INTERNATIONAL JOURNAL OF ROBOTICS
(THEORY AND APPLICATIONS)
E-mail: IJR@kntu.ac.ir

Real-time Compensatory Movement Detection in Upper Limb Rehabilitation Using Deep Learning Methods

J. Khoramdel[a], A. Moori[b], M. M. Moghaddam[a], and E. Najafi[b,*]

[a] Faculty of Mechanical Engineering, Tarbiat Modares University, Tehran, Iran
[b] Faculty of Mechanical Engineering, K. N. Toosi University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Submit: 2022-03-18
Revise: 2023-02-10
Accept: 2023-02-14

Keywords:
Deep Learning
Transformer
Recurrent Neural Networks
Gated Recurrent Unit
Long-Short-Term-Memory
Rehabilitation

ABSTRACT

This paper presents a real-time approach for detecting compensatory movements in upper limb rehabilitation for stroke patients using deep learning algorithms. The study applied Recurrent Neural Networks (RNN), Gated Recurrent Unit (GRU), Long-Short-Term-Memory (LSTM), and Transformer to analyze Microsoft Kinect data from the Toronto Rehab Stroke Pose dataset. The models were trained with focal loss to address imbalanced data distribution. The simulation results showed that the proposed deep learning algorithms are effective in detecting compensatory movements. The GRU-based models provide the fastest results and the transformer models exhibit the best accuracy and fastest inference time on the employed CPU.

* Corresponding address: K. N. Toosi University of Technology, Tehran, Iran,
E-mail address: Najafi.e@kntu.ac.ir.

1. Introduction

Many stroke survivors experience compensatory movements after the stroke, which results in poor functional outcomes [1]. After the stroke, it may take some time for the motor skills to be established, and the goal of the rehabilitation in stroke survivors is to improve these skills. However, it has been observed that the patients habitually do some additional movements to achieve the required motor skills [2]. For example, instead of stretching the arm to grasp an object, they may lean forward to reach that. Studies suggest that improper compensatory movement reduces the effect of rehabilitation on the patients [3]. Hence, it is essential to identify the compensatory movements during rehabilitation and prevent the patients from doing them.

Moreover, the joints of nine volunteers are captured in [4] during upper limb rehabilitation exercises with Microsoft Kinect. By analyzing the joint's statistical properties (mean and standard deviation) during movements, they concluded that these characteristics could be used for detecting compensation. However, no accuracy was reported for this approach. The researchers mentioned in [5] while a robotic arm can help a patient do the necessary exercises, it cannot ensure any compensation. Because of that, they used a Kinect camera in combination with their robotic arm to detect the compensation. The joint locations were obtained from the Kinect camera. Then, the joint is derived using simple vector analysis. A simple thresholding technique was used for finding the compensation based on the angle.

A comprehensive dataset has been collected, called Toronto Rehab Stroke Pose Dataset (TRSPD) from nine stroke survivors and ten healthy participants during exercises with Kinect [6]. They used a similar strategy, namely simple thresholding. However, simple thresholding is not a robust and accurate method. The authors of [6] have done another work, and Machine Learning algorithms like Support Vector Machine (SVM) and Long-Short-Term-Memory (LSTM) were employed to enhance the accuracy of the prediction [7]. Since the stroke survivors and healthy participant data had different characteristics, the models were trained and evaluated separately.

The TRSPD is a dataset with an imbalanced number of samples in each class. In order to solve the skewed classes issue, the authors of [8] combined several Machine Learning techniques like random under-sampling, oversampling, weighted cost (a cost function that is more sensitive to the classes with a lower number of instances) with the SVM classifier. Based on the idea which the random forest performs well on the imbalanced datasets, they also used random forest, weighted random forest, and isolation forest for classification. Like the experiments done in [7], the validation set was the same as the training set. Each algorithm was trained and evaluated on healthy participant data and stroke survivors' data independently. Although it is essential in machine learning to check for the generalization ability of the model by training and validating the algorithm on different sets, however [8] and [7] both trained and evaluated their approaches on a similar set.

A robot contact language has been proposed in [9] for manipulation planning and a manual control of a social robotic arm has been discussed in [10]. The implementation of reinforcement learning is another approach to handle unforeseen situation for a robotic system upon request [11]. In addition to the Kinect-based approaches, other equipment and sensor have been used in the literature for compensatory movement detection. The authors of [12] placed pressure mapping sensors on the chair and claimed that the patterns of the distribution of pressure on the chair are different for different compensations. They trained an SVM classifier to detect the compensation based on the extracted data from these arrays of pressure sensors.

The work done in [13] was based on attaching electrodes to the subject body and collecting the surface electromyography (sEMG) data to detect the compensation. The electrodes must be attached carefully to the patient's body in the appropriate locations. Commercial motion trackers like VICON are another option that provides accuracy within 0.01 mm, but the high price of setting up multiple cameras and their markers is a crucial limit [14]. The location of the joints can be tracked in the RGB images with the pose estimation network. In the reference [15], the famous OpenPose network [16] was applied to extract the location of the joints from the RGB images.

In the 2D images, the Depth information is lost, and because of that, 2 RGB cameras have been manipulated to reconstruct the lost information. It is computationally expensive to process two images simultaneously while running a heavy network like OpenPose. Among the approaches in the literature, detecting the compensatory movements based on the Kinect is more economical but yet practical. Hence, in this paper, the Kinect camera data will be used for identifying these abnormal movements during the exercises. Sequential composition is another control approach which has been implemented in [17], [18] to enable cooperation between robotic manipulators and mobile robots [19].

This paper investigates the use of deep learning algorithms, namely Recurrent Neural Networks

(RNNs), Gated Recurrent Units (GRUs), Long-Short-Term-Memories (LSTMs), and the Transformers to detect compensatory movements in stroke patients during upper limb rehabilitation. Moreover, it explores the use of Microsoft Kinect data from the Toronto Rehab Stroke Pose dataset and the training of the models using focal loss to address the issue of imbalanced data distribution.

2. Implemented dataset

In this paper, the TRSPD is used for training and evaluation, see [6] for detailed information. The both healthy (10 candidates) and stroke survivors (9 candidates) attended the experiments. At each frame, the X-Y-Z coordinates of 25 joints of the subject were captured in the Kinect frame, then translated to the participant-centric coordinate frame. The joint coordinates were normalized in the preprocessing to make the data invariant to the body size [7]. The compensatory movements in this dataset are 1- No Compensation (NC), 2- Lean Forward (LF), 3- Shoulder Elevation (SE), 4- Trunk Rotation (TR). The distribution of the data in each class has been given in Table 1. The NC class has the highest frequency in both subsets, and the dataset is imbalanced.

Table 1. The distribution of data in each class

Participant	NC	LF	SE	TR
Healthy	32,697	4,424	8,030	4,759
Stroke Survivors	19,103	290	1, 023	519

3. Deep learning algorithms

The idea of this work is take the coordinates of the joints and model them with sequential models. Each tuple of (X, Y, Z) is treated as a member of a sequence, and a whole sequence consists of 25 tuples. The problem of sequential data modeling is one of the noticed fields in artificial intelligence. Therefore, there are lots of algorithms developed for this problem recently. In this section, famous algorithms are introduced and explained.

3.1. Long-Short-Term-Memory (LSTM) unit

For the first time, this method was proposed by Hochreiter and Schmidhuber in [20]. This method was a significant improvement in the regiment of Recurrent Neural Networks (RNN) architectures. Since then, some LSTM versions have been developed that have different modifications of the naive implementation of LSTM. In this paper, we used the implementation that has been used in [21]. In this implementation, every LSTM unit memorizes c_t at time t. According to this, the output of the LSTM unit h_t obtain from [22] as

$$h_t = o_t * \tanh(c_t). \quad (1)$$

To compute LSTM output, we need output gated that that modulates the amount of memory content exposure. So, the output gate computes by

$$o_t = \sigma(W_o.x_t + U_o h_{t-1}) + V_O c_t + b_o, \quad (2)$$

where σ is a logical sigmoid function and b_o is an output gate bias. To update the memory cell, the both existing memory and new memory have been used as

$$c_t = f_t.c_{t-1} + i_t.\tilde{c_t}, \quad (3)$$

where f_t and i_t are the forget gate matrix and input gate matrix, respectively. These matrices are computed by

$$f_t = \sigma\big(W_f.x_t + U_f.h_{t-1} + V_f.c_{t-1}\big), \quad (4)$$

$$i_t = \sigma(W_i.x_t + U_i h_{t-1} + V_i.c_{t-1}) \quad (5)$$

The new memory is also calculated from

$$\tilde{c_t} = \tanh(W_c.x_t + U_c.h_{t-1}). \quad (6)$$

Please note that in these formulations, V_o, V_f and V_i are diagonal matrices. are diagonal matrices. The primary difference between traditional recurrent unit and LSTM is the introduced gates. These gates give LSTM the capability to handle the essential information over a long period and maintain the essential input features over time.

3.2. Gated Recurrent unit

The Gated Recurrent Unit (GRU) is an improved version of a standard recurrent neural network that using a gating mechanism [23]. The GRU uses cell memory to save sequence information, same as LSTM. As regards this method does not have separate memory cells. This feature is due to simpler architecture and faster trains. The GRU activation function at each time step computes from the compilation of previous time step activation h_{t-1} and candidate activation $\widetilde{h_t}$. This vector holds information for the current unit and passes it down to the network [22], as

$$h_t = (1 - z_t)\odot h_{t-1} + z_t \odot \widetilde{h_t}. \quad (7)$$

For obtaining h_t calculation for the update gate z_t and candidate activation $\widetilde{h_t}$ is needed. The update gate z_t helps the model determine how much of the past information (from previous time steps) is helpful to keep for future steps. To compute the update gate, (8) is used.

$$z_t = \sigma(W_z.x_t + U_z.h_{t-1}). \quad (8)$$

The candidate activation $\widetilde{h_t}$ or new memory content will use the reset gate to store the relevant

information from the past. It is calculated as follows

$$\widetilde{h_t} = \tanh\big(W.x_t + U.(r_t \odot h_{t-1})\big), \qquad (9)$$

where r_t is a set of reset gates. This gate must decide how much of the past information to forget. The

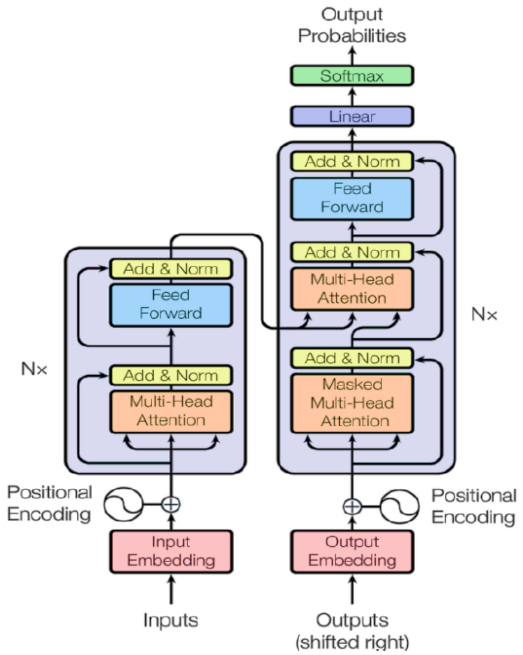

Figure 1. The transformer architecture, taken from [24]

formula of the reset gate is very similar to the update gate. The only difference is because of the weights and the gate's usage

$$r_t = \sigma(W_r.x_t + U_r.h_{t-1}). \qquad (10)$$

3.3. Transformer unit

The transformer networks are a type of deep learning algorithm that uses encoder-decoder architecture based on attention layers. Same as the last two methods, the transformer is used for sequential input data. However, the significant difference between this method and RNN is that the transformers do not necessarily process data in order and can pass input in parallel [24].

The transformer was initially released for translating from a source language to a target language. Because of that, like other algorithms for solving the translation task, the transformer consists of an encoder-decoder architecture. The transformer architecture is shown in Figure 1. There is no need for a decoder for the sequence classification task (like the one used in this paper). Hence, only the encoder part is considered. At the beginning of the encoder, a positional encoder is added to the projected input sequence. This positional encoder helps the network to remember the relative position of the members of the input sequence. Inside the encoder, the network maps an input sequence to a sequence of continuous representations with its sub-layers. The function of each encoder layer is to generate encodings that contain information about the connection of each part of the input together.

As shown in Figure 1, each encoder unit contains two blocks: multi-Head attention and feed-forward network. In the encoder, each sub-layer is followed by layer normalization [25]. The attention layer is a function that can access all previous states and weights. The attention layer does this work with three inputs called query, key, and value as follows

$$Attention(Q,K,V) = softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) \times V, \qquad (11)$$

where d_k is dimension of queries and keys. The multi-head attention sub-layer is the part of the architecture that uses the attention function. The output of multi-head attention is calculated as

$$MultiHead(Q,K,V) = concat(Head_1,..,Head_n)W_o, \qquad (12)$$

where $Head_i$ is defined by

$$Head_i = Attention\big(Q.W_i^Q, K.W_i^K, V.W_i^V\big). \qquad (13)$$

This is common to feed the three inputs of the multi-head attention with the same values. This idea is followed in this study.

3.4. Focal loss function

In object detection, lots of the area in the image belongs to the background, as described in [26]. The focal loss, which is an extension to the binary cross-entropy loss, was introduced to resolve this issue. The binary cross-entropy can be written as

$$CE = -p.\log\hat{p} - (1-p).\log(1-\hat{p}), \qquad (15)$$

where p is the true label and $\hat{p}$ is the predicted probability. The focal loss modifies this loss function with an imbalance handling parameter α and a modulating factor which is a function of γ and the $\hat{p}$ is given by

$$FL = -(1-\hat{p})^{\gamma}.p.\log\hat{p} - p^{\gamma}.(1-p).\log(1-\hat{p}) \qquad (16)$$

The modulating factor γ scales down the loss both for correctly classified and misclassified samples but the scaling factor lowers the loss value for true predictions much more than it lowers the

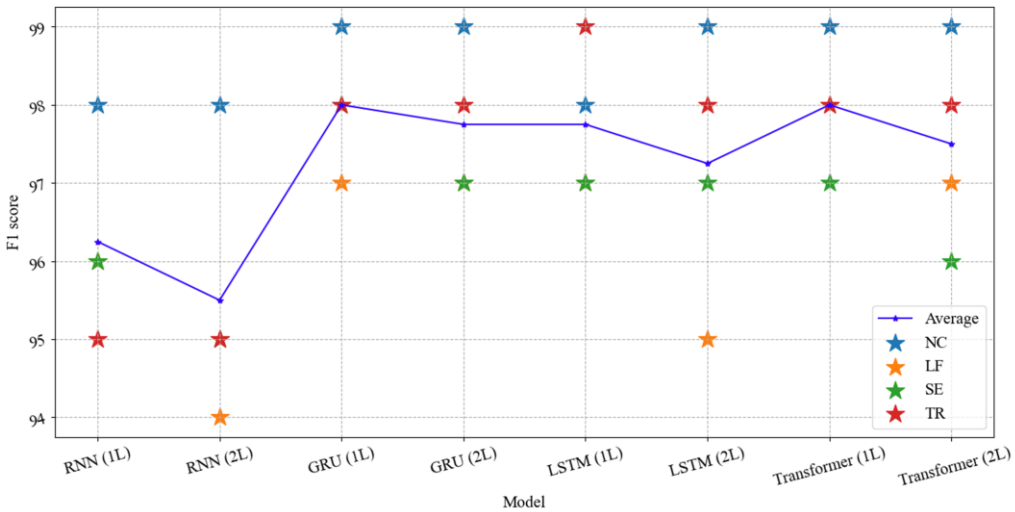

Figure 2. The evaluation results of the proposed models on the validation subset

loss value for misclassified instances. Thus, the importance of the misclassified instances increases. The focal loss can be extended to multi-class classification with a minor modification [27] as

$$FL = -\alpha(1-\hat{p})^{\gamma}.\,\mathrm{p}.\log(\hat{p}), \qquad (16)$$

where p is a one-hot vector which shows the true class and $\hat{p}$ is a vector of probabilities obtained from applying softmax function. Since the TRSPD dataset suffers from the imbalanced distribution, the modified version of the focal loss is applied in this paper to enhance the accuracy of prediction for all classes.

3.5. Implemented deep learning models

The models used in this paper are based on the recurrent (Simple RNN, GRU, LSTM) and also the attention (Transformer) mechanisms. The recurrent models were implemented with one recurrent layer with 20 units with the ReLU output activation. To check for the possibility of increment in the accuracy with increasing the depth of the network in this study, these models also have been deployed with two recurrent layers. However, the number of units in each layer was set to 10. In both cases, the output of the recurrent layers goes through a fully connected layer with ten neurons. This fully connected layer is connected to the classifier.

The transformer models require the same input size all over the transformer layers. So, in the case of the transformer models, only the depth of the network is increased from 1 to 2, but the number of the units in these layers remains unchanged. The number of the attention heads is set to 12. The attention is only calculated across the patch dimension. Same as the recurrent models, the output of the transformer layers is followed by a fully connected layer with ten neurons. The output of this fully connected layer is used for classification.

4. Simulation results

In this section, experiments will be conducted on the TRSPD dataset. All the models have been trained with a batch size of 128 for 50 epochs in all experiments. The initial learning rate is set to 0.01, and the Adam optimizer optimizes the loss function. Although it is not convenient in deep learning to train and test on the same set, this strategy is compared to the works done in the literature. All the models were trained on the healthy participant data and evaluated on the same dataset in the first experiment. The results are given in Table 3. The trained models have obtained better performance in terms of per class and average precision, recall, and F1 score compared to the works done in [8] and [7]. The LSTM and GRU with one recurrent layer have achieved the highest F1 score compared to the other models. Except for the simple RNN models, other F1 scores are close to each other.

After training on the healthy people data, the models were evaluated on the patients' data, but all the models failed to classify the patient's data with acceptable precision. The precision was only high in the NC class (over 90%), while the precision on all other classes was below 30%. This may be due to the discrepancy between the behavior of healthy and stroke survivors during the exercises, which makes different patterns in the collected sets.

In the second experiment, the data collected from the stroke survivors were used for training and evaluation. The results of the evaluation are presented by Table 4. In this case, the issue of an imbalanced dataset was more severe. The models

trained in [8] and [7] failed to reach an acceptable performance for all classes and only performed well on the class NC with the highest number of samples in the dataset. The focal loss has helped the networks to overcome the skewed classes issue. In this case, the transformer model with two layers obtained the best F1 score. After that, the transformer model with one layer, then the GRU with two layers, has the highest score.

In the LSTM and GRU case, the models with two layers reached better accuracy than the models with one layer, but the results are pretty close to each other (only 1% different in average F1-score). This can be because of the random weight initializing, data selection, or the number of the trainable parameters. In the case of the RNN, the model with one layer had better performance, while in the previous experiment, the situation was different. This is because the number of samples for stroke survivors was less than the total data for healthy people. The complexity and number of the trainable parameters were not adequate in the case of the RNN, and the accuracy increased with increasing the trainable parameters.

It should be noted that other algorithms were also implemented in [8] (SVM) and [7] (SVM + under-sampling, SVM + oversampling, etc.), but in the previous experiments, the best performing methods from those papers were compared to the implemented algorithms in this paper.

4.1. Generalization

To check whether the models could extend what they learned from the data to further unseen examples, the experiments were conducted again with a different strategy. Around 80% of the data from the healthy participant were randomly selected for training, and the rest of the dataset was used for validation.

The networks were trained from scratch, and Figure 2 shows the outcome of the experiment. Again, in this experiment, the RNN showed the lowest F1-score compared to the others. The transformer with one layer and the GRU models all reached the same performance in terms of the average F1-score. It seems the number of trainable parameters was a bit high for LSTM models and the transformer model with two layers. Although they showed an acceptable result, their accuracies are slightly lower than the accuracy of the GRU and one-layer transformer. Other metrics like precision, and recall are also calculated. The full results are presented in Table 7.

The stroke survivor's data were also split with the same rate (80% for training and 20% for validation). All the models were trained from scratch. Figure 3 represents the results. In this respect, the transformer model with two layers and GRU with one layer outperforms all the other models. After them, the transformer model with one layer has the best F1 score, and the LSTM models perform better than the simple RNN models. The evaluation information is given in Table 8.

4.2. Speed Test

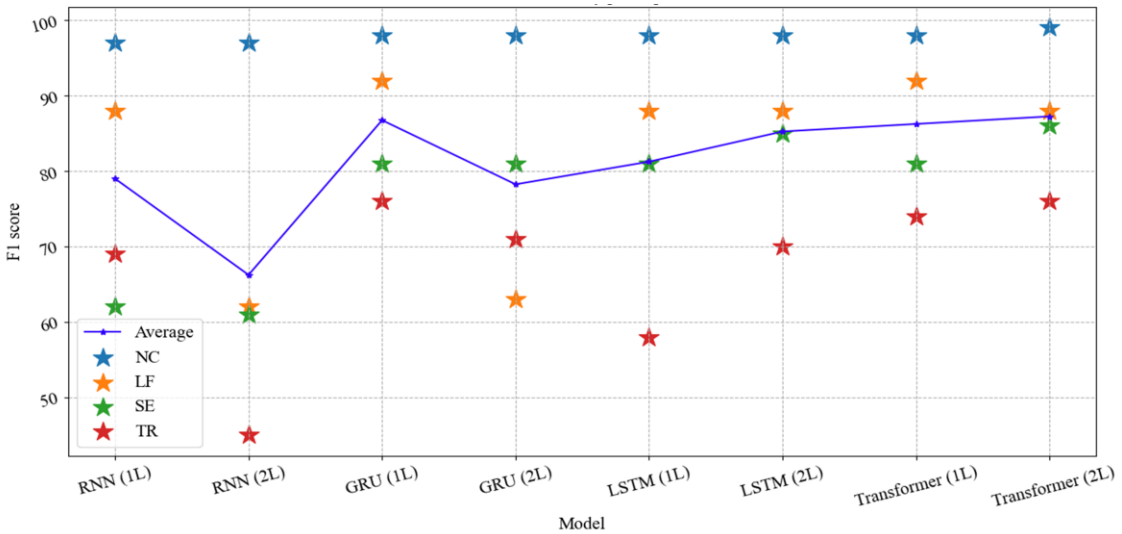

Figure 3 The evaluation results of the proposed models on the validation subset of the patient data, where (1L), and (2L) show the number of recurrent (or attention) layers

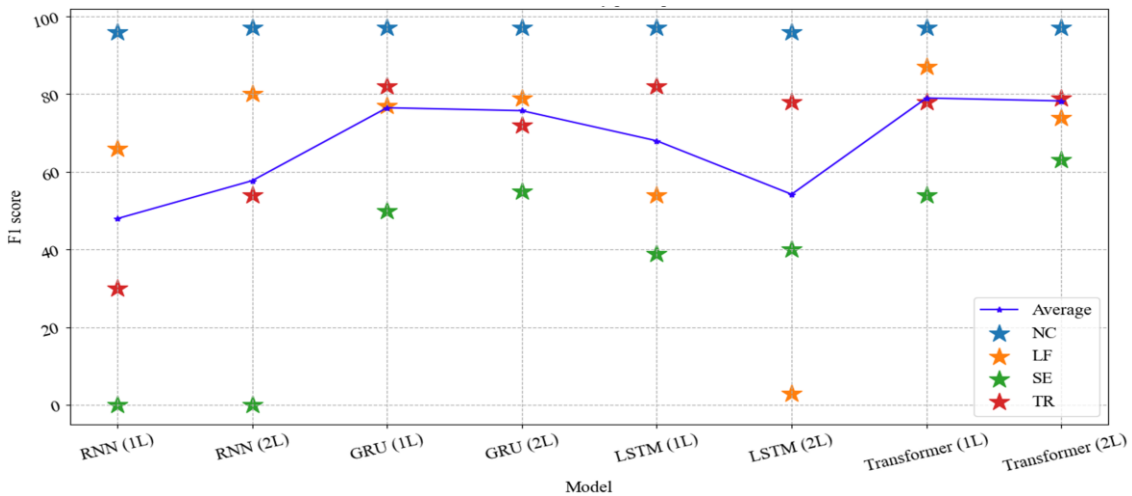

Figure 4 The evaluation results of the proposed models on the validation subset of the patient data, where (1L), and (2L) show the number of recurrent (or attention) layers

The compensatory movement detection algorithm must perform real-time predictions during the exercise. A speed test has been conducted on all of the models. Due to the recent advancement in deep learning, many research centers are equipped with GPU accelerated systems, but many medical rehabilitation centers do not have GPUs. Because of that, the tests are carried out both with GPU and CPU. The system specification is presented in Table 2. A random sample was chosen as the input for the networks, and the networks made one thousand predictions on that sample; then, the average prediction time was calculated and reported in **Error! Reference source not found.**.

When the models are running on the CPU, the transformer model with one layer is the fastest, and after that, RNN with one layer and transformer with two layers has the least prediction time. The GRU model has fewer parameters compared to LSTM, but on CPU, they are slower than LSTM models. The tests were performed multiple times, and in each test, the results followed the same pattern.

In order to take advantage of GPU accelerated calculation with GRU and LSTM, some conditions need to be satisfied. The activation of the output and recurrent units should be Tanh and Sigmoid, respectively. If the activations are chosen differently, GPU cannot speed up the calculations for these two models. The GRU model with one layer is the fastest when the GPU is enabled. In the case of the simple RNN and the transformer models, the improvement is insignificant.

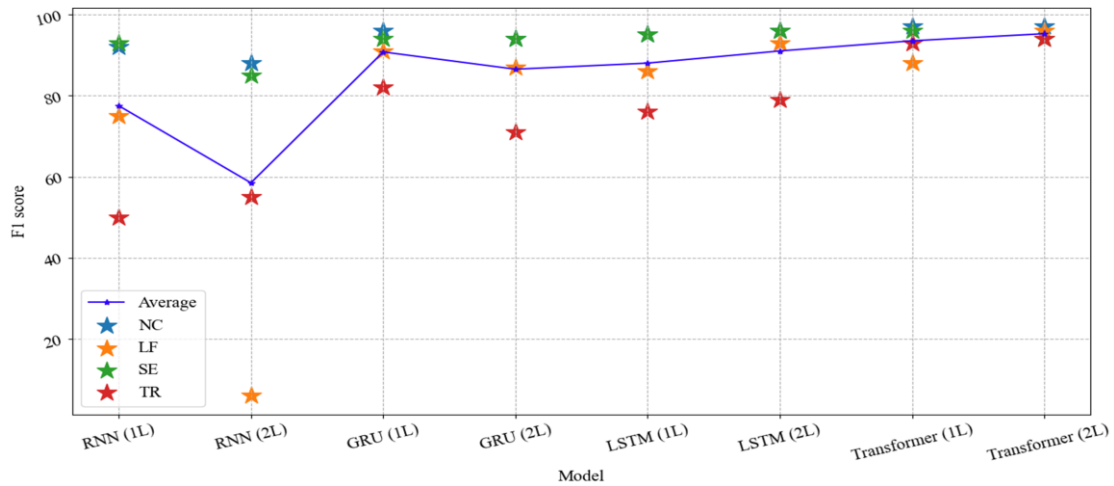

Figure 5 The evaluation results of the proposed models on the validation subset of the patient data, where (1L), and (2L) show the number of recurrent (or attention) layers

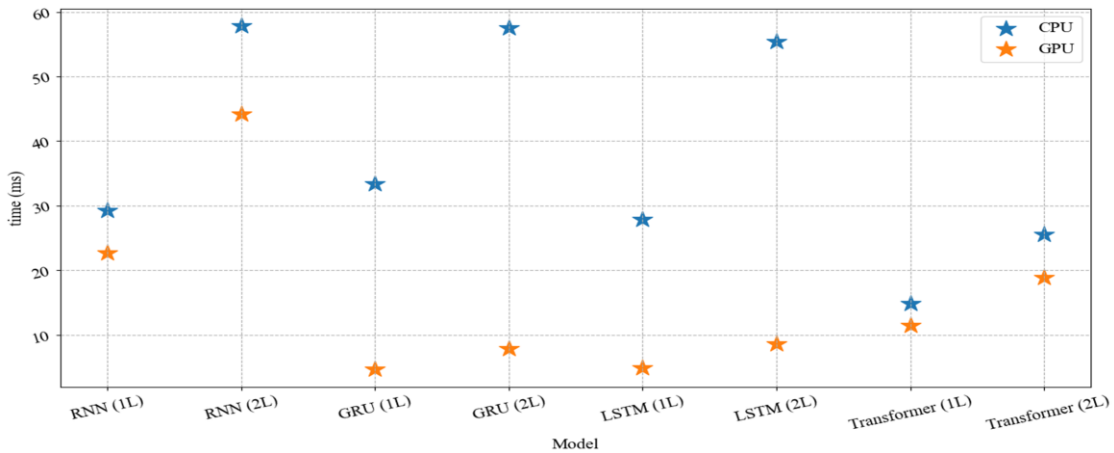

Figure 6 The simulation results of speed test

Table 2. Device specification

CPU	Intel(R) Xeon(R)
CPU Freq.	2.30GHz
RAM	12GB
GPU	Nvidia T4
GPU memory	12GB

4.3. Occlusion

So far, the assumption was that the Kinect had captured the location of 25 joints in each frame. However, during rehabilitation, some joints may be occluded, and therefore, the accuracy of the models would drop. In order to make the models robust to the occlusion, the following experiment has been conducted. A dropout layer with a dropping rate of 20% was added after the input layer. This dropout layer would synthesize the effect of occlusion by randomly removing the information of 5 joints in each iteration. Thus, the trained models do not depend on the presence of all the joints for each prediction. The models were trained on the training subset of the dataset for the healthy participant and stroke survivors.

Figure 5, shows the F1 score of all the models for each class on the validation subset of the healthy participant. Compared to the previous results, the average accuracy of all models has decreased, but the transformer-based models are affected more petite than the others. The evaluation result on the validation subset of the stroke survivor's data in Figure 6 indicates that the LSTM and RNN based models are highly affected by the occlusion while the transformer and GRU based models maintained their performance. Again, the transformer-based models have the highest accuracy. The full detailed results on healthy participant and stroke survivors' data are presented in Table 5 and Table 6, respectively.

5. Discussion and conclusion

This paper selected the TRSPD to train the RNN, GRU, LSTM, and transformer models to detect compensatory movements during the

rehabilitation exercises. The dataset included recorded coordinates of 25 joint subjects from healthy and stroke survivors during upper limb rehabilitation training. The dataset had a very imbalanced distribution, especially for the patient data. The modified version of the focal loss was deployed to help the model accuracy increase even for the samples with low frequency. The obtained results indicate that this goal was accomplished. In the experiments on the healthy participants, the GRU models showed better performance than the other ones, and the transformer model results were pretty close to the GRU models. In the experiments with the stroke survivors, the transformer models performed better than all the other techniques, while the GRU accuracy is close to transformer models.

Using the Microsoft Kinect in combination with a neural network model can detect compensatory movement. This economical approach can be accurate enough to be used in practice, but precaution needs to be taken. In some situations, some joints may be occluded, and this is not studied in this paper. Moreover, although the number of the data was adequate for training, the limited number of participants (9 stroke survivors and ten healthy people) may not be enough to claim that the generalization ability of the trained models is reliable enough to be used by itself in the clinics without the supervision of any experts. However, the algorithm can be employed to help them. This paper selected the TRSPD to train the RNN, GRU, LSTM, and transformer models to detect compensatory movements during the rehabilitation exercises.

The dataset included recorded coordinates of 25 joint subjects from healthy and stroke survivors during upper limb rehabilitation training. The dataset had a very imbalanced distribution, especially for the patient data. The modified version of the focal loss was deployed to help the model accuracy increase even for the samples with low frequency. The obtained results indicate that this goal was accomplished. In the experiments on the healthy participants, the GRU models showed better performance than the other ones, and the transformer model results were pretty close to the GRU models. In the experiments with the stroke survivors, the transformer models performed better than all the others, and the GRU accuracies were close to transformer models.

The speed test results show that the GRU-based models perform the best when using a GPU as an accelerator, with minimal latency. This means that real-time predictions can be performed efficiently during an exercise session. On the other hand, the transformer-based models performed well on the CPU, with a maximum delay of 20 milliseconds. This performance is sufficient for real-time applications with the Kinect device, which captures 30 frames per second. It is important to note that many medical rehabilitation centers do not have GPU systems and the test results indicate that the transformer models can perform well even on a CPU-only system.

Since occlusion is inevitable in real-life applications, a dropout layer was added at the input of the networks to synthesize the effect of occlusion. The dropout layer randomly removed the information of 5 joints in each iteration, thus making the models independent of the presence of all the joints. The models were trained on the training subset of the dataset for healthy participants and stroke survivors. The results showed that the average accuracy of all models decreased after the addition of the dropout layer, but the transformer-based models were less affected than the other models. The LSTM and RNN models were highly affected by the occlusion, while the transformer and GRU models maintained their performance, with the transformer models having the highest accuracy.

In conclusion, using the Microsoft Kinect in combination with a neural network model can detect compensatory movement. This economical approach can be accurate enough to be used in practice, but precaution needs to be taken. Although the number of the data was adequate for training, the limited number of participants (nine stroke survivors and ten healthy people) may not be enough to claim that the generalization ability of the trained models is reliable enough to be used by itself in the clinics without the supervision of any experts. However, the algorithm can be employed to help them.

6. Future work

The paper presented in this study has shown the feasibility of using deep learning algorithms, such as RNN, GRU, LSTM, and Transformer, to detect compensatory movements in upper limb rehabilitation. However, there is room for improvement and further research to fully realize the potential of deep learning algorithms for compensatory movement detection. This can be achieved by optimizing the models through incorporating other deep learning algorithms and feature engineering techniques, expanding the scope of the study to include other types of movements and parts of the body, and evaluating the performance of the models on larger and more diverse datasets. Additionally, exploring the relationship between compensatory movements and other rehabilitation outcomes, validating the models through clinical trials, and incorporating them into a rehabilitation system could provide

valuable insights into their practical utility and potential for widespread use.

References

[1] K.-H. Lee, "The role of compensatory movements patterns in spontaneous recovery after stroke," *Journal of physical therapy science*, vol. 27, no. 9, pp. 2671–2673, 2015.

[2] M. Reilly and K. Kontson, "Computational musculoskeletal modeling of compensatory movements in the upper limb," *Journal of Biomechanics*, vol. 108, p. 109843, 2020.

[3] M. Cirstea and M. F. Levin, "Compensatory strategies for reaching in stroke," *Brain*, vol. 123, no. 5, pp. 940–953, 2000.

[4] A. T. Garcia, A. L. d. S. Kelbouscas, L. L. Guimaraes, S. A. e Silva,˜ and V. M. Oliveira, "Use of rgb-d camera for analysis of compensatory trunk movements in upper limbs rehabilitation," in *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, vol. 1, 2020, pp. 243–248.

[5] E. B. Brokaw, P. S. Lum, R. A. Cooper, and B. R. Brewer, "Using the kinect to limit abnormal kinematics and compensation strategies during therapy with end effector robots," in *2013 IEEE 13th International Conference on Rehabilitation Robotics (ICORR)*, 2013, pp. 1–6.

[6] E. Dolatabadi, Y. X. Zhi, B. Ye, M. Coahran, G. Lupinacci, A. Mihailidis, R. Wang, and B. Taati, "The toronto rehab stroke pose dataset to detect compensation during stroke rehabilitation therapy," in *Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare*, 2017, pp. 375–381.

[7] Y. X. Zhi, M. Lukasik, M. H. Li, E. Dolatabadi, R. H. Wang, and B. Taati, "Automatic detection of compensation during robotic stroke rehabilitation therapy," *IEEE journal of translational engineering in health and medicine*, vol. 6, pp. 1–7, 2017.

[8] S. R. U. Uy and P. A. Abu, "Analysis of detecting compensation for robotic stroke rehabilitation therapy using imbalanced learning and outlier detection," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2020, pp. 432–437.

[9] E. Najafi, A. Shah, and G. A. Lopes, "Robot contact language for manipulation planning," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 3, pp. 1171–1181, 2018.

[10] P. Khajehpour and E. Najafi, "Design and manual control of a 3 degrees of freedom social robotic manipulator," in *Proceedings of the 21st International Conference on Research and Education in Mechatronics*, 2020, pp. 1–5.

[11] E. Najafi, R. Babuska, and G. A. Lopes, "Learning sequential compo-˜ sition control," *IEEE transactions on cybernetics*, vol. 46, no. 11, pp. 2559–2569, 2015.

[12] S. Cai, X. Wei, E. Su, W. Wu, H. Zheng, and L. Xie, "Online compensation detecting for real-time reduction of compensatory motions during reaching: a pilot study with stroke survivors," *Journal of neuroengineering and rehabilitation*, vol. 17, no. 1, pp. 1–11, 2020.

[13] K. Ma, Y. Chen, X. Zhang, H. Zheng, S. Yu, S. Cai, and L. Xie, "semgbased trunk compensation detection in rehabilitation training," *Frontiers in neuroscience*, vol. 13, p. 1250, 2019.

[14] N. Nordin, S. Q. Xie, and B. Wunsche, "Assessment of movement¨ quality in robot-assisted upper limb rehabilitation after stroke: a review," *Journal of neuroengineering and rehabilitation*, vol. 11, no. 1, pp. 1–23, 2014.

[15] Y. Fu, X. Wang, Z. Zhu, J. Tan, Y. Zhao, Y. Ding, and W. Chen,

"Vision-based automatic detection of compensatory postures of afterstroke patients during upper-extremity robot-assisted rehabilitation: A pilot study in reaching movement," in *2020 International Conference on Assistive and Rehabilitation Technologies (iCareTech)*, 2020, pp. 62–66.

[16] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," 2019.

[17] E. Najafi and G. A. Lopes, "Towards cooperative sequential composition control," in *Proceedings of the 55th IEEE Conference on Decision and Control*, 2016, pp. 4758–4763.

[18] E. Najafi, G. A. Lopes, S. P. Nageshrao, and R. Babuska, "Rapid learning˘ in sequential composition control," in *Proceedings of the 53rd IEEE Conference on Decision and Control*, 2014, pp. 5171–5176.

[19] G. A. Lopes, E. Najafi, S. P. Nageshrao, and R. Babuska, "Learning com-˘ plex behaviors via sequential composition and passivity-based control," in *Handling Uncertainty and Networked Structure in Robot Control*. Springer, 2015, pp. 53–74.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[22] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[23] K. Cho, B. Van Merrienboer, D. Bahdanau, and Y. Bengio, "On the¨ properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[25] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[26] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for´ dense object detection," 2018.

[27] M. Bacher, "LinkedIn multi-class focal loss," https://www.linkedin.com/pulse/multi-class-focal-loss-marcelo-bacherphd, accessed: 2021-09-12.

Table 3. The evaluation result of the examined models on the entire healthy people data

Models	class	Precision	Recall	F1	No. param
RNN (1 layer)	NC	98	98	97	734
	LF	99	90	94	
	SE	90	96	93	
	TR	95	97	96	
	Avg.	95	95	95	
RNN (2 layers)	NC	97	98	97	504
	LF	99	90	94	
	SE	94	93	94	
	TR	92	97	94	
	Avg.	96	95	95	
GRU (1 layer)	NC	99	100	99	1754
	LF	100	98	99	
	SE	99	96	98	
	TR	99	98	99	
	Avg.	99	98	99	
GRU (2 layers)	NC	98	99	99	1264
	LF	96	98	97	
	SE	98	94	96	
	TR	98	97	97	
	Avg.	98	97	97	
LSTM (1 layer)	NC	100	99	99	2174
	LF	97	95	98	
	SE	97	99	99	
	TR	99	95	99	
	Avg.	98	97	99	
LSTM (2 layers)	NC	99	99	99	1554
	LF	99	99	97	
	SE	97	97	98	
	TR	99	97	97	
	Avg.	98	98	98	
Transformer (1 layer)	NC	99	99	99	1491
	LF	98	99	98	
	SE	98	97	98	
	TR	100	99	98	
	Avg.	99	99	98	
Transformer (2 layers)	NC	99	99	99	2091
	LF	96	99	98	
	SE	98	97	98	
	TR	99	99	99	
	Avg.	99	99	98	
LSTM [7]	NC	81	90	86	-
	LF	84	77	81	
	SE	31	24	27	
	TR	66	45	53	
	Avg.	65	59	62	
Weighted Random Forest [8]	NC	-	-	82	-
	LF	-	-	80	
	SE	-	-	27	
	TR	-	-	53	
	Avg.	-	-	60	

Table 4. The evaluation result of the examined models on the entire patient data

Models	class	Precision	Recall	F1	No. param
RNN (1 layer)	NC	96	98	97	734
	LF	71	93	80	
	SE	69	65	67	
	TR	70	18	29	
	Avg.	76	68	68	
RNN (2 layers)	NC	97	98	97	504
	LF	78	90	83	
	SE	79	78	78	
	TR	70	26	38	
	Avg.	81	73	74	
GRU (1 layer)	NC	98	99	98	1754
	LF	85	81	83	
	SE	95	66	78	
	TR	74	85	79	
	Avg.	88	83	85	
GRU (2 layers)	NC	98	98	98	1264
	LF	77	97	86	
	SE	88	85	87	
	TR	77	71	73	
	Avg.	85	88	86	
LSTM (1 layer)	NC	97	99	98	2174
	LF	82	90	86	
	SE	84	87	85	
	TR	77	27	40	
	Avg.	85	76	77	
LSTM (2 layers)	NC	98	97	98	1554
	LF	84	87	86	
	SE	78	84	81	
	TR	66	77	71	
	Avg.	82	86	84	
Transformer (1 layer)	NC	99	98	98	1491
	LF	76	100	86	
	SE	79	95	86	
	TR	80	74	77	
	Avg.	83	92	87	
Transformer (2 layers)	NC	99	98	98	2091
	LF	87	95	91	
	SE	85	89	87	
	TR	79	73	76	
	Avg.	87	89	88	
LSTM [7]	NC	92	97	95	-
	LF	26	13	17	
	SE	22	13	7	
	TR	43	20	27	
	Avg.	46	36	31	
Weighted Random Forest [8]	NC	-	-	83	-
	LF	-	-	1	
	SE	-	-	6	
	TR	-	-	25	
	Avg.	-	-	29	

Table 5. The evaluation result on healthy participant data with occlusion

Models	class	Precision	Recall	F1	No. param
RNN (1 layer)	NC	86	99	92	734
	LF	95	65	75	
	SE	94	93	93	
	TR	100	34	50	
	Avg.	94	73	78	
RNN (2 layers)	NC	81	96	88	504
	LF	61	3	6	
	SE	75	98	85	
	TR	100	38	55	
	Avg.	66	54	58	
GRU (1 layer)	NC	92	99	96	1754
	LF	96	86	91	
	SE	100	89	94	
	TR	99	70	82	
	Avg.	97	86	91	
GRU (2 layers)	NC	90	98	94	1264
	LF	97	80	87	
	SE	91	96	94	
	TR	98	55	71	
	Avg.	94	82	86	
LSTM (1 layer)	NC	91	99	95	2174
	LF	100	75	86	
	SE	93	98	95	
	TR	98	62	76	
	Avg.	95	83	88	
LSTM (2 layers)	NC	93	99	96	1554
	LF	95	90	93	
	SE	95	97	96	
	TR	99	65	79	
	Avg.	95	88	91	
Transformer (1 layer)	NC	97	96	97	1491
	LF	81	98	88	
	SE	98	94	96	
	TR	97	89	93	
	Avg.	94	93	94	
Transformer (2 layers)	NC	98	97	97	2091
	LF	94	97	96	
	SE	97	90	94	
	TR	89	99	94	
	Avg.	95	96	95	

Table 6. The evaluation result patient data with occlusion

Models	class	Precision	Recall	F1	No. param
RNN (1 layer)	NC	86	99	92	734
	LF	95	65	75	
	SE	94	93	93	
	TR	100	34	50	
	Avg.	94	73	78	
RNN (2 layers)	NC	81	96	88	504
	LF	61	3	6	
	SE	75	98	85	
	TR	100	38	55	
	Avg.	66	54	58	
GRU (1 layer)	NC	92	99	96	1754
	LF	96	86	91	
	SE	100	89	94	
	TR	99	70	82	
	Avg.	97	86	91	
GRU (2 layers)	NC	90	98	94	1264
	LF	97	80	87	
	SE	91	96	94	
	TR	98	55	71	
	Avg.	94	82	86	
LSTM (1 layer)	NC	91	99	95	2174
	LF	100	75	86	
	SE	93	98	95	
	TR	98	62	76	
	Avg.	95	83	88	
LSTM (2 layers)	NC	93	99	96	1554
	LF	95	90	93	
	SE	95	97	96	
	TR	99	65	79	
	Avg.	95	88	91	
Transformer (1 layer)	NC	97	96	97	1491
	LF	81	98	88	
	SE	98	94	96	
	TR	97	89	93	
	Avg.	94	93	94	
Transformer (2 layers)	NC	98	97	97	2091
	LF	94	97	96	
	SE	97	90	94	
	TR	89	99	94	
	Avg.	95	96	95	

Table 7. The evaluation result of the proposed models on the validation subset of the patient data

Models	class	Precision	Recall	F1	No. param
RNN (1 layer)	NC	97	98	97	734
	LF	84	93	88	
	SE	79	51	62	
	TR	60	81	69	
	Avg.	80	81	79	
RNN (2 layers)	NC	95	99	97	504
	LF	96	45	62	
	SE	78	50	61	
	TR	74	33	45	
	Avg.	86	57	66	
GRU (1 layer)	NC	98	98	98	1754
	LF	86	100	92	
	SE	84	75	81	
	TR	69	85	79	
	Avg.	85	89	80	
GRU (2 layers)	NC	98	98	98	1264
	LF	96	47	63	
	SE	85	78	81	
	TR	73	86	79	
	Avg.	88	77	80	
LSTM (1 layer)	NC	97	99	98	2174
	LF	91	85	88	
	SE	89	74	81	
	TR	84	44	58	
	Avg.	90	74	81	
LSTM (2 layers)	NC	98	99	98	1554
	LF	81	97	88	
	SE	90	81	85	
	TR	78	63	70	
	Avg.	87	85	85	
Transformer (1 layer)	NC	98	98	98	1491
	LF	86	98	92	
	SE	83	79	81	
	TR	64	88	74	
	Avg.	83	91	86	
Transformer (2 layers)	NC	99	98	99	2091
	LF	84	93	88	
	SE	82	90	86	
	TR	82	71	76	
	Avg.	86	88	87	

Table 8. The evaluation result of the proposed models on the validation subset of the patient data

Models	class	Precision	Recall	F1	No. param
RNN (1 layer)	NC	97	98	97	734
	LF	84	93	88	
	SE	79	51	62	
	TR	60	81	69	
	Avg.	80	81	79	
RNN (2 layers)	NC	95	99	97	504
	LF	96	45	62	
	SE	78	50	61	
	TR	74	33	45	
	Avg.	86	57	66	
GRU (1 layer)	NC	98	98	98	1754
	LF	86	100	92	
	SE	84	75	81	
	TR	69	85	79	
	Avg.	85	89	80	
GRU (2 layers)	NC	98	98	98	1264
	LF	96	47	63	
	SE	85	78	81	
	TR	73	86	79	
	Avg.	88	77	80	
LSTM (1 layer)	NC	97	99	98	2174
	LF	91	85	88	
	SE	89	74	81	
	TR	84	44	58	
	Avg.	90	74	81	
LSTM (2 layers)	NC	98	99	98	1554
	LF	81	97	88	
	SE	90	81	85	
	TR	78	63	70	
	Avg.	87	85	85	
Transformer (1 layer)	NC	98	98	98	1491
	LF	86	98	92	
	SE	83	79	81	
	TR	64	88	74	
	Avg.	83	91	86	
Transformer (2 layers)	NC	99	98	99	2091
	LF	84	93	88	
	SE	82	90	86	
	TR	82	71	76	
	Avg.	86	88	87	

Biography

Javad Khoramdel received his BSc. in Mechanical Engineering from K.N. Toosi University of Technology, Tehran, Iran, in 2019. Since then, he is currently an MSc. student studying Mechatronics Engineering at Tarbiat Modares University, Tehran, Iran.

Ahmad Moori was graduated with a BSc. in Mechanical Engineering from K.N. Toosi University of Technology in 2019. He persuaded his education in MSc. in K.N. Toosi University of Technology, Tehran, Iran.

Majid Moghaddam received the PhD degree in Mechanical Engineering from the University of Toronto, Canada, in 1996. He is a Professor of Mechanical Engineering at the Tarbiat Modares University, Tehran, Iran. His current research, which focuses on applied robotics and robust H-infty control, is concerned with haptic robotics, rehabilitation robotics, inspection robotics, and rough terrain mobile robot design. He is a member of the Administrative Committee of Robotics Society of Iran.

Esmaeil Najafi received his PhD degree in Systems and Control from the Delft University of Technology in 2016. He was a postdoctoral research fellow at the Eindhoven University of Technology from 2016 till 2018. Esmaeil is an Assistant Professor at the K. N. Toosi University of Technology. His current research is focused on using artificial intelligence in Industry 4.0, intelligent robots, and mechatronic systems.