UDC 004.032:57.087.1 EDN: FQPDTF

RNN-BASED HAND GESTURE RECOGNITION USING sEMG

L. Assalama assalama.l@edu.spbstu.ru
V.V. Potekhin slava.potekhin@spbstu.ru

SPbPU, St. Petersburg, Russian Federation

Abstract

Using Deep Learning methods for processing sEMG signals expanded widely in the last decade, giving promising results concerning improving the performance in both directions: evaluation metrics values and fastness of training and testing stages. One of the most common methods in this field are RNN since their structure suits the nature of sEMG signal as a time series signal. In this article, we designed two models using RNN; one was one-layered and the other was multi-layered. We used the original signals values as the features, rather than extracting any other features like RMS, aiming at reducing the complexity of the processing. We used NinaPro-DB5 sEMG dataset. To overcome the obstacle of the dataset being of a small size, which most of such a type of biomedical signals datasets suffer from, we used augmentation by averaging the samples. This approach allowed building classifiers that can classify the samples into the entire 53 classes of NinaPro-DB5 sEMG dataset, with an accuracy of 99.6 and 99.8 % for 1L-RNN and ML-RNN respectively. While the processing time during the test stage ranges were in the range of 60–240 μs per sample for both models, which is fast. These results show that augmentation helped achieve state-of-the-art performance

Keywords

sEMG, hand gestures recognition, NinaPro-DB5, RNN, augmentation

Received 19.04.2025
Accepted 20.10.2025

Introduction. Many researches nowadays use Deep Learning methods to process sEMG signals for different purposes. There are many early examples of such approaches that led to promising results, which could be implemented in real life application, one example is the [1], they designed a ConvNet with TL model, and it scored an accuracy of 98.31 % when applied on the Myo dataset, and 68.96 % when applied on the NinaPro-DB5 dataset. Another example is work [2] did, they built their model on RNN, it scored 91 % when applied on the NinaPro-DB5 (considering only 41 gestures out of the 53 available in the dataset), work [3] also

used RNN-based model, trained and tested it using the NinaPro-DB5 dataset, considering only 8 gestures. The model accuracy reached 91 %. Other examples of using RNN with sEMG are found in [4–6], other approaches include using hybrid models that include RNN with other types of networks like CNN, like the work of [7, 8]. Another example of researchers who used special type of RNN like LSTM and GRU is the work [9].

We used in our work NinaPro-DB5 dataset because its samples have been collected and preprocessed systematically. Moreover, it is widely used by researchers in the field, which allows us to compare our results with a wide range of researches, including our previous work using XGBoost [10].

Unfortunately, like most of biomedical signal's datasets, the dataset is small. Having small number of samples makes it more difficult for the NN to learn, especially when considering a bigger number of sensors and a bigger number of classes (gestures considered). To overcome this difficulty, we augmented the dataset using averaging; using the same methodology from our previous work [10]. Where we chose the suitable signal length, following the same steps, then we augmented the dataset in two ways. The first is doubling the dataset size by averaging every two successive samples of each class, mostly averaging the samples of each subject, which enhances the common features of the samples of each subject. The second way is doubling the dataset size by $C_n = (n(n-1))/2$, by averaging each sample of each class with each sample of the rest samples of the same class.

RNN. The structure of a RNN is shown in Fig. 1. Let $X = \{x_1, x_2, \ldots, x_T\}$ be the input sequence of length T. The following equations represent an RNN.

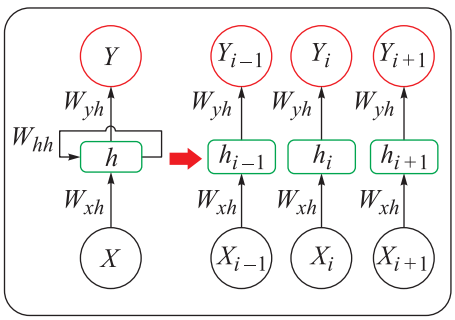

Fig. 1. RNN structure

The hidden state equation: $h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_n)$, where h_t is the hidden state at time t; f is a non-linear activation function; W_{hx} are the weights for the inputs of the hidden layer; x_t is the input at time t; W_{hh} are the weights for the hidden states; b_h is the bias of the hidden layer.

The output equation: $y_t = g(W_{yh}h_t + b_y)$, where y_t is the output at time t; g is a non-linear activation function; W_{yh} are the weights for the output layer (hidden layer to output layer connections); h_t is the hidden state at time t; b_y is the bias of the output layer.

This type of NN has many properties that makes it suitable for processing time-series signals like sEMG. These properties are as follows: sequential processing; memory; feedback loops; handling variable-length sequences; capturing temporal relationships.

This encouraged us to choose it, but most of the researchers use it with the features extracted from sEMG rather than with the original signals, like RMS, WL, etc. This adds extra load on the processing stage, whether before or during the learning stage (by using feature extracting layers to extract certain features). This means longer processing time too, which might affect the system performance when using it in real applications in real-time. Considering these factors, we decided to study RNN's performance using the original sEMG signals. Moreover, we studied the effect of the signal's length (number of time-steps of the sample) on the performance.

NinaPro dataset. Many researchers use NinaPro datasets (DB1–DB10) to train and test machine learning models, some of the early examples are:

– work [11]: authors applied K-NN on the NinaPro-DB5, taking into consideration all the 53 classes of the samples, this approach scored 61.3 % accuracy;

– work [12]: authors extracted a group of features from the EMG signal (RMS, WL, mDWT, MIST, MEAN), and used LDA to classify the samples into 40 classes, and it scored 61.14 % accuracy when tested.

Since it was first released, NinaPro spread widely in the machine learning society that handles EMG signals. Its datasets were systematically obtained, and they were collected with the help of subjects, who are diverse in terms of gender and physical attributes (height, weight). Moreover, intact and amputated subjects participated. This empowers the dataset because it covers a broad spectrum of the differences in EMG signals, so using it will elevate the model's applicability. Atzori and Muller explained the acquisition methodology fully is in their work [13]. All NinaPro datasets (DB1–DB10) are available free[1] with instructions on everyone.

Our approach. We used the original signals of the dataset without any features extracted from them. Each sample has a different length since they were obtained from a volunteer subject, which affected the start of recording the response and the stop moment according to the subject's reaction. To create a balance between the samples and to avoid the case where the model learns a signal better than any other one; we chose to set a fixed length for all the samples following the same methodology in [10].

[1] https://ninapro.hevs.ch/ (accessed: 06.03.2026).

In the first stage of preparing the dataset, we separated the samples and provided each one with its label. In the next step, classes were balanced. The classes in the original NinaPro-DB5 dataset are unbalanced, since the number of the Rest gesture (the first class) is equal to the number of the samples of the other classes together, this makes it challenging for the model to learn the gestures equally. Thus, we balanced the classed by choosing a number of Rest (class 0) samples close to the number of samples of the other classes (which are already close to each other except for class 0).

Only one armband's sensors signals were considered, which are 8 out of 16. The aim of this was to reduce processing complexity during training stage by considering a fewer number of values, and to reduce usage complexity during the use of the resulting model in real life application by requesting the user to wear one armband rather than two.

We used functions from the NP-DB5 and support *Python* libraries which we built specifically for this. *TensorFlow* packages and others were used for the training and processing, which was done using *Python*. The processing and training were done using Intel® Core™ i7-10870 processor.

The algorithm of this work is applied in four steps as follows.

1. Choosing the sample's size by balancing between getting as much samples as possible and keeping as much time steps in each sample as possible.
2. Doubling the samples by two or more.
3. Training the model.
4. Testing and evaluating the performance.

We used four criteria to evaluate the model's performance, accuracy, loss, training time and testing time. We tried different sizes as part of our study of the effect of the sample's size on the learning and overall performance of the model, and compared the performance using the original samples from the original dataset and using the augmented datasets.

Figure 2, *a* shows the structure of the 1L-RNN, it consists of three layers, an input layer with 8 channels, a simple RNN layer and an output layer with 53 units.

Figure 2, *b* shows the structure of the ML-RNN, it consists of five layers, an input layer with 8 channels, three simple RNN layers and an output layer with 53 units. This structure was inspired by *concept formation* or *conceptualization*, which is the process by which our brains decompose information and then recompose it to create our own concepts. This involves *decomposition*, that is breaking down complex information or experiences into simpler parts (more components), then, recomposition, where the brain recomposes these elements to form new concepts or ideas (less components).

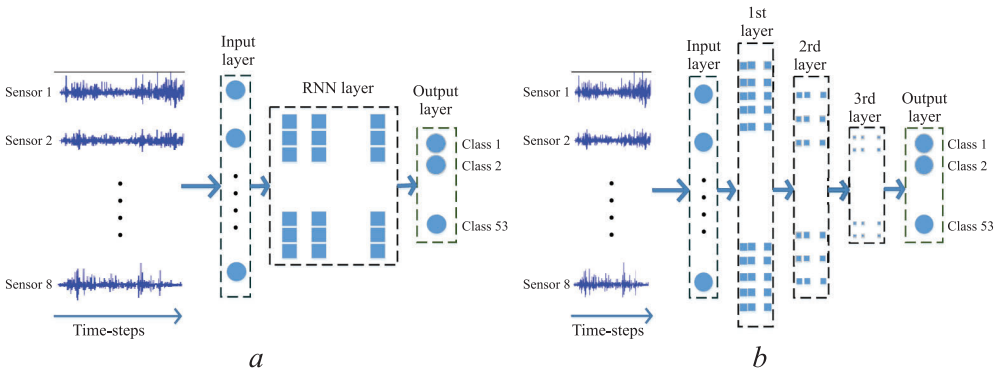

Fig. 2. Structures of 1L RNN (*a*) and ML-RNN (*b*) models

Results. *Augmentation effect.* In the first experiment, we used the original dataset without augmentation, and tested the model performance. However, this did not work, not with any length of the samples. Moreover, using longer samples did not affect the result significantly or usefully.

Next, we chose three signal's lengths (200, 400, 650 point). We doubled the dataset size by calculating the average of every two successive samples of a class, and add the result to the class. In addition, we doubled the dataset many times by calculating the average of each two samples in a class, and then add the result to the class. We repeated that (generating the doubled and multi-times doubled datasets) for each of the chosen signal's lengths, i.e., we had nine datasets to train and test the models, the original, the doubled and the multi-times doubled datasets for each signal's length. We tested both models in every stage: one-layered simple RNN; multi-layered simple RNN.

Figure 3 shows the accuracy and the loss of the model during the training and validation stages using the original dataset for training one-layered simple RNN model. It is obvious how the system failed completely to learn when it used only the original dataset (2 548 samples), even though it was trained using a high number of training epochs. Although it scored a very high accuracy during the training stage, it could not recognize the samples used to test it, which means that the high accuracy reflects an overfitting rather than real learning.

We trained both the models using the original dataset, and repeated the experiments using different input sizes. none of the models scored any relevant accuracy on any dataset. When using the doubled dataset (5 096 for the 650-point signal's length), 1L-RNN model accuracy reached 30 %, while the loss was too high, which makes the model's performance irrelevant.

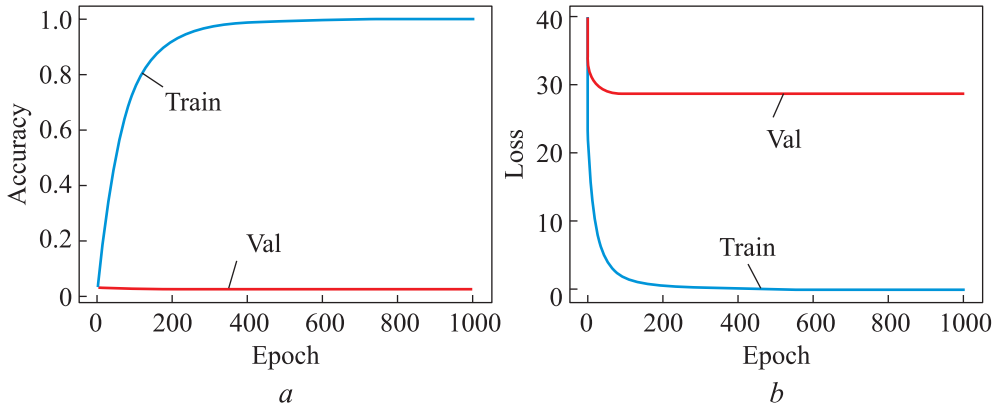

Fig. 3. 1L-RNN model accuracy (*a*) and loss (*b*) using the original dataset for training (2 548 sample, sample length 650 time-steps)

Figure 4 shows the accuracy and the loss of all the models when using the doubled dataset with different input sizes, only multi-layered RNN and multi-layered RNN-LSTM show meaningful results (good accuracy with low loss). It is clear that doubling the dataset's size by 2 is still not enough to get the desired performance.

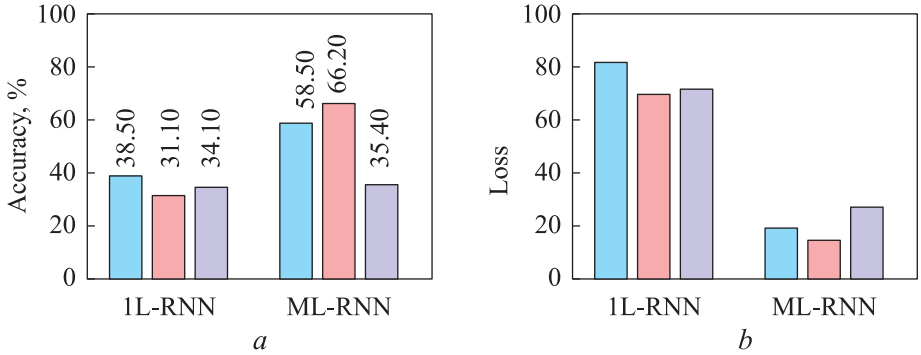

Fig. 4. Models' accuracy (*a*), and models' loss (*b*) using the doubled datasets (different input sizes 200, 400, 650)

1L-RNN. Figure 5, *a* shows the model accuracy. As it is obvious in the figure, the model scored the highest value of 99.6 % when it was trained using the signals with the size of 650 time-steps. Moreover, it differs by 2 and 2.7 % from the results when using signals of the size 400 time-steps and 200 time-steps respectively, which is unneglectable when handling biological signals. Taking into account that the features in our case are the raw values of the signals from the eight different sensors, this result is expected, since having more features and longer signals with more relations to time would help the model build

more connections and hence improve its performance. Moreover, that is approved too in Fig. 5, *b* which shows that training the model with the longest signals allowed the model to decrease the testing loss to about 0.02.

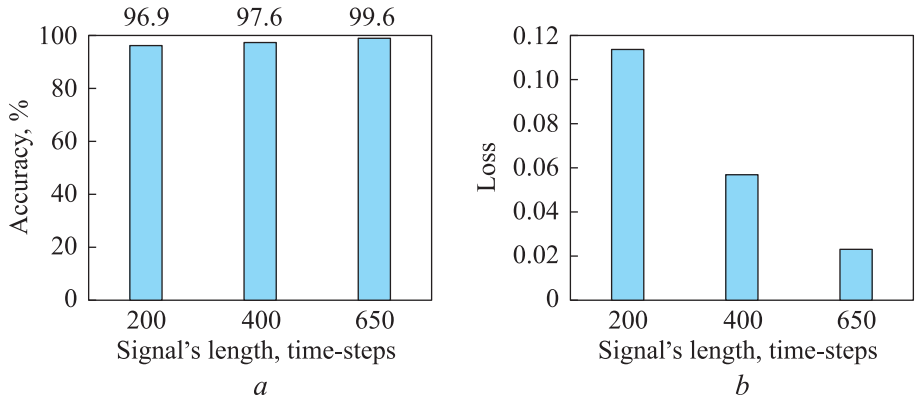

Fig. 5. 1L-RNN trained with the multi-times doubled dataset with different input sizes model's accuracy (*a*), and model's loss (*b*)

ML-RNN. Figure 6, *a* shows the model accuracy. The model scored high accuracy when trained on all the three different sizes datasets, and even its loss (Fig. 6, *b*) values are close to each other. This means that having many layers in the special structure we chose overcame the length issue, and this model works as desired even with smaller signals.

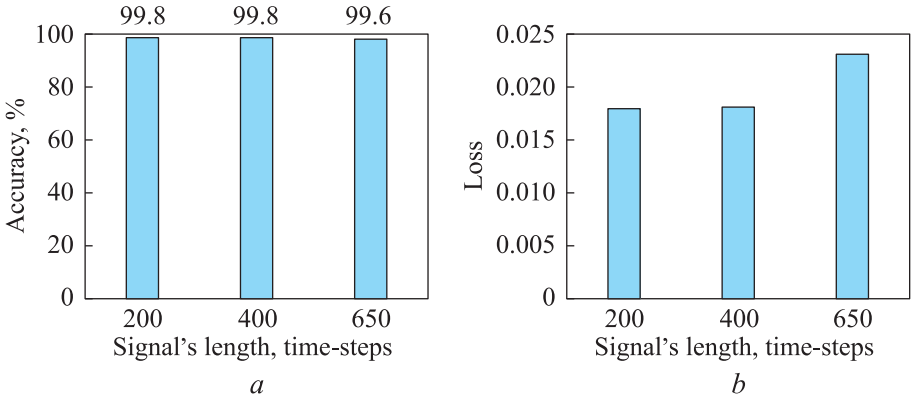

Fig. 6. ML-RNN trained using the multi-times doubled dataset with different input sizes model's accuracy (*a*), and model's loss (*b*)

Comparing 1L-RNN to ML-RNN. Figures 7 show the model accuracy and loss of 1L-RNN and ML-RNN respectively, when trained using the multi-times doubled dataset of the 650 time-step sEMG signals. Both models achieved high accuracy and low loss.

The two models have the same performance in terms of accuracy and loss (Table 1), but they differ when it comes to training time and testing time as shown in Fig. 7. 1L-RNN is a lot faster in the training stage with its 7 seconds training time, compared to the ML-RNN which needed 621 seconds, and that is justified completely since the later has much more parameters to update. However, surprisingly, the later needed less time to test.

Table 1

Comparing 1L-RNN / ML-RNN performance

Signal length, time-steps	Accuracy, %	Loss	Training time, s	Testing time, μs
200	96.9 / 99.8	0.11 / 0.02	13 / 175	138 / 799
400	97.6 / 99.8	0.06 / 0.02	14 / 174	170 / 798
650	99.6 / 99.6	0.02 / 0.02	7 / 621	212 / 60

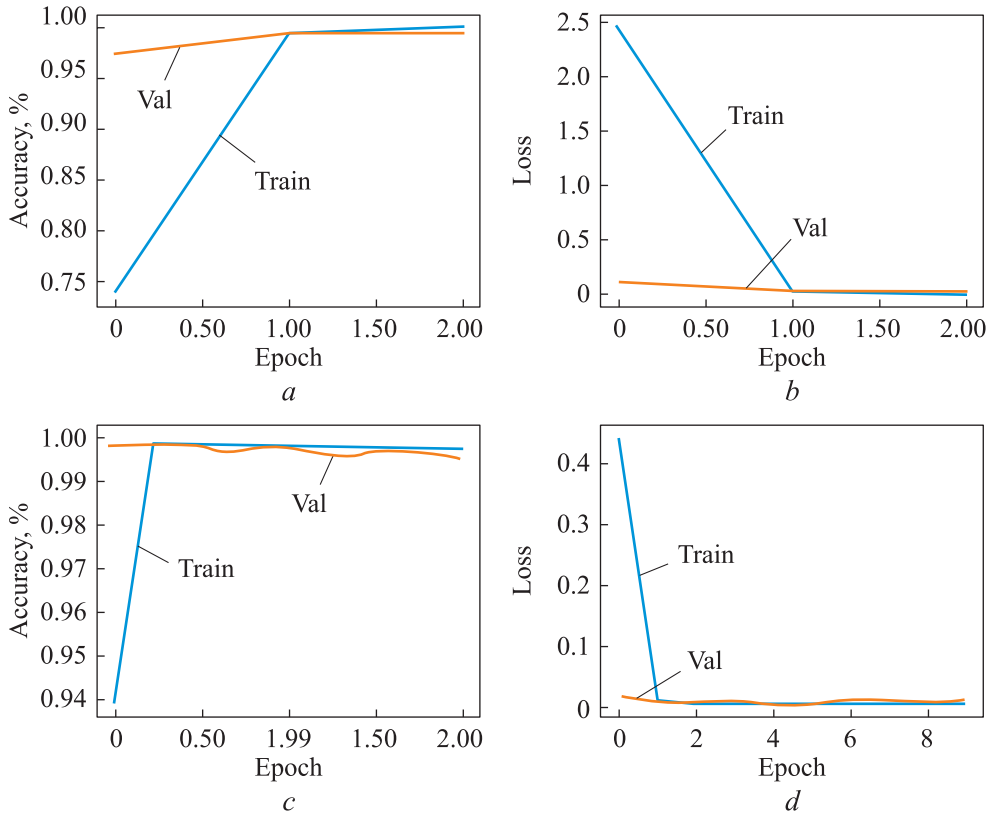

Fig. 7. Models' accuracy (*a*, *c*) and loss (*b*, *d*), trained using the multi-times doubled dataset of 650 time-steps signals:
a, *c*) 1L-RNN model; *b*, *d*) ML-RNN model

Discussion of the results obtained. Both the models we built, achieved state-of-the-art accuracy when compared to the results of other researches in the field. Table 2 shows the results of our approach and some of the state-of-the-art researches in the field. To compare them we must take into consideration the number of classes, since having more classes complicates the task for the ML model. Even then, our approach of augmentation and using the original values of the sEMG signals with different DL models could result in high performance.

Table 2

Model accuracy of our approach and some of the state-of-the-art approaches

Approach	Dataset	Gestures num (classes num)	Model accuracy, %
K-NN [5]	NinaPro-DB5	53	61.1 amputated 80.16 intact
LDA [6]	NinaPro	40	61.14
DNN [2]	NinaPro-DB5	41	91
RNN [3]		8	91
CNN [14]		52	74.51
EELM [15]		52	77.9
XGBoost [10]		53	98.6
1D-CNN-RNN [16]	Theirs	20	91 real-time 98.96 training
NN [17]	NinaPro-DB5	9	73.04
RNN [18]		8	87.9
RNN-GWO [19]		18	97.63
BRNN* [20]	–	15	93.33
Our 1L-RNN	NinaPro-DB5	53	99.6
Our ML-RNN		53	99.8

* Center of Intelligent Mechatronic Systems at the University of Technology at Sydney.

Conclusion. In this article, we designed two deep-learning models based on RNN to classify sEMG signals. The first one-layered RNN, and the second multi-layered RNN, whose structure was inspired by the principle of conceptualization. We studied the effect of the dataset's size and samples' size on the models' performance; we found that for the one-layered model it is better to use a signal that is long enough to cover the muscle's activity while performing the gesture. Outside this range, the signal's values have no significance; more-

over, they become a burden on the model. Moreover, the multi-layered model overcame this issue and could perform well regardless of the sizes of the input. In addition, the dataset has to have a big number of samples so the model can learn better, that is why we augmented the dataset by averaging every two samples of the same class and added the resulting signal to the dataset. As a result, the suggested models succeeded in classifying the samples resembling all the 53 gestures into the right class each, with an accuracy of 97.8 % and reached 99.8 % for 1L-RNN and ML-RNN respectively. All this indicate that RNN are suitable for handling raw sEMG signals.

Comparing these results to those of other approaches proves that using original values of the sEMG signals as the features helped the models score high accuracy and low testing time; this is promising in terms of providing sEMG-based control system with high performance and low complexity classifiers that will help improve their quality.

REFERENCES

[1] Côté-Allard U., Fall C.L., Drouin A., et al. Deep learning for electromyographic hand gesture signal classification using transfer learning. *IEEE Trans. Neural Syst. Rehabil. Eng.*, 2019, vol. 27, no. 4, pp. 760–771.
DOI: https://doi.org/10.1109/TNSRE.2019.2896269

[2] Chaiyaroj A., Sri-Iesaranusorn P., Buekban C., et al. Deep neural network approach for hand, wrist, grasping and functional movements classification using low-cost sEMG sensors. *IEEE BIBM*, 2019, pp. 1443–1448.
DOI: https://doi.org/10.1109/BIBM47256.2019.8983049

[3] Simão M., Neto P., Gibaru O. EMG-based online classification of gestures with recurrent neural networks. *Pattern Recognit. Lett.*, 2019, vol. 128, pp. 45–51.
DOI: https://doi.org/10.1016/J.PATREC.2019.07.021

[4] Koch P., Dreier M., Maass M., et al. RNN with stacked architecture for sEMG based sequence-to-sequence hand gesture recognition. *28th EUSIPCO,* 2021, pp. 1600–1604.
DOI: https://doi.org/10.23919/Eusipco47968.2020.9287828

[5] Ketykó I., Kovács F., Varga K.Z. Domain adaptation for sEMG-based gesture recognition with recurrent neural networks. *IJCNN*, 2019.
DOI: https://doi.org/10.1109/IJCNN.2019.8852018

[6] Sosin I., Kudenko D., Shpilman A. Continuous gesture recognition from sEMG sensor data with recurrent neural networks and adversarial domain adaptation. *15th ICARCV*, 2018, pp. 1436–1441. DOI: https://doi.org/10.1109/ICARCV.2018.8581206

[7] Hu Y., Wong Y., Wei W., et al. A novel attention-based hybrid CNN-RNN architecture for sEMG-based gesture recognition. *PLoS ONE*, 2018, vol. 13, no. 10, art. e0206049.
DOI: https://doi.org/10.1371/journal.pone.0206049

[8] Rahimian E., Zabihi S., Atashzar S.F., et al. Surface EMG-based hand gesture recognition via hybrid and dilated deep neural network architectures for neurorobotic prostheses. *J. Med. Robot. Res.*, 2020, vol. 5, no. 01n02, art. 2041001.
DOI: https://doi.org/10.1142/S2424905X20410019

[9] Samadani A. Gated recurrent neural networks for EMG-based hand gesture classification. A comparative study. *40th EMBC*, 2018.
DOI: https://doi.org/10.1109/EMBC.2018.8512531

[10] Potekhin V.V., Assalama L. Hand gesture recognition using sEMG with XGBoost, and averaging for augmentation. *26th Int. Sci., Educational and Pract. Conf. System Analysis in Engineering and Control*, 2022, vol. 2, pp. 74–88.
DOI: https://doi.org/10.18720/SPBPU/2/id23-83

[11] Atzori M., Müller H., Baechler M. Recognition of hand movements in a trans-radial amputated subject by sEMG. *IEEE 13th ICORR*, 2013.
DOI: https://doi.org/10.1109/icorr.2013.6650486

[12] Atzori M., Gijsberts A., Müller H., et al. Classification of hand movements in amputated subjects by sEMG and accelerometers. *36th Ann. Int. Conf. of the EMBC*, 2014, pp. 3545–3549. DOI: https://doi.org/10.1109/embc.2014.6944388

[13] Atzori M., Müller H. The Ninapro database: a resource for sEMG naturally controlled robotic hand prosthetics. *37th Ann. Int. Conf. of the IEEE EMBC*, pp. 7151–7154.
DOI: https://doi.org/10.1109/embc.2015.7320041

[14] Shen S., Gu K., Chen X.R., et al. Gesture recognition through sEMG with wearable device based on deep learning. *Mobile Netw. Appli.*, 2020, vol. 25, no. 6, pp. 2447–2458.
DOI: https://doi.org/10.1007/s11036-020-01590-8

[15] Peng F., Chen C., Lv D., et al. Gesture recognition by ensemble extreme learning machine based on surface electromyography signals. *Front. Hum. Neurosci.*, 2022, vol. 16, art. 911204. DOI: https://doi.org/10.3389/fnhum.2022.911204

[16] Li S., Zhang Y., Tang Y., et al. Real-time sEMG pattern recognition of multiple-mode movements for artificial limbs based on CNN-RNN algorithm. *Electronics*, 2023, vol. 12, no. 11, art. 2444. DOI: https://doi.org/10.3390/electronics12112444

[17] Mora M.C., García-Ortiz J.V., Cerdá-Boluda J. sEMG-based robust recognition of grasping postures with a machine learning approach for low-cost hand control. *Sensors*, 2024, vol. 24, no. 7, art. 2063. DOI: https://doi.org/10.3390/s24072063

[18] Simão M., Neto P., Gibaru O. EMG-based online classification of gestures with recurrent neural networks. *Pattern Recognit. Lett.*, 2019, vol. 128, pp. 45–51.
DOI: https://doi.org/10.1016/j.patrec.2019.07.021

[19] Emimal M., Hans W.J., Inbamalar T.M., et al. Classification of EMG signals with CNN features and voting ensemble classifier. *Comput. Methods Biomech. Biomed. Eng.*, 2025, vol. 28, no. 7, 1042–1056. DOI: https://doi.org/10.1080/10255842.2024.2310726

[20] Azhiri R.B., Esmaeili M., Nourani M. Real-time EMG signal classification *via* recurrent neural networks. *IEEE BIBM*, 2021, pp. 2628–2635.
DOI: https://doi.org/10.1109/bibm52615.2021.9669872

Assalama L. — Post-Graduate Student, School of Cyber-Physical Systems, SPbPU (Politekhnicheskaya ul. 29, St. Petersburg, 195251 Russian Federation).

Potekhin V.V. — Cand. Sc. (Eng.), Assoc. Professor, School of Cyber-Physical Systems, SPbPU (Politekhnicheskaya ul. 29, St. Petersburg, 195251 Russian Federation).

Please cite this article as:

Assalama L., Potekhin V.V. RNN-based hand gesture recognition using sEMG. *Herald of the Bauman Moscow State Technical University, Series Instrument Engineering*, 2026, no. 1 (154), pp. 47–58. EDN: FQPDTF