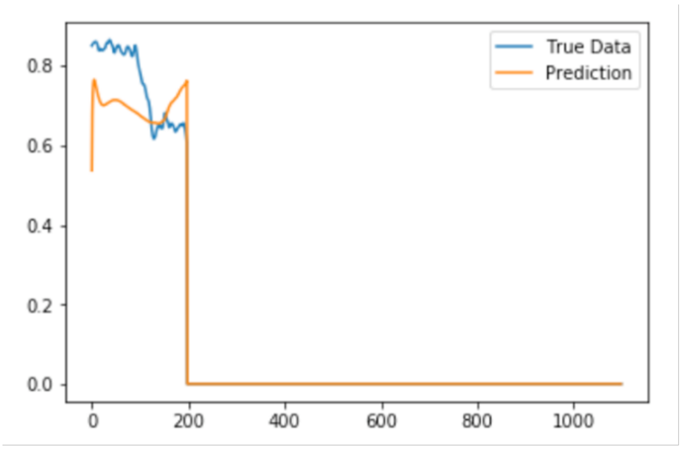

Figure 5: Prediction and True data of sequence 286

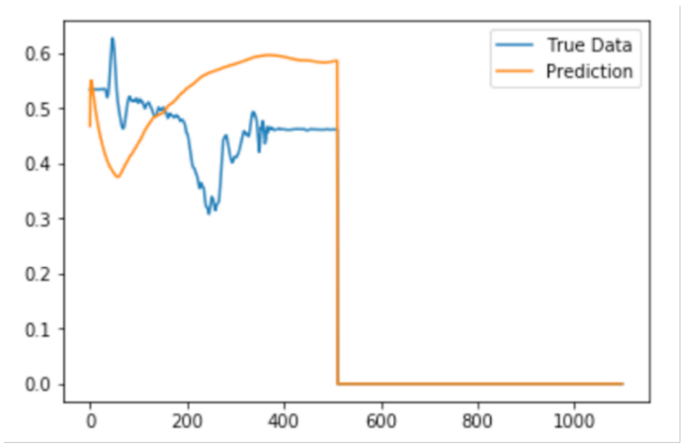

Figure 7: Prediction and True data of sequence 10

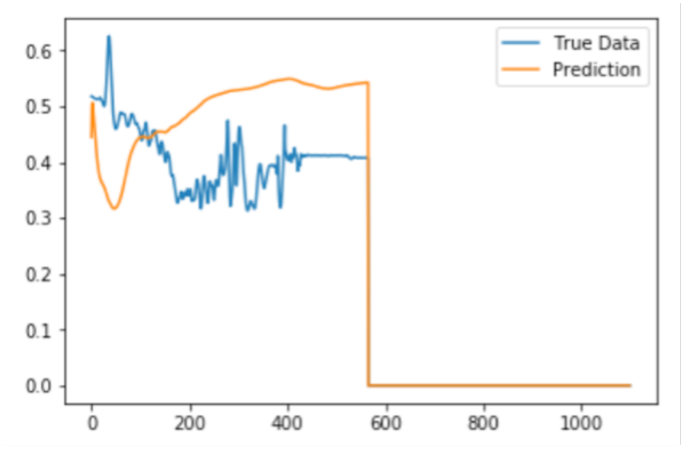

Figure 6: Prediction and True data of sequence 18

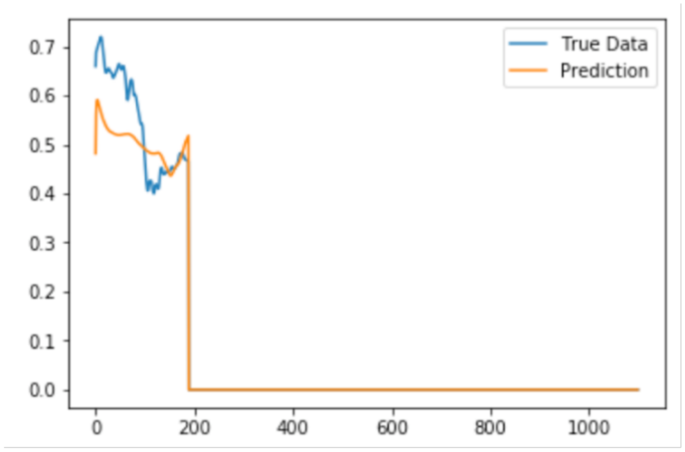

Figure 8: Prediction and True data of sequence 171

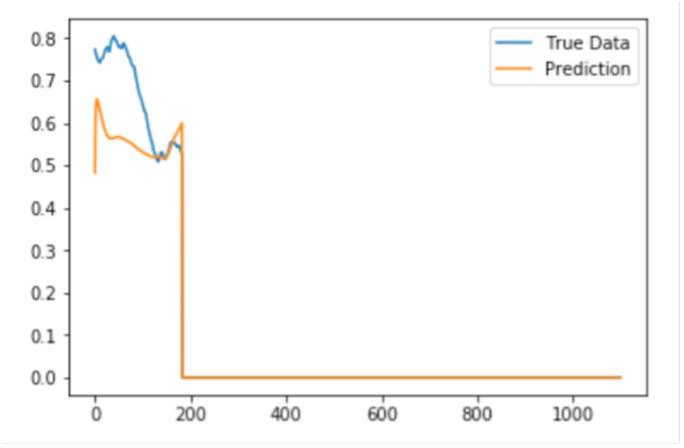

Figure 9: Prediction and True data of sequence 267

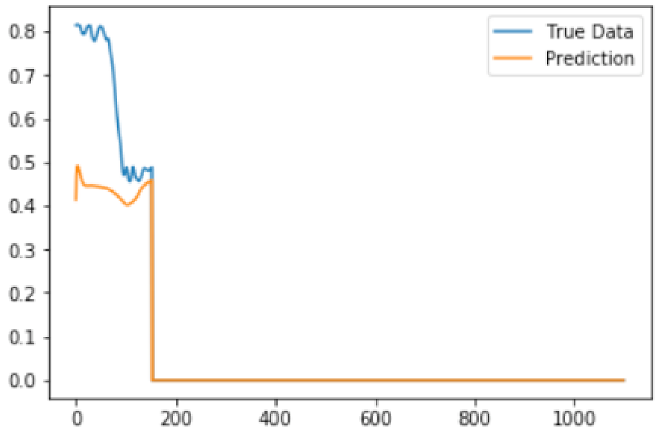

Figure 10: Prediction and True data of sequence 203

in the “Learning to pour” paper and use same environment to produce a better performance on this problem.

References

[1] Abien Fred M Agarap. A neural network architecture combining gated recurrent unit (gru) and support vector machine (svm) for intrusion detection in network traffic data. In *Proceedings of the 2018 10th international conference on machine learning and computing*, pages 26–30, 2018.

[2] Apeksha Shewalkar. Performance evaluation of deep neural networks applied to speech recognition: Rnn, lstm and gru. *Journal of Artificial Intelligence and Soft Computing Research*, 9(4):235–245, 2019.

[3] Yu Hu, Yongkang Wong, Wentao Wei, Yu Du, Mohan Kankanhalli, and Weidong Geng. A novel attention-based hybrid cnn-rnn architecture for semg-based gesture recognition. *PloS one*, 13(10):e0206049, 2018.

[4] Tsuyoshi Adachi, Kazuki Fujimoto, Sho Sakaino, and Toshiaki Tsuji. Imitation learning for object manipulation based on position/force information using bilateral control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3648–3653. IEEE, 2018.

[5] Lei Sun, Jun Du, Li-Rong Dai, and Chin-Hui Lee. Multiple-target deep learning for lstm-rnn based speech enhancement. In *2017 Hands-free Speech Communications and Microphone Arrays (HSCMA)*, pages 136–140. IEEE, 2017.

[6] Ben Athiwaratkun and Jack W Stokes. Malware classification with lstm and gru language models and a character-level cnn. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2482–2486. IEEE, 2017.

[7] Farah Shahid, Aneela Zameer, and Muhammad Muneeb. Predictions for covid-19 with deep learning models of lstm, gru and bi-lstm. *Chaos, Solitons & Fractals*, 140:110212, 2020.

[8] Peter T Yamak, Li Yujian, and Pius K Gadosey. A comparison between arima, lstm, and gru for time series forecasting. In *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, pages 49–55, 2019.

[9] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. *Eai Endorsed Transactions on Security and Safety*, 3(9):e2, 2016.

[10] Alaeddine Boukhalfa, Nabil Hmina, and Habiba Chaoui. A honey net, big data and rnn architecture for automatic security monitoring of information system. In *International Conference on Advanced Intelligent Systems for Sustainable Development*, pages 800–808. Springer, 2018.

[11] Armin Ziaie Tabari and Xinming Ou. A first step towards understanding real-world attacks on IoT devices. *arXiv e-prints*, March 2020. `https://arxiv.org/abs/2003.01218`.

[12] Armin Ziaie Tabari and Xinming Ou. A multi-phased multi-faceted iot honeypot ecosystem. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 2121–2123, 2020.

[13] Alex Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012.

[14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[15] Alex Graves. Offline arabic handwriting recognition with multidimensional recurrent neural networks. In *Guide to OCR for Arabic scripts*, pages 297–313. Springer, 2012.

[16] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772, 2014.

[17] Konstantinos Makantasis, Anastasios Doulamis, Nikolaos Doulamis, and Konstantinos Psychas. Deep learning based human behavior recognition in industrial workflows. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1609–1613. IEEE, 2016.

[18] Mustafa Raoof, Carolijn LMA Nota, Laleh G Melstrom, Susanne G Warner, Yanghee Woo, Gagandeep Singh, and Yuman Fong. Oncologic outcomes after robot-assisted versus laparoscopic distal pancreatectomy: Analysis of the national cancer database. *Journal of surgical oncology*, 118(4):651–656, 2018.

[19] Byunghun Choi, Wonsuk Lee, Gyuhyun Park, Youngwoo Lee, Jihong Min, and Seongil Hong. Development and control of a military rescue robot for casualty extraction task. *Journal of Field Robotics*, 36(4):656–676, 2019.

[20] S Smys and G Ranganathan. Robot assisted sensing control and manufacture in automobile industry. *Journal of ISMAC*, 1(03):180–187, 2019.

[21] Yongqiang Huang and Yu Sun. Learning to pour. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7005–7010. IEEE, 2017.

[22] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

[23] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[24] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[25] Yongqiang Huang and Yu Sun. A dataset of daily interactive manipulation. *The International Journal of Robotics Research*, 38(8):879–886, 2019.

[26] Keras.io. https://keras.io/.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.