Hand Gesture Recognition Based on Electromyography Signals and Deep Learning Techniques

Mai H. Abdelaziz *, Wael A. Mohamed, and Ayman S. Selmy

Benha Faculty of Engineering, Benha University, Benha, Egypt
Email: may.hassan@bhit.bu.edu.eg (M.H.A.); wael.ahmed@bhit.bu.edu.eg (W.A.M.);
ayman.mohamed01@bhit.bu.edu.eg (A.S.S.)
*Corresponding author

***Abstract*—Hand gesture recognition based on Electromyography (EMG) signals is a challenging approach for developing natural and intuitive human-computer interfaces. In this paper, a hand gesture recognition system will be proposed that uses deep learning techniques, specifically a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) by merging them into one architecture called CNN+LSTM model. CNN is used to extract relevant features from the EMG signals, while the LSTM captures the temporal dynamics of the gestures. The proposed model is a fusion technique that combines the strengths of CNN and LSTM. Therefore, incorporating CNN+LSTM would be crucial in improving the accuracy of the model. The proposed system was trained and evaluated on two datasets publicly available. The first one, DualMyo, includes EMG signals recorded from one subject performing 8 different hand gestures, with each class of gestures recorded 110 times. The second dataset was collected from 36 subjects performing 8 different hand gestures. Results demonstrate that our proposed system achieved outstanding performance, with an average recognition accuracy for both data sets of approximately 99% for the DualMyo and about 97% for the second. To tackle the testing time issue, we introduce a second model that incorporates cascading CNN and max pooling layers, achieving a substantial reduction rate of 1/20 compared to the first model in testing time without significantly compromising recognition accuracy significantly, ultimately achieving the shortest testing time with good accuracy compared to the related methods. Experimental results demonstrate the efficacy of this approach, making it suitable for real-time applications in gesture-controlled systems.**

***Keywords*—hand gesture recognition, signal electromyography, Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM)**

I. Introduction

The Electromyography (EMG) signal is a biomedical signal that measures electrical activity produced by skeletal muscles. The EMG has been employed for many purposes including the estimation of finger motions, gesture recognition, diagnosis of neuromuscular illnesses and control systems such as robots and prosthetics. The development of relevant machine-assisted systems that promote the autonomy of individuals with special needs can be facilitated using EMG signals in conjunction with efficient gesture recognition [1].

Hand gesture recognition based on EMG signals is a challenging research area in Human-Computer Interaction (HCI) with various applications, including sign language recognition, prosthetic control, and virtual reality. Hand gesture recognition based on EMG signal has garnered significant attention in the field of assistive technology, particularly for individuals with limb amputations. Myoelectric prosthetic devices typically operate by analyzing and categorizing the recorded EMG signals, thereby enabling the synthesis of desired hand gestures [2]. Traditional approaches to hand gesture recognition, such as vision-based methods, have limitations such as being affected by lighting conditions and occlusion [3].

Utilizing Machine Learning (ML) for EMG-based gesture recognition is a popular yet challenging endeavor. It necessitates diverse and accurately labeled EMG datasets, considering variable EMG signals influenced by factors like fatigue and electrode placement. Moreover, meaningful feature extraction from raw signals, the selection of appropriate ML algorithms or neural architectures, optimal hyperparameter configuration, noise/artifact handling, user-specific model adaptation, real-time processing, intent recognition, and ethical considerations all come into play.

Deep learning is a subset of ML algorithms, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have achieved remarkable success in various applications, including image and speech recognition. One distinguishing factor of deep learning techniques, compared to conventional ML approaches, is their integration of feature extraction as part of the model construction, thereby eliminating the need for manually crafted features [4]. In recent years, deep learning has also been applied to EMG-based hand gesture recognition. A deep learning-based approach offers the

Manuscript received August 24, 2023; revised September 18, 2023; accepted October 11, 2023; published February 24, 2024.

potential to automatically learn features that are important for recognizing hand gestures from raw EMG signals. Therefore, the combination of EMG signals and deep learning techniques holds great promise for developing robust and accurate hand gesture recognition systems [5].

EMG-based hand gesture recognition utilizing CNN and Long Short-Term Memory (LSTM) networks separately faces certain limitations, prompting the exploration of a combined approach to mitigate these drawbacks. CNN, while adept at learning spatial features from data, might struggle with capturing the intricate temporal dynamics inherent in EMG signals. The grid-like nature of CNN doesn't inherently align with the sequential nature of EMG data, potentially causing the loss of essential temporal information. Furthermore, CNN might not always discern the most relevant features specific to EMG signals, leading to suboptimal performance [6].

On the other hand, LSTM networks excel at modeling sequences and temporal dependencies, making them well-suited for processing time-series data like EMG signals. However, the length of EMG sequences can lead to computational overhead during training. Additionally, vanishing gradient issues might arise when dealing with prolonged sequences, hindering the LSTMs' ability to capture long-term dependencies.

By combining CNNs and LSTMs, a hybrid architecture, which can address these challenges and harness the strengths of both approaches, emerges. CNNs can serve as effective feature extractors, capturing local spatial patterns within the EMG data. The CNN layers' outputs can then be fed into LSTMs, enabling the modeling of temporal dependencies over the sequence.

In this paper, an approach is proposed for hand gesture recognition based on EMG signals and deep learning techniques. Our approach consists of preprocessing EMG signals, building and combining a CNN+LSTM architecture, and training the model on a large dataset of hand gestures. The performance of the proposed approach is evaluated then compared with the state-of-the-art methods.

The rest of the paper is organized as follows. In Section II, we give a summary of the associated gesture recognition techniques. Section III provides the methodology of the proposed system. Results and discussion are presented in Section IV. Finally, the paper is concluded, and future work is described in Section V.

II. Literature Review

Hand gesture recognition has been an active research topic in recent years due to its potential applications in various fields such as human-computer interaction, robotics, and healthcare. Among different modalities, EMG signals have been widely used for hand gesture recognition because they are non-invasive, easy to acquire, and can reflect the muscle activities related to hand movements. In addition, deep learning techniques have shown promising results in EMG-based hand gesture recognition by automatically extracting discriminative features from the raw EMG signals.

Miguel *et al.* [7] used R-CNN network and Wavelet Packet Transform (WPT) feature extraction to learn the features from the EMG signals and achieved a recognition accuracy of 96.48%. However, their method utilized a 2-channel EMG signal collection approach, which may not have captured all the nuances of muscle activity associated with both the contraction and relaxation phases during different hand gestures. This limitation could have had an impact on the overall classification performance. In Ref. [8], a dilated CNN-based approach for EMG classification is presented, which can capture both local and global features from the EMG signals. Their method achieved an accuracy of 99% on a dataset of six hand gestures.

Recurrent Neural Networks (RNNs) have also been explored for EMG-based hand gesture recognition. For example, Li *et al.* [1] proposed a deep RNN-based approach using LSTM units. Their method achieved an accuracy of 97.75% on a dataset of four hand gestures. Miguel *et al.* [9] used LSTM network for EMG-based hand gesture recognition which achieved an accuracy of 95% for the DualMyo and about 91% for the NinaPro DB5 datasets. Toro *et al*. [10] introduced a gesture classifier based on RNN with LSTM units and dense layers. Their study aimed to enhance the model's scalability for embedded systems by reducing the number of Electromyography (EMG) channels and overall complexity. Using only four EMG channels, the proposed model successfully identified five hand movements, significantly reducing the required electrodes. The researchers trained the model with a dataset of gesture EMG signals recorded using a custom EMG armband, spanning 20 seconds. During training and validation, the model achieved up to 97% accuracy. In real-time testing, it demonstrated 87% accuracy.

Transfer learning has also been applied to EMG-based hand gesture recognition to improve the performance of deep learning models. Tayyip *et al.* [11] proposed a transfer learning approach that fine-tunes a pre-trained CNN on EMG dataset. achieving an accuracy of 98.40%.

In the realm of practical applications for gesture recognition, Yash *et al.* [12] introduced a controller designed for easy attachment to a user's arm band. This controller enables the control of an Unmanned Aerial Vehicle (UAV) using EMG signals. Gesture recognition is achieved through the utilization of an Artificial Neural Network, effectively harnessing the combined capabilities of both the user and the chosen UAV, which in this case, is a quadcopter.

In summary, EMG-based hand gesture recognition using deep learning techniques has achieved significant progress in recent years. Different deep learning architectures, such as CNNs, RNNs, and two-stream networks, have been explored to extract discriminative features from EMG signals. Transfer learning has also been applied to improve the performance of deep learning models on small EMG datasets. These developments have paved the way for the practical applications of EMG-based hand gesture recognition in various fields.

EMG-based hand gesture recognition using CNN and LSTM networks offers a powerful approach, but it's not without its disadvantages. When CNNs and LSTMs are used separately, limitations arise. CNNs are excellent at extracting spatial features from EMG data but may struggle to capture temporal dependencies in gestures. On the other hand, LSTMs excel at modeling temporal information but may not fully exploit the spatial features present in the EMG signals. However, combining these two techniques can effectively address these drawbacks. By integrating CNNs for spatial feature extraction and LSTMs for modeling temporal dependencies, the resulting hybrid model can achieve higher accuracy and robustness in hand gesture recognition. This approach benefits from the complementary strengths of both architectures, enabling it to capture intricate patterns and nuanced movements in EMG data. Consequently, it enhances the precision and reliability of hand gesture recognition systems, making them more suitable for applications in prosthetics, virtual reality, and human-computer interaction, where accurate and intuitive gesture recognition is paramount.

III. Methodology

In this section, we explain the data characteristics, preprocessing for model fitting and testing, the network's architecture, training methodology and performance metrics. The block diagram of the proposed hand gesture recognition network is shown in Fig. 1, which will be detailed in the next subsections.

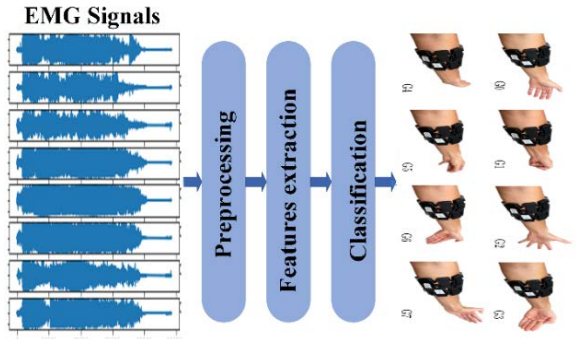

Fig. 1. The block diagram of the proposed hand gesture recognition network.

A. Datasets

The provided methodology was tested on two datasets, the first synthetic sequences from the UC2018 DualMyo dataset and second dataset we called EMG36, both used for EMG-based hand gesture detection. A fixed data split of 60% for training, 20% for validation, and 20% for testing is employed for all tests. While the performance of the validation set is used to optimize the hyperparameters of classification model

The UC2018 DualMyo was gathered from two EMG sensors (Myo) where the participant made 8 different hand gestures, each gesture has been recorded 110 times in total during 5 recording sessions. The UC2018 DualMyo dataset, comprising a substantial 880 samples, was further enhanced through data augmentation techniques, resulting in an extensive dataset containing 9,062 samples [13]. This augmentation involved generating additional variations of the recorded gestures, introducing controlled noise, and systematically expanding the dataset to improve the robustness and diversity of the data for training and evaluating hand gesture recognition models [9]. Hand shape of gestures is shown in Fig. 2.

The EMG36 dataset was gathered from a MYO Thalmic bracelet worn on a user's forearm; 36 participants made a sequence of motionless hand gestures. Two series are performed by the subject, each consisting of six (or seven) fundamental gesture which are: unmarked data (0), hand at rest (1), hand clenched in a fist (2), wrist flexion (3), wrist extension (4), radial deviations (5), ulnar deviations (6), and the last one extended palm (7) which was not performed by all subjects. The duration of each gesture was 3 seconds, with a 3 second break in-between each gesture [14]. The EMG36 dataset comprises a substantial 4,237,907 samples.

Preprocessing is a crucial step in gesture recognition based on EMG signals. While filtering is often used to remove noise or artifacts; in the proposed approach, only the normalization is applied. The EMG signals of the two datasets were normalized to a common scale to make them easier to compare and analyze. This involved scaling each signal to a range of (0, 1) or standardizing them to have zero mean and unit variance. Normalization helps to prevent any features from dominating others due to their larger magnitudes and to make the data more suitable for the model to learn.

Fig. 2. UC2018 DualMyo dataset gestures: (G0) rest, (G1) closed fist, (G2) open hand, (G3) wave in, (G4) wave out, (G5) double-tap, (G6) hand down, (G7) hand up [9].

B. CNN+LSTM Architecture

Both CNN and LSTM networks have been widely used for sequential data analysis. While LSTM are proficient at identifying long-term dependencies in sequential data, CNNs are great at learning local patterns in data. These two network types can be combined to handle time series classification tasks more effectively. A one-dimensional CNN is used for processing and analyzing one-dimensional sequential data, such as time series, audio signals, natural language sentences, or any other form of sequential data. The basic idea behind a 1D CNN is to apply convolutional operations to the input sequence to extract relevant features and patterns. The convolutional layers in a 1D CNN consist of small filters (kernels) that slide over the sequence and compute dot products with local regions, generating feature maps capturing different patterns in the data. The convolution operation in a CNN

is defined by a filter (kernel) K of size k, which slides over the input sequence. The output feature map F is computed as in Eq. (1).

$$F_{i=} \sum(k_i \times X_{i+j} F_{i=} \sum(k_i \times X_{i+j}) \quad (1)$$

where X is the input data, and the sum is taken over the filter size k and j ranges from 0 to k–1. The process continues for each position i, generating the entire feature map F.

The CNN component of the architecture extracts spatial features from the raw EMG signals by convolving them with a set of learnable filters. The resulting feature maps are then passed through a series of pooling and activation layers to reduce their spatial dimensionality and enhance their discriminative power. The output of the CNN is a high-level feature representation of the EMG signals that remains invariant to small translations and rotations [15].

LSTMs, on the other hand, are effective at modeling sequential data and capturing long-term dependencies in the input sequence. The LSTM component of the architecture models the temporal dependencies between the extracted features by learning a sequence of hidden states that capture the history of the signals. The core idea behind LSTM is its ability to retain important information over extended time intervals, mitigating the vanishing and exploding gradient problems commonly faced by traditional RNNs. This is achieved by LSTM cells, which can remember and forget information over long time periods.

The LSTM model's ability to control the information flow through the input, forget, and output gates, along with the cell state update, enables it to effectively capture long-term dependencies in sequential data and make informed predictions.

1) Input gate (i_t)

The input gate determines how much of the new input (current time step) should be added to the cell state. It takes the current input (x_t) and the previous hidden state (h_{t-1}) as inputs and produces the input gate activation (i_t) using a sigmoid activation function as in Eq. (2).

$$i_t = \sigma\,(W_{xi} \times x_t + w_{hi} \times h_{t-1} + b_i) i_t = \sigma\,(W_{xi} \times x_t + w_{hi} \times h_{t-1} + b_i) \quad (2)$$

where W_{xi} and W_{hi} are weight matrices, and b_i is the bias vector for the input gate.

2) Forget gate (f_t)

The forget gate decides what information from the previous cell state (c_{t-1}) should be retained or forgotten. It takes x_t and h_{t-1} as inputs and produces the forget gate activation (f_t) using a sigmoid activation function as in Eq. (3).

$$f_t = \sigma\,(W_{xf} \times x_t + w_{hf} \times h_{t-1} + b_f) f_t = \sigma\,(W_{xf} \times x_t + w_{hf} \times h_{t-1} + b_f) \quad (3)$$

where W_{xf} and W_{hf} are weight matrices, and b_f is the bias vector for the forget gate.

3) Cell state update ($ć_t$)

The cell state update captures the new candidate information to be added to the cell state. It takes x_t and h_{t-1} as inputs and produces the candidate cell state update($ć_t$) using the tanh activation function as in Eq. (4).

$$ć_t = tanh\,(W_{xc} \times x_t + w_{hc} \times h_{t-1} + b_c) ć_t = tanh\,(W_{xc} \times x_t + w_{hc} \times h_{t-1} + b_c) \quad (4)$$

where W_{xc} and W_{hf} are weight matrices, and b_c is the bias vector for the forget gate.

4) Cell state (c_t) update

The cell state (c_t) is updated by combining the information from the forget gate and the candidate cell state update as in Eq. (5).

$$c_t = f_t \times c_{t-1} + i_t \times ć_t c_t = f_t \times c_{t-1} + i_t \times ć_t \quad (5)$$

where × represents elementwise multiplication.

5) Output gate (o_t)

The output gate determines what information from the updated cell state should be used to produce the current hidden state (h_t). It takes x_t and h_{t-1} as inputs and produces the output gate activation (o_t) using a sigmoid activation function as in Eq. (6).

$$o_t = \sigma\,(W_{xo} \times x_t + w_{ho} \times h_{t-1} + b_o) o_t = \sigma\,(W_{xo} \times x_t + w_{ho} \times h_{t-1} + b_o) \quad (6)$$

where W_{xo} and W_{ho} are weight matrices, and b_o is the bias vector for the output gate.

6) Hidden state (h_t) update

The hidden state (h_t) is computed by applying the output gate to the cell state as in Eq. (7).

$$h_t = o_t \times tanh\, c_t\, h_t = o_t \times tanh\, c_t \quad (7)$$

where × represents elementwise multiplication.

The output of the LSTM is a compressed representation of the EMG signals that encodes both the spatial and temporal information [16]. The final output of the CNN+LSTM architecture is obtained by passing the output of the LSTM through one or more fully connected layers, which perform the classification task. The output layer uses a SoftMax activation function to produce a probability distribution over the different hand gesture classes [17].

C. Optimization

Optimization, in the context of machine learning and deep learning, refers to the process of finding the best set of parameters for a model that minimizes or maximizes a specific objective function. The objective function is typically a measure of how well the model performs on a given task, such as minimizing the error or loss on a training dataset or maximizing the accuracy on a validation dataset. We used NADAM optimizer, Nadam, short for Nesterov-accelerated Adaptive Moment Estimation, is an optimization algorithm used to update the weights of a neural network during the training process. It is an extension of two popular optimization algorithms: Nesterov Accelerated Gradient (NAG) and Adaptive Moment Estimation (Adam). The Nadam optimizer combines the benefits of both NAG and Adam, making it an efficient and effective optimization method.

NAG is a variant of the traditional gradient descent method that incorporates momentum to accelerate

convergence. It calculates the gradient of the loss function not only at the current position but also at a point slightly ahead in the direction of the momentum term. The updated weights are then based on this adjusted gradient. NAG helps to reduce oscillations and overshooting during optimization.

Adam is another popular optimization algorithm that uses adaptive learning rates for each parameter. It keeps track of both the first-order moment (mean) and the second-order moment (uncentered variance) of the gradients. This allows Adam to scale the learning rates differently for each parameter based on their historical gradient behavior, leading to more stable and efficient updates.

Nadam combines the concepts of NAG and Adam to leverage their advantages. During each iteration, Nadam computes the gradient using NAG to account for the momentum effect and then adapts the learning rates based on the historical gradients using Adam [18]. the Nadam optimizer computes the gradient of the loss function (g) with respect to the weights as in Eq. (8), then update the first-order momentum (m_t) as in Eq. (9), and second-order momentum (v_t) as in Eq. (10), then computes the NAG-corrected gradient(m_{th}) as in Eq. (11). Finally Updates the weights (w) using the NAG-corrected gradient and the adapted learning rate (a) as in Eq. (12).

$$g = d_{loss}/d_{weights} \tag{8}$$

$$m_t = B_1 \times m_t + (1 - B_1) \times g \; m_t = B_1 \times m_t + (1 - B_1) \times g \tag{9}$$

$$v_t = B_2 \times v_t + (1 - B_2) \times g^2 v_t = B_2 \times v_t + (1 - B_2) \times g^2 \tag{10}$$

$$m_{th} = m_t/(1 - {B_1}^t) m_{th=m_t/(1-{B_1}^t)} \tag{11}$$

$$w = w - a \times \left(m_{th}/\sqrt{v_t} + e\right) w = w - a \times (m_{th}/\sqrt{v_t} + e) \tag{12}$$

where B_1 and B_2 are the exponential decay rates for the first and second moments, respectively, t is the current iteration, and e is a small value to prevent division by zero.

Nadam combines the momentum effect of NAG with the adaptive learning rate mechanism of Adam, resulting in faster convergence and improved performance in many cases, especially for high-dimensional and noisy optimization problems commonly encountered in training deep neural networks.

D. Proposed System Model

The architecture of the model comprises input nodes, a convolution layer, and a max pooling layer. These are followed by a recurrent layer that has 1,024 LSTM cells. The output from this layer is then fed into a dense layer that consists of 512 units. Finally, a SoftMax transfer function is applied to generate a probability distribution for the classification output. The architecture of the model is shown in Fig. 3. The hyperbolic tangent activation function is used in the dense, convolution, max pooling, and LSTM hidden layers. To optimize the parameters, the NADAM optimizer is utilized with a learning rate of 0.001 and a batch size of 256. Hyperparameters were carefully chosen by systematically experimenting with various configurations to identify the most optimal settings. We computed several evaluation metrics, including accuracy, precision, recall, F1-score, and generated a confusion matrix once the model has been tested on all data splits (training, validation, and testing).

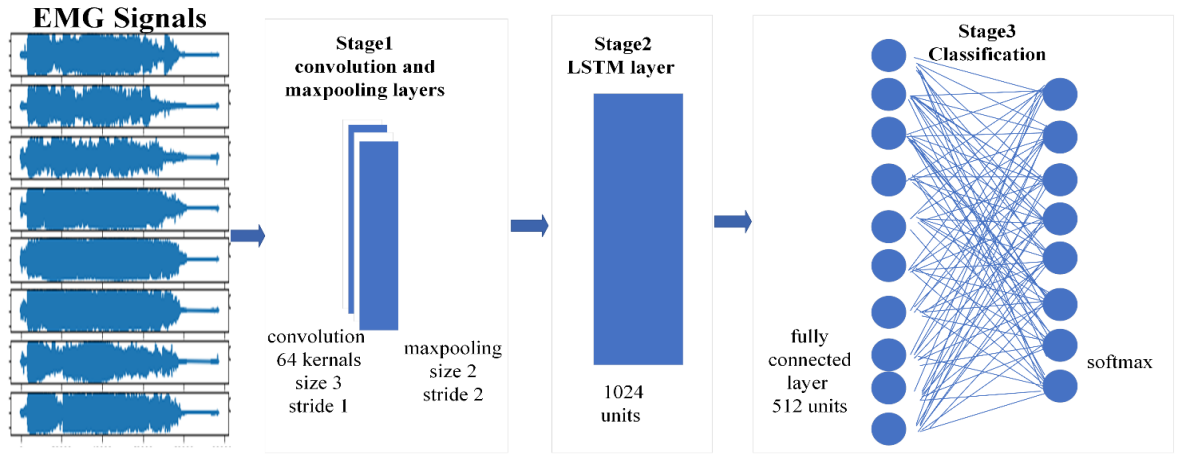

Fig. 3. The model architecture.

Accuracy is the proportion of correctly predicted instances out of the total instances in the dataset. It measures overall correctness. While accuracy is informative, it might not be suitable for imbalanced datasets where one class is significantly more prevalent than the other. Precision is a metric that focuses on the accuracy of the positive predictions made by the model. It calculates the proportion of true positive predictions (correctly predicted positives) out of all instances predicted as positive. Recall, also known as sensitivity or true positive rate, measures the model's ability to correctly identify all instances of a specific class. It calculates the proportion of true positive predictions out of all instances that actually belong to the positive class. Recall is crucial when the cost of false negatives is high, as it quantifies the model's ability to capture all instances of the positive class.

F1-score is the harmonic mean of precision and recall, offering a balance between accurate positive predictions and capturing all positive instances.

A confusion matrix is a tabular representation that displays the model's predictions against the actual class labels in a classification problem. It consists of four main values: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). These values help quantify the model's performance, making it easier to calculate metrics like accuracy, precision, recall, and the F1-score. The confusion matrix is a valuable tool for understanding the distribution of predictions across different classes and evaluating the model's strengths and weaknesses. These metrics and concepts collectively provide a comprehensive view of a model's performance, enabling practitioners to assess its accuracy, ability to make precise predictions, capacity to capture all relevant instances, balance between precision and recall, and the distribution of predictions across different classes [7].

IV. Result and Discussion

The Keras library using TensorFlow was used to define and train the networks. The hardware used was a laptop with an Intel i7-7820HQ CPU and 16 GB of RAM. The classification models were trained following the methodology described earlier. The performance of the CNN + LSTM model is compared with different deep learning techniques. Fig. 4 shows the training and validation accuracy of the model, where the final accuracy of training and validation were 99.66% and 98.64%, respectively for the DualMyo dataset and 98.94 % and 97.24% for the EMG36 dataset respectively. In Fig. 5, the test split accuracy, precision, recall, and F1-score is displayed. We display them for both datasets. In terms of accuracy, accuracy for DualMyo and EMG36 is 0.989 and 0.972 respectively. And the precision is 0.9896 and 0.9772 for two datasets respectively. And for the recall is 0.989 and 0.9719 for two datasets respectively. Finally, the F1-score is 0.9893 and 0.9720 for two datasets, respectively. The performances matrices are similar for both datasets.

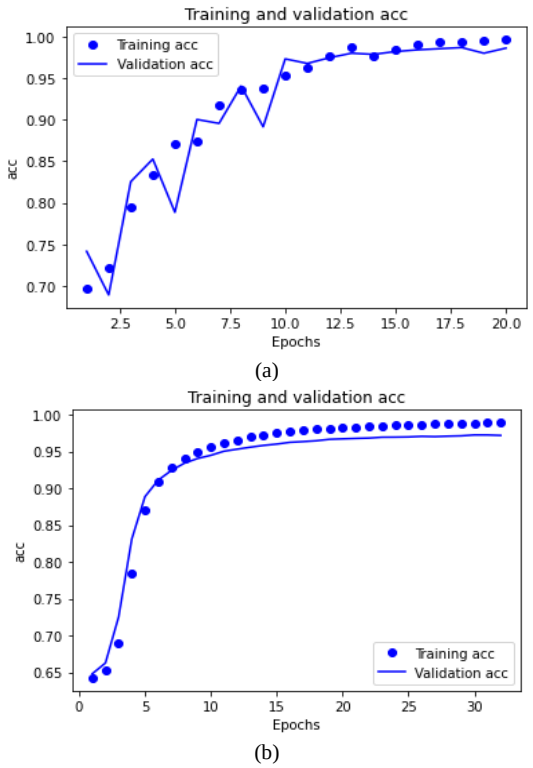

Fig. 4. The training and validation accuracy of the model (a) for DualMyo dataset (b) EMG36 dataset.

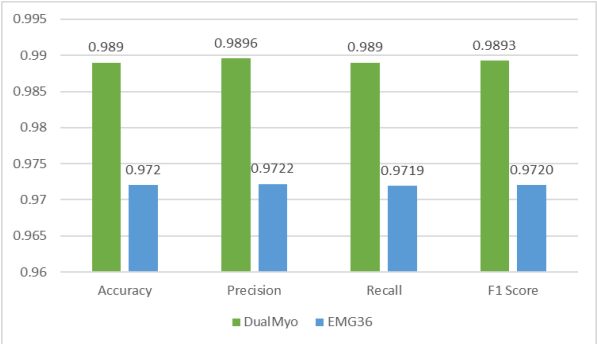

Fig. 5. The performance matrices for the two datasets.

TABLE I. Confusion Matrix for DualMyo Dataset

Output Class	Target Class								
	G0	G1	G2	G3	G4	G5	G6	G7	G8
G0	1123 64.99%	0 0.00%	3 0.17%	1 0.06%	0 0.00%	1 0.06%	1 0.06%	1 0.06%	99.38% 0.62%
G1	1 0.06%	79 4.57%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	1 0.06%	0 0.00%	97.53% 2.47%
G2	2 0.12%	0 0.00%	92 5.32%	0 0.00%	1 0.06%	0 0.00%	0 0.00%	0 0.00%	96.84% 3.16%
G3	1 0.06%	0 0.00%	0 0.00%	89 5.15%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	98.89% 1.11%
G4	0 0.00%	0 0.00%	0 0.00%	0 0.00%	80 4.63%	0 0.00%	0 0.00%	0 0.00%	100.00% 0.00%
G5	1 0.06%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	85 4.92%	0 0.00%	1 0.06%	97.70% 2.30%
G6	2 0.12%	0 0.00%	1 0.06%	0 0.00%	0 0.00%	0 0.00%	72 4.17%	0 0.00%	96.00% 4.00%
G7	1 0.06%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	89 5.15%	98.89% 1.11%
G8	99.29% 0.71%	100.00% 0.00%	95.83% 4.17%	98.89% 1.11%	98.77% 1.23%	98.84% 1.16%	97.30% 2.70%	97.80% 2.20%	98.90% 1.10%

The confusion matrices were analyzed to gain deeper insights into the model's performance. In Table I, which presents the confusion matrix for the DualMyo dataset, we observe that the accuracy of most gestures exceeded 97%. Notably, the gesture labeled as G2 exhibited the lowest accuracy at 95.83%. Similarly, in Table II, which displays the confusion matrix for the EMG36 dataset, most gestures achieved accuracies greater than 96%. Nevertheless, the "hand at rest" gesture (labeled as 2) displayed the lowest accuracy, standing at 93.23%.

TABLE II. CONFUSION MATRIX FOR EMG36 DATASET

Output Class	Target Class								
	0	1	2	3	4	5	6	7	8
0	535572 63.19%	3279 0.39%	1292 0.15%	1190 0.14%	1120 0.13%	1132 0.13%	1284 0.15%	50 0.01%	98.28% 1.72%
1	4512 0.53%	45630 5.38%	7 0.00%	5 0.00%	6 0.00%	12 0.00%	9 0.00%	0 0.00%	90.93% 9.07%
2	1327 0.16%	1 0.00%	46693 5.51%	129 0.02%	65 0.01%	82 0.01%	131 0.02%	9 0.00%	96.40% 3.60%
3	1456 0.17%	8 0.00%	119 0.01%	48059 5.67%	15 0.00%	72 0.01%	233 0.03%	9 0.00%	96.17% 3.83%
4	1652 0.19%	8 0.00%	117 0.01%	13 0.00%	48277 5.70%	203 0.02%	99 0.01%	13 0.00%	95.82% 4.18%
5	1731 0.20%	6 0.00%	82 0.01%	75 0.01%	169 0.02%	48239 5.69%	43 0.01%	5 0.00%	95.81% 4.19%
6	1285 0.15%	9 0.00%	125 0.01%	209 0.02%	106 0.01%	37 0.00%	48754 5.75%	19 0.00%	96.46% 3.54%
7	73 0.01%	2 0.00%	8 0.00%	10 0.00%	4 0.00%	7 0.00%	34 0.00%	2660 0.31%	95.07% 4.93%
8	97.80% 2.20%	93.23% 6.77%	96.39% 3.61%	96.72% 3.28%	97.02% 2.98%	96.90% 3.10%	96.38% 3.62%	96.20% 3.80%	97.20% 2.80%

We noticed that the model takes a significant amount of time during testing, to optimize testing time, we added cascading CNN and max pooling layers to the model. By incorporation of additional CNN layers and the utilization of max pooling, the data entering the LSTM network has been significantly reduced. Consequently, this reduction in data size has contributed to a notable decrease in testing time. CNN has the capability to effectively process substantial quantities of raw data with minimal pre-processing requirements. Additionally, adding more CNN layers to a model can enhance its ability to learn intricate features from data hierarchically. Deeper networks can capture increasingly abstract patterns, making them well-suited for larger datasets. The architecture of the second model is composed of a stack of five CNN layers, each accompanied by max-pooling layers directly afterward. After these CNN layers, an LSTM layer is introduced, followed by a dense layer. The model is finalized with the inclusion of a SoftMax layer. We trained the model on DualMyo, and we found that the testing time for the second model decreased by 1/20 compared to the first model.

Fig. 6 shows the training and validation accuracy of the second model, where the final accuracy of training and validation were 99.78% and 97.68%, respectively. In Fig. 7, we compared the performance matrices for the two models. In terms of accuracy, accuracy for Model 1 and Model 2 is 0.989 and 0.9769 respectively. And for the precision is 0.9896 and 0.978 for two models respectively. And for the recall is 0.989 and 0.9763 for two models respectively. Finally, the F1-score is 0.9893 and 0.9771 for the two models respectively. Where the accuracy, precision, recall and F1-score of the Model 2 slightly decreased compared to the Model 1. Table III displays the confusion matrix for Model 2.

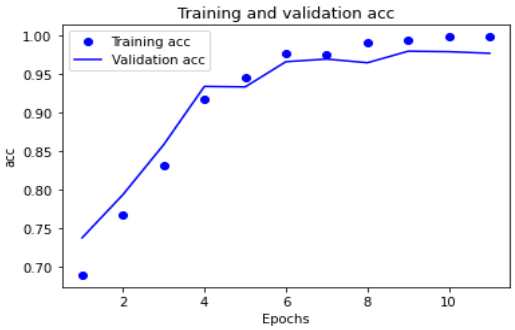

Fig. 6. The training and validation accuracy of the second model for DualMyo dataset.

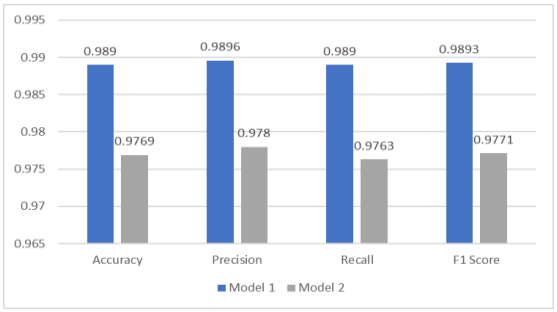

Fig. 7. Comparison of performance matrices for the two models.

We compared our work with other deep learning techniques. Table IV presents the results of this comparison, which includes details such as the technique type, test accuracy, and testing time achieved by each technique.

TABLE III. CONFUSION MATRIX OF MODEL 2 FOR DUALMYO DATASET

Output Class	Target Class								
	G0	G1	G2	G3	G4	G5	G6	G7	G8
G0	1123 64.99%	1 0.06%	1 0.06%	2 0.12%	1 0.06%	0 0.00%	2 0.12%	0 0.00%	99.38% 0.62%
G1	1 0.06%	79 4.57%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	1 0.06%	0 0.00%	97.53% 2.47%
G2	2 0.12%	0 0.00%	89 5.15%	0 0.00%	2 0.12%	0 0.00%	2 0.12%	0 0.00%	93.68% 6.32%
G3	2 0.12%	1 0.06%	0 0.00%	87 5.03%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	96.67% 3.33%
G4	0 0.00%	0 0.00%	2 0.12%	0 0.00%	78 4.51%	0 0.00%	0 0.00%	0 0.00%	97.50% 2.50%
G5	2 0.12%	0 0.00%	0 0.00%	2 0.12%	0 0.00%	83 4.80%	0 0.00%	0 0.00%	95.40% 4.60%
G6	1 0.06%	0 0.00%	6 0.35%	0 0.00%	0 0.00%	0 0.00%	68 3.94%	0 0.00%	90.67% 9.33%
G7	2 0.12%	0 0.00%	1 0.06%	0 0.00%	0 0.00%	6 0.35%	0 0.00%	81 4.69%	90.00% 10.00%
G8	99.12% 0.88%	97.53% 2.47%	89.90% 10.10%	95.60% 4.40%	96.30% 3.70%	93.26% 6.74%	93.15% 6.85%	100.00% 0.00%	97.69% 2.31%

TABLE IV. COMPARISON OF OUR WORK WITH OTHER DEEP LEARNING TECHNIQUES

Work	Technique	Testing accuracy	Testing time
Alejandro *et al.* [9]	LSTM	95%	3.8 s
Shanmuganathan *et al.* [7]	R-CNN	96.48%	–
Jiang *et al.* [19]	Stacked LSTM	97.1%	2.12 s
Pinzón *et al.* [8]	CNN	99%	–
Alejandro *et al.* [10]	LSTM	87.29±6.94%	–
Wang *et al.* [20]	CNN+LSTM + CBAM	92.159%	–
Our	CNN+LSTM	98.9%	6 s
	Cascading CNN+LSTM	97.69%	313 ms

In Ref. [9], both Feedforward Neural Networks (FFNN) and Recurrent Neural Networks (RNN) were employed in their study, resulting in comparable accuracies of approximately 95% for both models. Notably, the RNN model exhibited a shorter testing time, approximately 3.8 s. However, it is essential to recognize that while this reduction represents a significant improvement, it may still not meet the demands of real-time hand gesture recognition applications, where faster response times are often imperative. Shanmuganathan *et al.* [7] achieved a test accuracy rate of 96.48% through the utilization of R-CNN in conjunction with WPT feature extraction. They utilized a 2-channel EMG signal collection approach, which might not fully capture the intricacies of muscle activity responsible for both contraction and relaxation during various hand gestures. Consequently, this limitation could potentially reduce the overall classification performance.

Jiang *et al*. [19] utilized EMG signals and IMUs for gesture recognition, incorporating various models, including LSTM, to boost accuracy, achieving 97.1% accuracy rate. Nonetheless, with a testing time of 2.2 seconds, it may not fully meet the real-time requirements of practical applications. In Ref. [8], a Convolutional Neural Network (CNN) was employed for hand gesture recognition, targeting six specific gestures, and achieving an impressive 99% accuracy rate. However, it's important to acknowledge that the relatively small number of gestures in the dataset may have contributed to the high accuracy observed. Notably, the study did not provide information regarding testing time, which is a critical parameter, particularly in the context of real-time applications. While Ossaba *et al.* [10] work presents promising results in the reduction of EMG channels and improved scalability for embedded systems, it's worth noting that further efforts may be needed to enhance accuracy, particularly in real-time testing, where the model achieved an accuracy of 87%.

Additionally, the absence of information regarding testing time in the study leaves room for future investigations to address this critical parameter, which is vital for evaluating real-time applicability. Le *et al.* [20] presented an innovative approach for enhancing gesture recognition by employing a combination of CNN, LSTM, and Convolutional Block Attention Module (CBAM). While their reported accuracy of 92.159% is commendable and indicative of progress, questions arise regarding its suitability for specific applications within the field, with the accuracy not meeting certain application requirements. The proposed CNN+LSTM model achieved higher accuracy, while the cascading CNN+LSTM model not only notably reduced testing time but also demonstrated good accuracy, making it highly suitable for real-time applications in gesture-controlled systems.

V. CONCLUSION

In this study, a hand gesture recognition system based on EMG signals and deep learning techniques using a CNN+LSTM architecture was proposed. The system was trained and evaluated on two datasets, and our experimental results demonstrated that the proposed system achieved performance, with an average recognition accuracy of 98.9% for the DualMyo dataset and 97.2 for the EMG36 dataset.

To address the challenge of testing time, a second optimized model is presented. By incorporating cascading CNN and max pooling layers, we achieve a remarkable reduction rate of 1/20 in testing time compared to the first model, while maintaining a high level of recognition accuracy. Our comparative analysis revealed that the proposed model outperformed existing methods, including CNN, LSTM, and R-CNN, by achieving the shortest testing time while maintaining a high level of accuracy. This remarkable combination of speed and accuracy makes the model particularly well-suited for real-time hand gesture recognition applications. Its ability to process data quickly allows for near-instantaneous recognition and response, making it an ideal choice for interactive applications where low latency is crucial, such as sign language interpretation or gesture-controlled interfaces.

The experimental results provide strong evidence for the effectiveness of this approach, rendering it suitable for real-time applications in gesture-controlled systems. This paper demonstrates the potential of using EMG signals and deep learning techniques for developing natural and intuitive human-computer interfaces, particularly for individuals with physical disabilities. The proposed system has potential applications in a variety of domains, including prosthetic control, virtual reality, and gaming.

A significant future challenge lies in testing the model with real-time datasets, which will require adapting the model to process incoming data swiftly and deliver real-time recognition and responses. Future work may involve exploring the use of other deep learning architectures, such as attention-based models, to further improve the accuracy and robustness of the proposed system. Additionally, it may be useful to investigate the generalization performance of the system across different user populations, such as individuals with neuromuscular disorders. Expanding the gesture vocabulary to encompass a broader range of commands or intricate sign language gestures holds great potential. Augmenting datasets with diverse examples and exploring synthetic data generation techniques can improve model generalization.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

In the collaborative effort of this research, each author made significant contributions to the work. Wael was responsible for presenting the core research idea and played a pivotal role in the design of the model architecture. Mai constructed the model's architecture, while Ayman meticulously reviewed the model and fine-tuned its parameters to ensure optimal performance. Furthermore, Mai wrote the paper, while Wael and Ayman jointly reviewed and edited the manuscript, all authors had approved the final version.

References

[1] Z. Z. Li *et al*., "Intelligent classification of multi-gesture EMG signals based on LSTM," in *Proc. 2020 International Conference on Artificial Intelligence and Electromechanical Automation*, IEEE, 2020.

[2] M. Mansooreh *et al*., "Transformer-based hand gesture recognition from instantaneous to fused neural decomposition of high-density EMG signals," *Scientific Reports*, vol. 13, no. 1, 2023.

[3] K. D. Hande *et al*., "EMG based hand gesture classification using empirical mode decomposition time-series and deep learning," in *Proc. 2020 Medical Technologies Congress (TIPTEKNO)*, 2020.

[4] T. Panagiotis *et al*., "Deep learning in EMG-based gesture recognition," *Phycs.*, pp. 107–114, 2018.

[5] B. Domenico *et al*., "Deep learning for processing electromyographic signals: A taxonomy-based survey," *Neurocomputing*, vol. 452, pp. 549–565, 2021.

[6] X. Chen *et al*., "Hand gesture recognition based on surface electromyography using convolutional neural network with transfer learning method," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 4, pp. 1292–1304, 2020.

[7] V. Shanmuganathan *et al*., "R-CNN and wavelet feature extraction for hand gesture recognition with EMG signals," *Neural Computing and Applications*, vol. 32, pp. 16723–16736, 2020.

[8] P. A. J. Orlando, R. J. Moreno, and J. E. H. Benavides, "Convolutional neural network for hand gesture recognition using 8 different EMG signals," in *Proc. 2019 XXII Symposium on Image, Signal Processing and Artificial Vision (STSIVA)*, 2019.

[9] S. P. N. Miguel and O. Gibaru, "EMG-based online classification of gestures with recurrent neural networks," *Pattern Recognition Letters*, vol. 128, 20191.

[10] T. O. Alejandro *et al*., "LSTM recurrent neural network for hand gesture recognition using EMG signals," *Applied Sciences*, vol. 12, no. 19, 2022.

[11] O. Tayyip and A. Basturk, "Transfer learning-based convolutional neural networks with heuristic optimization for hand gesture recognition," *Neural Computing and Applications*, vol. 31, 2019.

[12] D. Yash and D. Nath, "Designing a drone controller using electromyography signals," in *Proc. 2021 International Conference on Communication information and Computing Technology (ICCICT)*, 2021.

[13] UC2018 dualmyo hand gesture dataset. [Online]. Available: https://zenodo.org/record/1320922#.YtwUt3ZBzIU

[14] EMG data for gestures. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/EMG+data+for+gestures

[15] L. C. Yann, Y. S. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, 2015.

[16] H. Sepp and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, 1997.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016, vol. 19, pp. 305–307.

[18] E. Ahmet and O. Aytug, "Automatic knee osteoarthritis severity grading using deep neural networks: Comparative analysis of network architectures and optimization functions," in *Proc. International Conference on Applied Engineering and Natural Sciences*, 2023, vol. 1. no. 1.

[19] Y. J. Jiang *et al*., "Multi-category gesture recognition modeling based on sEMG and IMU signals," *Sensors*, vol. 22, no. 15, 5855, 2022.

[20] W. Le *et al*., "Hand gesture recognition using smooth wavelet packet transformation and hybrid CNN based on surface EMG and accelerometer signal," *Biomedical Signal Processing and Control*, vol. 86, 2023.