ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/339832688

Classification of Electromyographic Hand Gesture Signals using Machine Learning Techniques

Article *in* Neurocomputing · March 2020

DOI: 10.1016/j.neucom.2020.03.009

CITATIONS
27

READS
200

4 authors, including:

Guangyu Jia
King's College London
13 PUBLICATIONS **121** CITATIONS
SEE PROFILE

Hak-Keung Lam
King's College London
481 PUBLICATIONS **13,156** CITATIONS
SEE PROFILE

Junkai Liao
King's College London
5 PUBLICATIONS **32** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Open Invited Track on Intelligent Control and Optimization in the framework of 6th IFAC Conference on Intelligent Control and Automation Sciences (ICONS 2022) Cluj-Napoca, Romania, 13-15 July 2022 View project

All content following this page was uploaded by Guangyu Jia on 06 August 2021.

The user has requested enhancement of the downloaded file.

Classification of Electromyographic Hand Gesture Signals using Machine Learning Techniques

Guangyu Jia[a], Hak-Keung Lam[a,*], Junkai Liao[a], Rong Wang[a]

[a]*Department of Engineering, King's College London, Strand, London WC2R 2LS, United Kingdom.*

Abstract

The electromyogram (EMG) signals from an individual's muscles can reflect the biomechanics of human movement. The accurate classification of individual and combined finger movements using surface EMG signals is able to support many applications such as dexterous prosthetic hand control. The existing research of EMG-based hand gesture classification faces the challenges of inaccurate classification, insufficient generalization ability and weak robustness. To address these problems, this paper proposes a deep learning model that combines convolutional auto-encoder and convolutional neural network (CAE+CNN) to classify an EMG dataset consisting of 10 classes of hand gestures. The proposed method shrinks the inputs into a smaller latent space representation using CAE and the resultant compressed features are served as inputs of CNN, which reduces the redundancy of EMG signals and improves the classification accuracy and training efficiency. Besides, to enhance the robustness and generalization ability for classification, a data processing approach is proposed which combines the windowing method and majority voting of the obtained results from the classifier. In addition, comprehensive comparative study is carried out with 8 widely applied and state-of-the-art classifiers in terms of classification accuracy, robustness subject to noise and statistical analysis (sensitivity, specificity, precision, F1 Score and Matthews correlation coefficient). The results demonstrates that the integration of windowing method, CAE+CNN and majority voting achieves the best performance (99.38% test accuracy for the data without adding noise, which is 3.78% higher than the best classifier used for comparison), strongest robustness (achieved 98.13% test accuracy when Gaussian noise of level 1e-5 is added to the raw dataset, which is 4.07% higher than the best classifier used for comparison) and statistical properties compared to other classifiers, which shows the potential for healthcare applications such as movement intention detection and dexterous prostheses control.

Keywords: convolutional auto-encoder, convolutional neural networks, deep learning, EMG signals classification, machine learning

*Corresponding author

Email addresses: guangyu.jia@kcl.ac.uk (Guangyu Jia), hak-keung.lam@kcl.ac.uk (Hak-Keung Lam), junkai.liao@kcl.ac.uk (Junkai Liao), wangrong@wimift.com (Rong Wang)

1. Introduction

The EMG signals from an individual's muscles can reflect the biomechanics of human movement. Many modern prostheses use EMG signals generated by the human muscles to derive control commands for powered upper-limb prostheses [1]. The EMG signals are generally obtained from the surface or implanted electrodes which are used for acquisition of the electric activity generated by human muscles and nerves [2, 3]. There is proportional relevance between the level of the muscle movements and current intensity, thus, EMG signals are always used for detecting the motion intentions of the users and further, controlling the myoelectric prosthetic hands or external peripherals [4].

To accurately control the different finger postures of a prosthetic hand, the discrimination between individual and combined finger movements using surface EMG signals is of great significance [5, 6]. To achieve accurate classification of the EMG signals, in most cases, the implementation procedure consists of three sequential steps [7]: data pre-processing [8], feature extraction [9] and classification [10, 11, 12]. Since EMG signals always contain noise of environment and inherent equipment and the noise caused by the interaction of different tissues and electromagnetic radiation [4, 13], pre-processing is necessary in the beginning to ensure the accurate classification.

In the pre-processing stage, although existed literatures have shown that appropriate features extraction can bring about satisfactory classification performance [14, 15], the shortcomings of feature extraction are also obvious as it requires considerable efforts and experience to select appropriate features [7]. Besides, for many conventional classifiers, the lower dimensional inputs are much more preferable than higher dimensional inputs, which means only small amount of information would be used for classification and considerable information might be lost [7, 16].

The development of deep learning overcomes these shortcomings by automatically learning the sophisticated features from the raw data and taking all the information of the inputs into account during training [17, 18, 19]. The literatures show that machine learning methods especially deep learning approaches recently have been employed in classification of EMG signals [7, 20]. Allard et al. used CNN for the robotic arm guidance and achieved the accuracy of 97.9% in real-time hand gesture classification [21]. Park et al. utilized CNN for movement intention decoding on the kinematic and EMG data and showed applicable above 90% of classification accuracy [22]. Zhai et al. adopted CNNs for neuroprosthesis control on the NinaPro Database (DB) 2&3 and achieved above 80% classification accuracy [23].

However, the existed study on EMG-based hand gesture classification also have some limitations and challenges. At first, the trained models tend to be subject-specific and are not easy to be generalized to other datasets, since different subjects have different data characteristics such as contractions' strength, skin thickness and muscular mass [24]. Also, EMG signals are prone to contaminations caused by real-world factors such as electrode shift and muscle fatigue, small changes of the EMG traces have great impacts on classification results [25, 24]. To compensate the weak robustness of the established model, more data are employed for training in some literatures [25, 24], which further increases the computational costs.

In order to reach high classification performance and try to overcome the limitations in terms of high computational costs, weak generalization ability and robustness of the existed methods, we proposed the following framework to classify EMG signals.

Inspired by manually extracting important features of the raw data before training, we are looking for an automatic representation learning and dimensionality reduction algorithms of the inputs without compromising the classification accuracy. Convolutional auto-encoder (CAE), a semi-supervised learning algorithm which aims to reduce the dimensions of the input into a smaller space representing the most important information of the original data [26, 27], is adopted in this paper. More specifically, CAE is employed to compress the inputs, then the compressed result, i.e., the encoded features, serves as the starting point of a CNN. For simplicity, we denote this structure by CAE+CNN. Through the utilization of CAE, the redundancy of raw EMG signals and the computation load of classification can be largely reduced.

To increase the generalization ability and robustness of the model, we employed the data from all subjects to avoid subject-specific problem and proposed a method integrating windowing approach and majority voting. Windowing method uses sliding windows within signals to generate more samples [28] and shuffle the samples before training, which mainly aims to further improve the classification performance of the model. At the results collecting stage, majority voting is adopted to offer the final decision of each sample based on the outputs of the classifier, which aims to rectify the wrong classification results of windowing samples and improve the robustness of the proposed model.

The test results, robustness and statistical analysis in this paper demonstrate the feasibility of the CAE+CNN scheme. It reaches an average test accuracy of 99.38%, which is the highest accuracy compared to other classic machine learning methods used in this paper. For comparison purpose, the employed classifiers including CNNs, neural network (NN), K-nearest neighbor (KNN), random forest, decision tree, support vector machine (SVM), logistic regression and naïve Bayes method.

Also, feature extraction method is used for evaluating its impacts on classification process. The classifiers are fed by the data processed by feature extraction and without feature extraction. The results show that feature extraction can boost the performance of some conventional classifiers but has little improvement of deep learning models' performance. Similarly, the impacts of windowing method are also evaluated. All classifiers in this paper are fed respectively by the windowing data and the original data (without windowing), the results show that the integration of windowing and majority voting largely improves the classification performance of CAE+CNN, CNN, NN, random forest, decision tree and SVM.

The rest of paper is organized as following: Section II formulates the preliminaries of the classifiers utilized in this paper. Section III illustrates the data and the classification methods. The test results, the statistical analysis as well as the robustness property are demonstrated in Section IV. Section V makes a conclusion.

2. Preliminaries

In this Section, the preliminaries about the classifiers used in this paper are presented.

2.1. Convolutional Neural Network

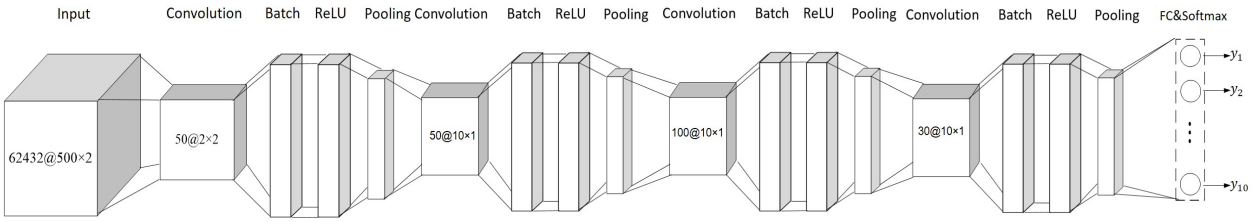


Figure 1: CNN architecture

CNNs have advantages in processing the high-dimensional data, such as images and time-series data due to (a) its convolutional setting in which the hidden units are not fully connected to the input but instead divided into locally connected segments; (b) pooling methods which reduce the dimensionality of the feature space and achieve invariance to small local distortions [29]. One example of CNN used in this paper is shown in Fig. 1.

Mathematically, the forward propagation in the convolutional layer can be represented as follows [30]:

$$x_j^\ell = f(\sum_i x_i^{\ell-1} * k_{ij}^\ell + b_j^\ell), \tag{1}$$

where x_j^ℓ represents the j-th output feature map of the l-th layer, $f(\cdot)$ is the output activation function, k_{ij}^ℓ is the filter applied to the i-th output feature map in the $(\ell-1)$-th layer and producing the j-th feature map in the ℓ-th layer x_j^ℓ, b_j^ℓ is the bias of the j-th feature map in layer l, $*$ denotes the convolution operation. One of the most important elements of the backpropogation procedure is to compute the sensitivities for a given feature map, based on which the gradients for the biases and kernel weights can be obtained. Similarly, assuming that E denotes the error function between the actual output and the desired output, then the following set of equations can be written down [30]:

$$\frac{\partial E}{\partial k_{ij}^\ell} = \sum_{u,v} (\delta_j^\ell)_{uv} (p_i^{\ell-1})_{uv} = \widetilde{\delta}_j^\ell * x_i^{\ell-1}, \tag{2}$$

$$\frac{\partial E}{\partial b_j^\ell} = \sum_{u,v} (\delta_j^\ell)_{uv}, \tag{3}$$

$$\delta_j^\ell = f'(u_j^\ell) \circ (\delta_j^{\ell+1} * \widetilde{k}_{ij}^\ell), \tag{4}$$

where $\widetilde{\delta}_j^\ell$ and $\widetilde{k}_{ij}^\ell$ are obtained by rotating δ_j^l and k_{ij}^ℓ by 180 degrees, '$\circ$' denotes element-wise multiplication, $(p_i^{\ell-1})_{uv}$ is the patch in $x_i^{\ell-1}$ that was multiplied element-wise by k_{ij}^ℓ during convolution so as to compute the element at (u, v) in the output convolution map x_j^ℓ [30], p_i^{l-1} is the corresponding image block in the input feature maps, $f'(\cdot)$ is the derivative of $f(\cdot)$.

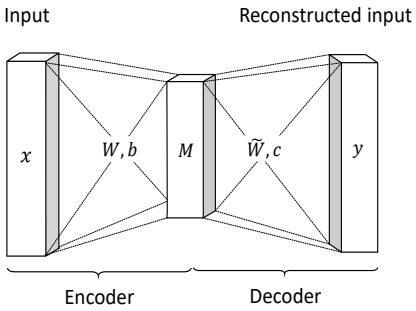


Figure 2: The structure of a CAE.

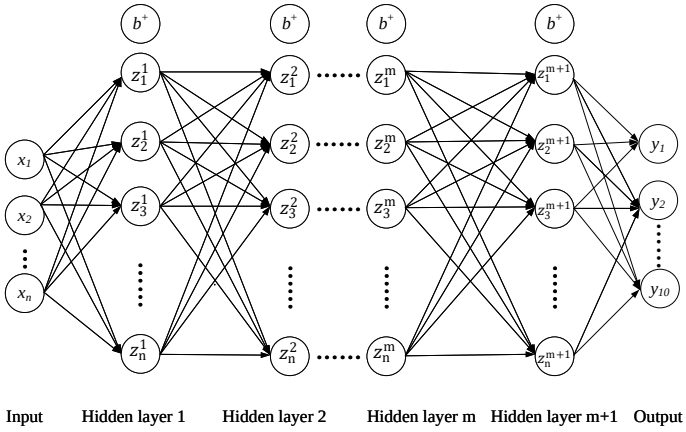


Figure 3: The structure of NN.

2.2. Convolutional Auto-Encoder

Convolutional auto-encoder, a type of hierarchical semi-supervised feature extractor, is utilized here to automatically learn non-trivial features from the inputs and scale the high-dimensional inputs [31]. Like what shows in Fig. 2, the former part of the network which reduces the dimensions of the inputs is an encoder while the second part that reconstructed the inputs to the full size from the encoder features is called the decoder. For a CAE, assume that the input is x, the latent representation of the k-th feature map (M^k) is given by [31] as follows:

$$M^k = f(x * W^k + b^k), \tag{5}$$

where $W^k \in W$ denotes the weights learned by training, $*$ denotes the 2D convolution, b^k is the bias broadcasted to the whole k-th feature map, $f(\cdot)$ is an activation function (i.e., ReLU in this paper). Accordingly, the reconstruction is given by [31] as follows:

$$y = f(\sum_k M^k * \widetilde{W}^k + c^k), \tag{6}$$

where c^k is the bias of the k-th feature map. $\widetilde{W}$ is obtained through the flip operation over both dimensions of the weights W. The training of CAE is to achieve:

$$J(W, b, c) = \min_{W,b,c} \frac{1}{2} \sum_{n=1}^{N} ||y^{(n)} - x^{(n)}||^2, \tag{7}$$

where N is the number of training samples. The backpropagation procedures of CAE are the same as those of CNNs. Deep hierarchy of CAE can be obtained by several CAE stacked, namely, feeding the hidden layer feature maps of current stack as the input to the next stack to form CAE stacks.

2.3. Other Machine Learning Models

To compare the classification performance of the above models, other machine learning models in the literature including NN, random forest [32], decision tree [33], KNN [34].

Particularly, when training NN, the optimization algorithms used in this paper including Levenberg-Marquardt (LM) method, conjugate gradient backpropagation (cgb) with Powell-Beale restarts and adaptive moment estimation (Adam), in which LM algorithm achieved the best performance in most cases. The update rule of LM algorithm is given below [35].

$$W_{k+1} = W_k - (J_k^t J_k + \lambda I)^{-1} J_k e_k, \tag{8}$$

where k is the index of iterations, W_k represents the weight to be updated, J_k denotes the Jacobian matrix, λ is the combination coefficient, I is the identity matrix and e_k denotes the error vector containing the output errors for each input vector [35]. An example of NN structure used in this paper is shown in Fig. 3, where there are $m+1$ hidden layers with n nodes in each layer.

3. Methods

In this paper, machine learning techniques are employed to classify 10 hand gestures. We employ a CAE to shrink the inputs into a smaller latent space representation, then the compressed features serve as the start point of a CNN. For comparative purpose, 8 conventional classifiers mentioned above (CNN, NN, random forest, decision tree, KNN, naive Bayes, SVM, logistic regression) are employed.

In the implementation stage, data normalization is carried out at first. To observe the impacts of feature extraction, the classifiers adopted in this paper are fed by the data with and without feature extraction. Also, windowing method is utilized in data processing stage. The new samples obtained through windowing method will get the same labels as the original samples, and majority voting is adopted after training to provide the final decision of original samples. Similarly, to compare the effects of windowing method, the classifiers are fed respectively by the data processed with and without windowing scheme.

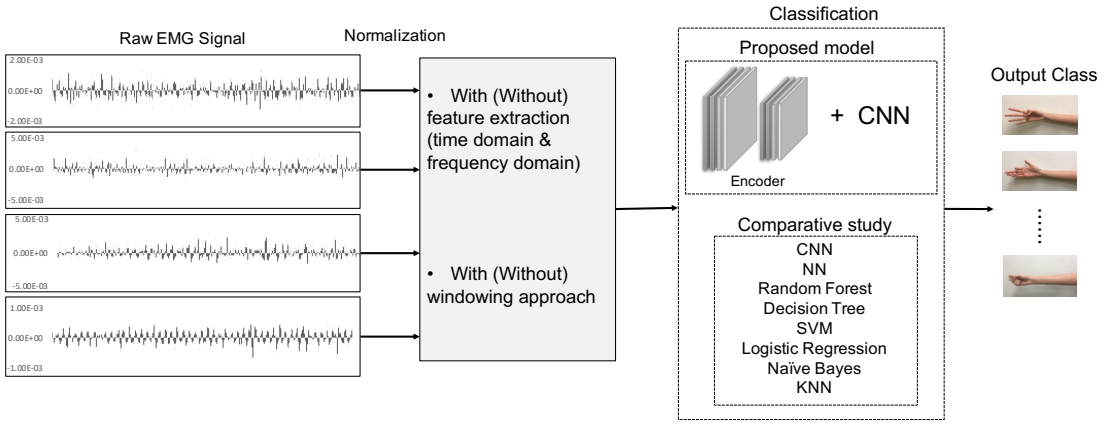


Figure 4: Block diagram of the implementation procedure in this paper.

Fig. 4 shows the overall implementation procedure of this paper. As shown in Fig. 4, the raw EMG signals are processed using normalization, feature extraction and windowing methods, which will be elaborated in Section 3.1, 3.2 and 3.3 respectively. To compare the

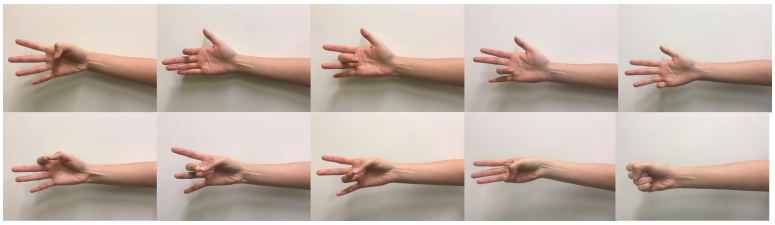


Figure 5: Ten classes of finger movements. The sub-figures from left to right correspond to the single finger movements (Thumb (T), Index (I), Middle (M), Ring (R) and Little (L) in this first line) and combined finger movements (Thumb–Index (TI), Thumb–Middle (TM), Thumb–Ring (TR), Thumb–Little (TL) and hand close (HC) in this second line), respectively.

impacts of windowing and feature extraction methods, the original data are processed into 4 categories for classification (see Fig. 7), which will be explained in Section 3.4. At the classification stage illustrated in Section 3.5, we use CAE to reduce the dimensions of inputs and CNN as the classifier, besides, other classifiers' results are used for comparative study.

3.1. Data Description and Pre-processing

Table 1: Data Information

Sampling rate	Sampling time	Number of electrodes	Sample's size	Number of subjects	Number of samples
4000 Hz	5 seconds	2	20000×2	8	480 (training: 320, test: 160)

The raw data of the surface EMG signals are from the the experiment in [5], where 8 subjects aged between 20 and 35 years old without neurological or muscle disorders were recruited to execute the required order of the finger movements. The summary of data information is shown in Table 1. This dataset from [5] was collected using two EMG channels (Delsys DE 2.x series EMG sensors) and processed by the Bagnoli Desktop EMG Systems from Delsys Inc. Besides, a 12-bit analog-to-digital converter (National Instruments, BNC-2090) was applied with a sampling rate of 4000 Hz. The subjects were instructed to relax and hold the hand gestures for a period of 5 seconds [5]. Thus, for each sample performed by each subject, the data size is 20000×2, where 2 corresponds to 2 channels obtained from 2 electrodes, as the sampling time is 5 seconds, the length of the data is 20000. Eight subjects repeated each required finger movement six times, and in total ten classes were performed which included the flexion of each individual fingers. Ten classes of finger(s) movements are shown in Fig. 5 which includes Thumb (T), Index (I), Middle (M), Ring (R), Little (L) and the pinching of combined Thumb-index (TI), Thumb-Middle (TM), Thumb-Ring (TR), Thumb-Little (TL), and finally the hand close (HC).

This dataset [5] has the advantage that it detected single and combined finger movements with just two electrodes, which leads to less intrusion, lower computation load and more dexterous prosthesis design in practical. Plenty of other related datasets either used more than two EMG electrodes for data collection [36, 37, 38, 39] or did not consider combined finger movements [40, 41].

The sampling method used in the machine learning model is the stratified sampling which is implemented by dividing the dataset into subgroups and selecting the samples from each group. As there are 10 classes of hand gestures, to make the inputs contain the balanced amount of samples from each class, we randomly select training, validation and test samples from each class and combine them to form the final datasets. Thus the classifier is fed by the data consisting of the equal number of samples of each class.

It is inevitable that the raw data may contain noise, outliers or have large difference between subjects. Normalization was employed to rescale the data so as to de-noise and eliminate the large difference of signal amplitude between different subjects. In implementation, the input signals of each subject were rescaled to the range [0, 1] using the following equation:

$$y_N = \frac{y - y_{\min}}{y_{\max} - y_{\min}}, \tag{9}$$

where y is the signal to be normalized, y_N is the normalized signal, $y_{\min}$ and $y_{\max}$ denote the minimum and maximum elements in y, respectively.

3.2. Feature Extraction

Feature extraction is a useful technique to extract important information from the raw data and reduce the redundancy. The data size of each sample in this paper is 20000×2, which is considerably large for some basic classfiers like NNs and KNNs in the process of training and test, and requires large memory during computation. Extracting informative and non-redundant feature is a common method to solve these problems. In this paper, after trying more than 30 time-domain and frequency-domain features as well as their combinations [42, 43, 44], the two features which achieve the best performance were selected, i.e., in time domain, the sum of absolute value (SAV) of each channel:

$$\mathrm{SAV} = \Sigma_{i=1}^{L}|e_i|, \tag{10}$$

where e_i represents the EMG signal in a segment i, L denotes length of the whole signal and $|\cdot|$ represents the absolute value of the signal. The second feature is obtained in frequency domain, that is, the standard deviation (STD) of each channel:

$$\begin{aligned} \mathrm{STD} &= \sqrt{\frac{1}{L-1}\sum_{i=1}^{L}(|F(e_i)| - \mu)^2)}, \\ \mu &= \frac{1}{L}\sum_{i=1}^{L}|F(e_i)|, \end{aligned} \tag{11}$$

where $F(\cdot)$ denotes the Fourier transform of the signal.

3.3. Windowing Scheme & Majority Voting

Windowing (cropping) scheme is also an important preprocessing method which can help classifiers achieve better performance. As shown in Fig. 6, it is implemented by sliding a

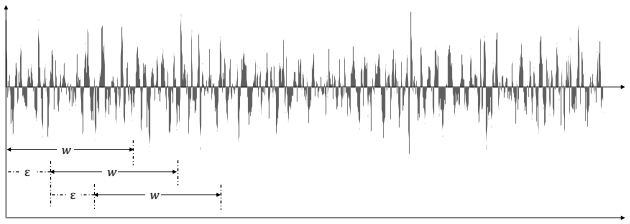


Figure 6: Windowing scheme with window size w and increment ϵ.

window of the size w with increment ϵ over the original sample to truncate the sample before and after the window while not modifying the contents within the window [45]. Through this operation, subsets of original samples can be obtained. Denote the number of newly generated samples by N, original data length by L, window size by W and the window increment by I, the number of generated samples can be computed as:

$$N = (L - W)/I + 1.$$

The windowing scheme is chosen when pre-processing for the following reasons: (I) Increasing the number of training samples: The total number of samples is 480 which is insufficient for deep learning training. The windowing scheme used in this paper largely increases the number of training samples. (II) Providing more flexible control of inputs' length and contents: Since each sample has the size of 20000×2 which contains redundant information and is relatively large to feed into the classifiers, the windowing scheme is used to crop the original data with the specified length. Also, as the research in [46] pointed out that the disjoint windowing approach may generate the inputs having incomplete information of the overall muscle activity, the overlapped windowing is adopted, which is used to determine how much shifted and overlapped information is included in the inputs. (III) Effectively improving the classification accuracy: After using the windowing scheme, the shifted neighbouring signals contain similar but oscillatory and translational information which helps improve accuracy and avert over-fitting problems.

The length of sliding window here is set to 500 with increment 10, therefore, windowing scheme increases our training-set size by a factor of 1951. The newly generated training data (though highly correlated) will get the same label as the original sample. In the scenarios that data are insufficient for training (we just have 480 samples in this paper), this method provides the large number of input data and can force the CNN to learn features from each windowing data rather than the complete sample, which has proved to be effective for deep learning training [28].

3.4. Four Types of Inputs Derived from Feature Extraction and Windowing Scheme

At the implementation stage, the data is normalized into the range [0, 1]. In order to compare the effectiveness of feature extraction and windowing approach for various classifiers, the normalized data were processed further into four categories. Fig. 7 shows these four

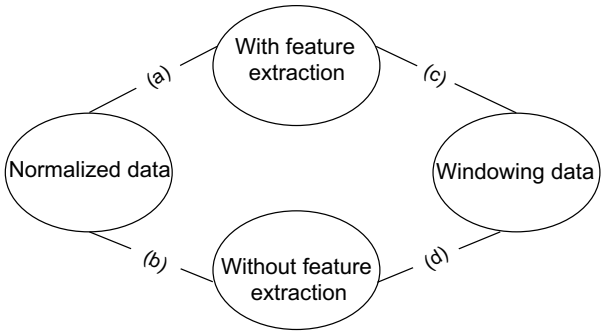


Figure 7: Four types of inputs used for comparing the impacts of feature extraction and windowing methods on classification performance (noting that windowing data are obtained by using a sliding window within the normalized data): (a) normalised data with feature extraction; (b) normalised data without feature extraction; (c) windowing (normalised) data with feature extraction; (d) windowing (normalised) data without feature extraction.

scenarios: (a) normalized data with feature extraction, i.e., extracting two features mentioned in Section 3.2 from the normalized data; (b) normalized data without feature extraction, i.e., just use the normalized data as inputs; (c) windowing data with feature extraction, i.e., as mentioned in Section 3.3, a window of the size 500×2 with the increment of the size 10 is sliding over the normalized data and creates the overlapped 'Windowing data', then feature extraction illustrated in Section 3.2 is implemented on the windowing data; (d) windowing data without feature extraction, i.e., just use the windowing data as inputs without any feature extraction operation.

3.5. Classification

In the classification phase, machine learning techniques including CAE+CNN, CNN, NN, KNN, random forest, decision tree, SVM, logistic regression and naïve Bayes classifier are employed to classify the EMG signals. The proposed model in this paper is the combination of CAE and CNN, which is shown in Fig. 8. The CAE is employed to compress the inputs into a smaller dimensions and automatically obtain the features to be classified. The structure of CAE used in this paper is shown in Fig. 9, each CAE stack contains a convolutional layer, a batch normalization layer, a ReLU layer and a max pooling layer. The CAE is trained at first, then the encoder part is employed to obtain the compressed inputs for CNN. The optimization algorithm of this structure is Adam and the loss is evaluated by MSE. As shown in Fig. 8, inputs size is reduced by CAE from 500×2 to 125×2, which aims to reduce the redundancy of the data and lower the computational load for the following classifier CNN.

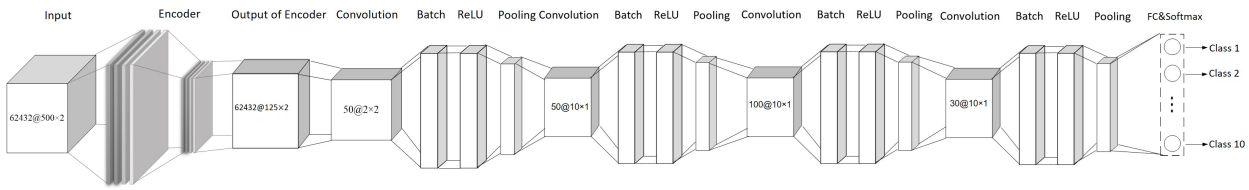


Figure 8: CAE+CNN's architecture.

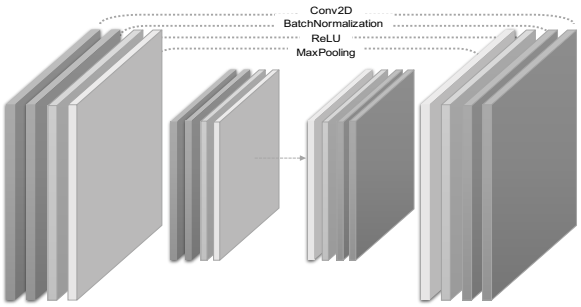


Figure 9: The structure of the CAE used in this paper.

Noting that NN used in this paper has 3 hidden layers with 21 nodes each layer, the optimization algorithms include LM, cgb and Adam corresponding to different inputs cases which will be mentioned in the following section. The detailed specifications of other comparative models refer to Table 2.

Table 2: Other Machine Learning Models' Specifications*.

KNN	Random Forest	Decision Tree	SVM	Logistic Regression	Naive Bayes
K neighbor:5	Bootstrap:True	Splitter:Best	Kenel:rbf	Fit intercept:True	Var smoothing:10^{-9}
Leaf size:30	Criterion:Gini	Criterion:Gini	Penalty**:1.0	Cross-validation:Stratified K-Folds	—
Metric:Minkowski	Class weight:1.0	Class weight:1.0	Coefficient:0.0	Penalty***:L2	—

* This Table shows part of the parameters, other specifications of each model are the default values of the above models in scikit-learn library.
** Penalty parameter.
*** The norm used in the penalization.

To offer fair comparison of the performance of various classifiers and test the effectiveness of different data processing methods, we applied the above classifiers to each type of the inputs in Fig. 7, and the robustness test was also implemented for each case. Four types of inputs shown in Fig. 7 are fed to the classifiers successively, the training details and results of each case will be illustrated in the following sections.

Noting that for cases (c) and (d), i.e., when using the windowing scheme, each original sample will produce N new samples (refer to Section 3.3). To determine the classification output of each original sample, majority voting is employed. That is, the final classification result of an original sample is determined by the voting of the corresponding N cropped samples' predictive results.

4. Experiment Configuration

This section will introduce the experiment procedure, platforms, classifiers' configurations and the methodology for calculating the final accuracy.

The experiments are implemented using Matlab R2018b, Tensorflow 1.12.0 and sped up

by Nvidia GPUs. Matlab is for training NN with LM and cgb algorithm, other classifiers are trained using Tensorflow.

4.1. Data splitting

In the data splitting phase, four repeated trials were used for training and the remaining two were for test. Thus, 320 samples in which each subject contributes 40 samples (10 gestures $\times$ 4 repetitions) are used as training data, while 160 samples in which each subject contributes 20 samples (10 gestures $\times$ 2 repetitions) are adopted as test data.

4.2. Experiment Procedures

As shown in Fig. 4, the general experiment procedures used in this paper consist of data pre-processing, feature extraction/windowing scheme, classification and results analysis. The raw EMG signals are processed into four categories, which is shown in Fig. 7. The experiment details of each category are elaborated as follows.

4.2.1. Case (a)

In this case, the data are first normalized, then the feature extraction is used to derive two features: SAV in time domain defined in (10) and STD in frequency domain defined in (11). As mentioned in Section 4.1, 320 samples from all subjects are used for training while 160 samples from all subjects are used for test, and each samples is of the size 20000×2. After feature extraction, 2 features from each channel were extracted, thus, the size of the training input and the test input is 320×4 and 160×4, respectively.

In the classification stage, all classifiers mentioned in Section 2.3 are adopted for comparison. Based on the data size and the experience coming from trail and error, the optimization algorithm for NN is set to LM and Adam is used for the training of CAE+CNN and CNN. For the proposed CAE+CNN model, 250 epochs with batch size 20 is used. For the compared CNN model, 100 epochs with batch size 15 is adopted. The training and test accuracy are computed through:

$$\text{Accuracy} = \frac{\text{Number of correctly classified samples}}{\text{Total number of samples}} \tag{12}$$

4.2.2. Case (b)

As no feature extraction is implemented in this case, the training dataset have the size of $320 \times 20000 \times 2$ while the test dataset have the size of $160 \times 20000 \times 2$. For NN, the adopted optimization algorithm is Adam. It is noticed that the algorithm LM is supposed to perform better in this case, but LM's training requires 5271.7G RAM due to the large input size and the Jacobi method, which is not feasible to implement. Thus in this case, Adam, rather than LM was chosen for NN. The optimization algorithm for the model CAE+CNN and CNN is also Adam. When training the proposed CAE+CNN model, 50 epochs with batch size 40 is used. For the CNN model, 50 epochs with batch size 20 is adopted in this case. The training and test accuracy are computed by (12).

4.2.3. Case (c)

In light of the training needs for deep learning techniques, we adopted overlapped windowing method to crop the data, which could largely increase the number of input samples. Using the formula $N = (L - W)/I + 1$ where $L = 20000$, $W = 500$ and $I = 10$, each sample generates 1951 new samples which share the same label. The values of W and I are chosen for the reason that after plenty of trials, we found the combination of $W = 500, I = 10$ provides the best performance. Accordingly, 320 training samples and 160 test samples will respectively generate 624320 and 312160 cropped samples and each sample has the size of 2×2 after feature extraction.

However, using all windowing samples may lead to large computational cost and overfitting problem. In order to reduce the computational cost without compromising classification accuracy, we attempted to reduce the number of data to 70%, 50%, 30% and 10% of the shuffled windowing inputs and found that using 10% randomly selected training samples can surprisingly obtain almost the same performance with using 100% windowing data. When training the proposed CAE+CNN model, 50 epochs with batch size 256 is used while for the CNN model, 30 epochs with batch size 256 is adopted. In this case, the windowing method is utilized to crop the data, thus, the prediction result of each sample is obtained by the majority voting method (introduced in Section 3.3). Also, the final accuracy is computed through (12).

Noting that the training samples are randomly selected from 10% cropped data of original samples, thus, the inconsistency of adjoining windowing samples makes it impossible to apply majority voting method to determine the predictive results of training samples. So here the training accuracy of this case refers to the proportion of correctly classified windowing samples rather than the original samples. Although the test dataset was also processed by windowing method, the consistency of test data was kept as no shuffling and selection of windowing samples have been done. Therefore, the test accuracy is considered in two aspects: the test accuracy before voting (i.e., the proportion of correctly classified windowing samples) and the test accuracy after voting (i.e., the proportion of correctly classified original samples, which is same with cases (a) and (b)). We choose the latter accuracy definition to calculate test accuracy in this case as it is computed based on the classification results of original samples.

4.2.4. Case (d)

Similar to case (c), 10% of the windowing samples are randomly selected for training. Since no feature extraction is used in this case, each sample's size is 500×2 ($W = 500, I = 10$). When training the proposed CAE+CNN model, 20 epochs with batch size 256 are used. For the CNN model, 20 epochs with batch size 128 are adopted. The number of test samples is still 312160, majority voting is used to obtain the classification results, the test accuracy after voting is used to compute the final test accuracy.

5. Results and Discussion

In this Section, the classification results of cases (a) to (d) are presented and compared. Robustness property and statistical analysis are also included to evaluate the performance of the proposed and comparative classifiers.

Table 3: Training and test accuracy of different classifiers on 4 types of inputs. The values in bracket in case (c) and (d) are the test accuracy before majority voting, whilst the values right above them represent the test accuracy after majority voting.

Classifier / Cases	CAE+CNN		CNN		Random forest		Decision tree		SVM	
	training	test	training	test	training	test	training	test	training	test
Case (a)	30.63%	26.25%	21.25%	21.25%	99.68%	91.88%	100.00%	88.75%	99.38%	72.50%
Case (b)	83.44%	30.05%	90.32%	21.15%	99.06%	26.87%	100.00%	19.38%	26.25%	18.13%
Case (c)	100.00%	87.50%	77.14%	81.25%	99.25%	93.75%	100.00%	95.00%	79.58%	85.00%
		(82.07%)		(74.56%)		(84.20%)		(81.70%)		(77.85%)
Case (d)	98.74%	99.38%	94.95%	95.00%	99.58%	78.13%	100.00%	84.00%	15.52%	16.25%
		(91.90%)		(83.25%)		(61.27%)		(45.91%)		(15.35%)

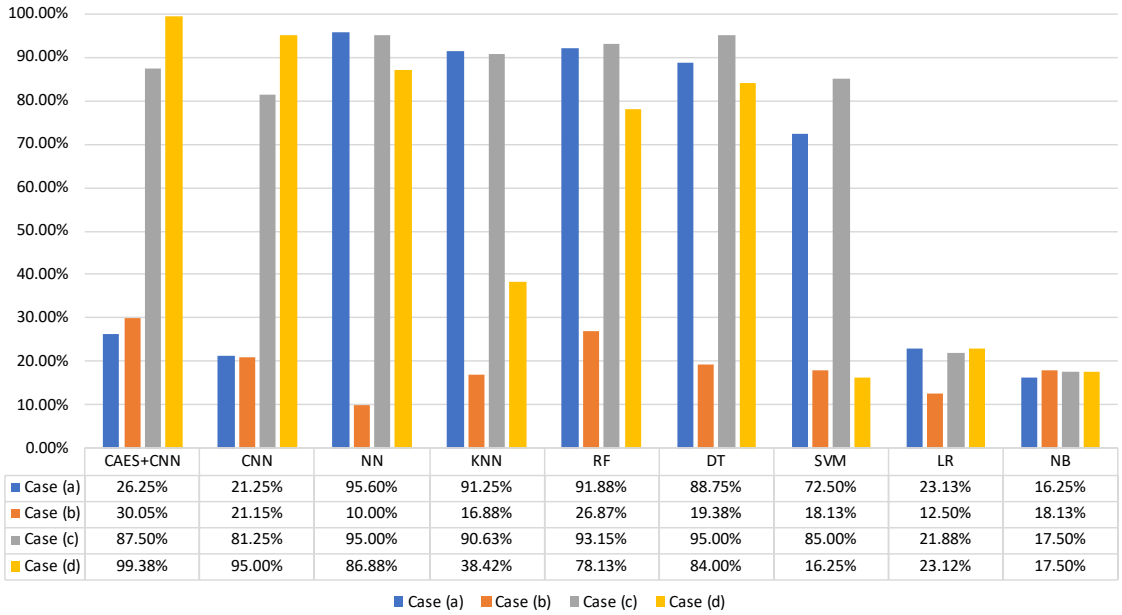
Classifier / Cases	KNN		Logistic regression		Naive Bayes		NN	
	training	test	training	test	training	test	training	test
Case (a)	97.19%	91.25%	20.31%	23.13%	16.25%	16.25%	100.00%	95.60%
Case (b)	24.06%	16.88%	100.00%	12.50%	30.31%	18.13%	10.00%	10.00%
Case (c)	84.52%	90.63%	20.77%	21.88%	18.41%	17.72%	88.45%	96.25%
		(76.25%)		(17.92%)		(17.50%)		(86.24%)
Case (d)	46.75%	38.42%	20.77%	21.88%	18.41%	17.72%	82.89%	86.88%
		(36.57%)		(17.92%)		(17.50%)		(81.27%)

Table 3 shows the training and test accuracy of the proposed CAE+CNN model as well as the comparative models. Noting that in cases (c) and (d), windowing method and majority voting are adopted, thus the accuracy before and after majority voting are presented in these two cases (the values in bracket represent the accuracy before majority voting). In order to render more intuitive and clear comparison of the results of the 4 cases, the overall picture of test accuracy (after majority voting) from cases (a) to (d) is shown in Fig. 10.

5.1. Results of cases (a) to (d)

5.1.1. Case (a)

From Table 3 and Fig. 10, it can be seen that NN with LM training algorithm performs the best among the classifiers. Random forest and KNN also achieve above 90% test accuracy. Deep learning models, CAE+CNN and CNN, have poor performance in this type of dataset, which is mainly due to small amount of training samples.



	CAES+CNN	CNN	NN	KNN	RF	DT	SVM	LR	NB
Case (a)	26.25%	21.25%	95.60%	91.25%	91.88%	88.75%	72.50%	23.13%	16.25%
Case (b)	30.05%	21.15%	10.00%	16.88%	26.87%	19.38%	18.13%	12.50%	18.13%
Case (c)	87.50%	81.25%	95.00%	90.63%	93.15%	95.00%	85.00%	21.88%	17.50%
Case (d)	99.38%	95.00%	86.88%	38.42%	78.13%	84.00%	16.25%	23.12%	17.50%

Figure 10: Test accuracy of various classifiers corresponding to inputs types (a) to (d).

5.1.2. Case (b)

Inputs type (b) shows the worst performance in four cases, most classifiers have the lowest test accuracy compared to other 3 cases (Table 3 and Fig. 10). From the comparison of case (a) and case (b), it can be concluded that feature extraction method indeed improves the classification performance of the models NN, KNN, random forest, decision tree, SVM, but fail to improve the accuracy of CAE+CNN, CNN, logistic regression and naive Bayes classifier. Although in this case each sample's size is much larger than that of inputs type (a), the quantity of training samples is still considerably small, which hinders the effectiveness of feature learning of CAE+CNN and CNN.

5.1.3. Case (c)

For the inputs type (c), the windowing method, majority voting and feature extraction are all employed. It can be seen from Table 3 and Fig. 10 that after using feature extraction, windowing method and majority voting, the classification performance of CAE+CNN, CNN is lessened (compared to case (d)) while the performance of basic machine learning classifiers like KNN, RF, DT and SVM is improved compared with that of inputs type (d), i.e., the windowing data without feature extraction.

5.1.4. Case (d)

As mentioned above, in case (d), the training data size is $62432 \times 500 \times 2$. The number of test samples are still 312160, majority voting is used to obtain the prediction results and (12) is used to compute the final accuracy. As shown in Table 3 and Fig. 10, CAE+CNN achieves the best performance of 99.38% as test accuracy. CNN also has satisfactory performance in this case, which indicates that larger number of inputs are preferable for training deep

Table 4: Robustness result: test accuracy* of different classifiers** on inputs type (a), i.e., raw data with feature extraction.

Noise Level \ Classifier	CAE+CNN	CNN	NN***	KNN	RF	DT	SVM	LR	NB
1e-5	23.75%	20.00%	78.13%	74.38%	73.13%	70.00%	19.38%	23.13%	16.25%
2e-5	22.50%	17.50%	58.13%	58.75%	62.50%	60.00%	13.75%	22.50%	15.00%
1e-4	22.50%	18.75%	29.38%	25.63%	25.00%	23.13%	12.50%	20.63%	16.25%

* The test accuracy is the average value of 10 trials.
** RF, DT, LR and NB denote random forest, decision tree, logistic regression and naive Bayes, respectively.
*** Optimization algorithm in training NN: LM.

learning structures whilst small dimensional inputs are more appropriate for classic classifiers. Besides, through observing the test accuracy before voting (values in bracket of Table 3) and after voting, it can be concluded that majority voting of windowing samples increases the classification performance.

5.2. Robustness Analysis

Robustness analysis aims to evaluate the vulnerability of models by feeding them the data with noise of different levels. Specifically, in order to test the robustness level of trained classifiers, we add the different levels of Gaussian noise to the raw test data, then use the normalization, feature extraction and windowing approach accordingly to produce data with noise for each inputs case. Then, feed the perturbed data to the trained classifiers and check the classification accuracy of the classifiers.

Assume that the inputs are represented by a matrix X whose dimensions are $s_1 \times s_2$. After adding additive form of noise, the inputs becomes

$$X' = X + m \cdot \text{randn}(s_1, s_2),$$

where X' is the contaminated inputs, m represents the noise levels which are positive values, randn() represents the function of Gaussian noise and '$\cdot$' denotes the element-wise multiplication, the size of $\text{randn}(s_1, s_2)$ is the same with X'. Given that the raw inputs' order of magnitude is 10^{-5}, the noise levels are accordingly set to $m = 1 \times 10^{-5}, 2 \times 10^{-5}, 1 \times 10^{-4}$.

Since the performance of all classifiers in case (b) are below 50% which is not acceptable in practical applications, its robustness test is omitted.

Table 4, Table 5 and Table 6 show the robustness results of cases (a), (c) and (d). For case (a), the results in Table 4 show that NN has the best robustness, the KNN and random forest follow. NN has the strongest robustness in this case mainly due to the use of LM algorithm which is proven to be a more robust method [35]. For cases (c) and (d), the robustness results are shown in Table 5 and Table 6. It can be concluded that windowing scheme largely improves the robustness of almost all classifiers, which is mainly benefited from the increased number of training samples. CAE+CNN offers the best robustness results

Table 5: Robustness result: test accuracy* of different classifiers** on inputs type (c), i.e., windowing data with feature extraction.

Noise Level \ Classifier	CAE+CNN	CNN	NN***	KNN	RF	DT	SVM	LR	NB
1e-5	85.00%	75.62%	85.00%	71.88%	85.00%	85.00%	73.12%	25.62%	16.25%
2e-5	76.25%	68.13%	65.63%	64.38%	70.00%	71.23%	65.00%	25.62%	16.25%
1e-4	30.00%	51.88%	45.00%	30.00%	30.00%	26.87%	30.00%	24.38%	17.35%

* The test accuracy is the average value of 10 trials.
** RF, DT, LR and NB denote random forest, decision tree, logistic regression and naive Bayes, respectively.
*** Optimization algorithm in training NN: LM.

Table 6: Robustness result: test accuracy* of different classifiers** on inputs type (d), i.e., windowing data without feature extraction.

Noise Level \ Classifier	CAE+CNN	CNN	NN***	KNN	RF	DT	SVM	LR	NB
1e-5	98.13%	94.06%	75.62%	71.88%	75.62%	81.88%	16.25%	20.63%	16.25%
2e-5	93.75%	92.50%	60.63%	41.25%	73.13%	78.13%	15.35%	21.88%	17.35%
1e-4	41.25%	58.96%	24.37%	32.50%	53.75%	56.25%	16.25%	20.63%	16.87%

* The test accuracy is the average value of 10 trials.
** RF, DT, LR and NB denote random forest, decision tree, logistic regression and naive Bayes, respectively.
*** Optimization algorithm in training NN: cgb.

among the classifiers, CNN, NN, decision tree and random forest follow successively. This fact demonstrates that CAE+CNN has better generalization ability which can accept wider range of data and its suitability to real-life application when there are inevitable data perturbations caused by electrode shift, muscle fatigue, etc..

5.3. Statistical Analysis

To analyze the statistical properties of the classification results, sensitivity, specificity, precision, F1 score and Matthews correlation coefficient [47, 48, 49] are considered in this paper. We select 3 classifiers which have the top three performance in cases (a), (c) and (d) to implement the statistical analysis. As the accuracy of all classifiers in case (b) are below 50% which is not acceptable in practical applications, its statistical analysis is omitted.

Table 7: The formulations of sensitivity, specificity, precision, F1 score and Matthews correlation coefficient.

Sensitivity	TP/(TP+FN)
Specificity	TN/(TN+FP)
Precision	TP/(TP+FP)
F1 Score	2TP/(2TP+FP+FN)
MCC	TP × TN − FP ×FN/sqrt((TP+FP) × (TP+FN) × (TN+FP) × (TN+FN))

Sensitivity, specificity precision, F1 score and Matthews correlation coefficient are the statistical measures of the performance of a binary classification test. Sensitivity measures the percentage of actual positives that are correctly classified as such while specificity measures the percentage of actual negatives that are correctly identified as such [47]. Precision is the proportion of positive output that are truly positive [48]. F1 score is the harmonic average of precision and recall [49] which is more comprehensive and regarded as the balance of precision and recall. Matthews correlation coefficient is the correlation coefficient between the labelled and predicted binary classifications.

Since there are ten classes of hand movements, the above statistics indexes are calculated as follows. Suppose that $E_1, E_2, ..., E_{10}$ represent 10 hand gestures in this paper. 'TP' of E_1 denotes all the E_1 samples that are classified as E_1. 'TN' of E_1 is defined by all the non-E_1 samples that are not classified as E_1. 'FP' of E_1 denotes all the non-E_1 samples that are classified as E_1. 'FN' of E_1 means all the E_1 samples that are not classified as E_1. Replacing E_1 with $E_2, E_3, ..., E_{10}$ and the four terms can be obtained accordingly. The formulations of sensitivity, specificity, precision, F1 score and Matthews correlation coefficient (MCC) refer to Table 7.

The statistical analysis of top three classifiers concerning type (a), (c) and (d) can be found in Table 8, 9 and 10. The tables show that CAE+CNN of type (d) has the best statistical properties, while other classifiers of each case are less capable of classifying some categories of hand movements such as classes 3, 5, 7 and 9.

Table 8: Statistical analysis of top three classifiers in type (a).

Type (a)	Sensitivity			F1 Score			Precision			F1 Score			MCC		
	RF	NN	KNN	RF	NN	KNN	RF	NN	KNN	RF	NN	KNN	RF	NN	NN
Class 1	94%	100%	100%	99%	99%	99%	88%	94%	89%	91%	97%	94%	90%	97%	94%
Class 2	88%	81%	100%	99%	98%	100%	93%	81%	100%	90%	81%	100%	89%	79%	100%
Class 3	94%	94%	100%	100%	100%	99%	100%	100%	94%	97%	97%	97%	96%	96%	97%
Class 4	88%	88%	100%	99%	98%	99%	93%	82%	94%	90%	85%	97%	89%	83%	97%
Class 5	94%	100%	100%	99%	100%	100%	88%	100%	100%	91%	100%	100%	90%	100%	100%
Class 6	88%	88%	94%	100%	100%	100%	100%	100%	100%	93%	93%	97%	93%	93%	96%
Class 7	94%	88%	81%	97%	99%	100%	75%	88%	100%	83%	88%	90%	82%	86%	89%
Class 8	100%	100%	100%	99%	99%	98%	94%	89%	84%	97%	94%	91%	97%	94%	91%
Class 9	88%	94%	88%	100%	100%	100%	100%	100%	100%	93%	97%	93%	93%	96%	93%
Class 10	88%	88%	94%	99%	99%	100%	88%	88%	100%	88%	88%	97%	86%	86%	96%

Table 9: Statistical analysis of top three classifiers in type (c).

Type (c)	Sensitivity			F1 Score			Precision			F1 Score			MCC		
	DT	NN	RF	DT	NN	RF	DT	NN	RF	DT	NN	RF	DT	NN	NN
Class 1	100%	100%	100%	99%	99%	100%	94%	94%	100%	97%	97%	100%	97%	97%	100%
Class 2	100%	100%	88%	99%	100%	100%	89%	100%	100%	94%	100%	93%	94%	100%	93%
Class 3	81%	81%	100%	100%	100%	99%	100%	100%	94%	90%	90%	97%	89%	89%	97%
Class 4	100%	100%	94%	100%	100%	100%	100%	100%	100%	100%	100%	97%	100%	100%	96%
Class 5	100%	100%	100%	99%	99%	99%	89%	89%	94%	94%	94%	97%	94%	94%	97%
Class 6	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Class 7	88%	100%	94%	97%	97%	98%	74%	76%	83%	80%	86%	88%	78%	86%	87%
Class 8	88%	88%	94%	100%	100%	100%	100%	100%	100%	93%	93%	97%	93%	93%	96%
Class 9	81%	81%	94%	100%	100%	100%	100%	100%	100%	90%	90%	97%	89%	89%	96%
Class 10	100%	100%	100%	100%	100%	99%	100%	100%	94%	100%	100%	97%	100%	100%	97%

5.4. Computational Costs of the Proposed Approach

To obtain the computation time, we used the same computer (i7-7700 CPU, Nvidia Geforce GTX 1050) for average training time recording. The training of CAE+CNN consumed 308 seconds while the individual CNN took 589 seconds, which demonstrates that CAE+CNN improves the training efficiency. Besides, after data compression carried out by CAE, the input size and the following parameters of classifier CNN are largely reduced, the memory usage during training decreases to half accordingly when feeding the same type of the inputs.

From the above results, it can be concluded that the proposed CAE+CNN improves the computational efficiency and reduce the memory usage. Besides, the feature extraction ability enables CAE to capture more stable structures of inputs such that the model is more robust to partial destruction of the inputs. Also, for EMG signal processing, CAE has fewer number of model parameters due to the utilization of convolutional layers and has non-linear transformation, which shows advantages over conventional dimension reduction method like principal component analysis with dense and linear transformation.

Table 10: Statistical analysis of top three classifiers in type (d).

Type (d)	Sensitivity			F1 Score			Precision			F1 Score			MCC		
	CAE+CNN	CNN	NN	CAE+CNN	CNN	NN	CAE+CNN	CNN	NN	CAE+CNN	CNN	NN	CAE+CNN	CNN	NN
Class 1	100%	100%	100%	100%	100%	99%	100%	100%	89%	100%	100%	94%	100%	100%	94%
Class 2	100%	100%	100%	100%	100%	99%	100%	100%	89%	100%	100%	94%	100%	100%	94%
Class 3	100%	75%	88%	100%	99%	100%	100%	86%	100%	100%	80%	93%	100%	78%	93%
Class 4	100%	100%	88%	100%	99%	96%	100%	94%	70%	100%	97%	78%	100%	97%	76%
Class 5	100%	94%	100%	99%	99%	99%	94%	94%	94%	97%	94%	97%	97%	93%	97%
Class 6	100%	100%	75%	100%	100%	100%	100%	100%	100%	100%	100%	86%	100%	100%	85%
Class 7	100%	88%	63%	100%	98%	94%	100%	82%	56%	100%	85%	59%	100%	83%	54%
Class 8	100%	100%	75%	100%	100%	99%	100%	100%	86%	100%	100%	80%	100%	100%	78%
Class 9	94%	94%	100%	100%	99%	100%	100%	94%	100%	97%	94%	100%	96%	93%	100%
Class 10	100%	100%	81%	100%	100%	100%	100%	100%	100%	100%	100%	90%	100%	100%	89%

6. Conclusion

In this paper, machine learning techniques are employed to classify 10 hand gestures based on EMG signals. Data normalization, feature extraction and windowing method, which form four inputs cases, are utilized for data pre-processing. The results demonstrate that the combination of windowing method, CAE+CNN and majority voting achieves the best performance, strongest robustness and satisfactory statistical properties. The main contributions of this paper is the design of CAE+CNN scheme which reduces the dimensions, redundancy and computational costs for EMG signals processing and achieves 99.38% classification accuracy. Auxiliary methods such as windowing scheme, feature extraction, majority voting and 8 widely used and state-of-the-art classifiers are also evaluated and compared in this paper. Feature extraction improves performance for NN, KNN, random forest, decision tree, SVM but has little improvement of deep learning models' performance. Besides, in most cases, the combination of feature extraction and windowing approach offers models better performance and stronger robustness.

Future work may investigate the real-time EMG signals classification using the CAE+CNN structure with windowing and voting scheme, which could produce and adjust class decisions along the sliding windows, so that the results can be applied to the real-time control of upper-limb prostheses.

Disclosure statement

No potential conflict of interest was reported by the authors.

Acknowledgement

This work was partly supported by King's College London and the China Scholarship Council.

References

[1] G. Li, Electromyography pattern-recognition-based control of powered multifunctional upper-limb prostheses, in: Advances in applied electromyography, InTech, 2011.

[2] A. D. Orjuela-Cañón, A. F. Ruíz-Olaya, L. Forero, Deep neural network for EMG signal classification of wrist position: Preliminary results, in: Proceedings of the 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI), IEEE, 2017, pp. 1–5.

[3] R. M. Rangayyan, Biomedical signal analysis, Vol. 33, John Wiley & Sons, 2015.

[4] M. R. Ahsan, M. I. Ibrahimy, O. O. Khalifa, Optimization of neural network for efficient EMG signal classification, in: Proceedings of the 2012 8th International Symposium on Mechatronics and its Applications, IEEE, 2012, pp. 1–6.

[5] R. N. Khushaba, S. Kodagoda, M. Takruri, G. Dissanayake, Toward improved control of prosthetic fingers using surface electromyogram (EMG) signals, Expert Systems with Applications 39 (12) (2012) 10731–10738.

[6] B. Paaßen, A. Schulz, J. Hahne, B. Hammer, Expectation maximization transfer learning and its application for bionic hand prostheses, Neurocomputing 298 (2018) 122–133.

[7] O. Faust, Y. Hagiwara, T. J. Hong, O. S. Lih, U. R. Acharya, Deep learning for healthcare applications based on physiological signals: A review, Computer methods and programs in biomedicine 161 (2018) 1–13.

[8] R. Rao, R. Derakhshani, A comparison of EEG preprocessing methods using time delay neural networks, in: Proceedings of the 2nd International IEEE EMBS Conference on Neural Engineering, IEEE, 2005, pp. 262–264.

[9] R. Kohavi, G. H. John, Wrappers for feature subset selection, Artificial intelligence 97 (1-2) (1997) 273–324.

[10] H. K. Lam, U. Ekong, B. Xiao, G. Ouyang, H. Liu, K. Y. Chan, S. H. Ling, Variable weight neural networks and their applications on material surface and epilepsy seizure phase classifications, Neurocomputing 149 (2015) 1177–1187.

[11] U. Ekong, H. K. Lam, B. Xiao, G. Ouyang, H. Liu, K. Y. Chan, S. H. Ling, Classification of epilepsy seizure phase using interval type-2 fuzzy support vector machines, Neurocomputing 199 (2016) 66–76.

[12] S. Alty, H. K. Lam, J. Prada, On the applications of heart disease risk classification and hand-written character recognition using support vector machines, in: Computational Intelligence And Its Applications: Evolutionary Computation, Fuzzy Logic, Neural Network and Support Vector Machine Techniques, World Scientific, 2012, pp. 213–253.

[13] M. Reaz, M. Hussain, F. Mohd-Yasin, Techniques of emg signal analysis: detection, processing, classification and applications (correction), Biological procedures online 8 (1) (2006) 163–163.

[14] A. Subasi, Classification of EMG signals using combined features and soft computing techniques, Applied soft computing 12 (8) (2012) 2188–2198.

[15] Y. Wang, C. Wang, Z. Wang, X. Wang, Y. Li, Hand gesture recognition using sparse autoencoder-based deep neural network based on electromyography measurements, in: Nano-, Bio-, Info-Tech Sensors, and 3D Systems II, Vol. 10597, International Society for Optics and Photonics, 2018, p. 105971D.

[16] Y. Bengio, A. C. Courville, P. Vincent, Unsupervised feature learning and deep learning: A review and new perspectives, CoRR, abs/1206.5538 1 (2012).

[17] A. Brunetti, D. Buongiorno, G. F. Trotta, V. Bevilacqua, Computer vision and deep learning techniques for pedestrian detection and tracking: A survey, Neurocomputing 300 (2018) 17–33.

[18] N. Zeng, Z. Wang, H. Zhang, W. Liu, F. E. Alsaadi, Deep belief networks for quantitative analysis of a gold immunochromatographic strip, Cognitive Computation 8 (4) (2016) 684–692.

[19] N. Zeng, Z. Wang, B. Zineddin, Y. Li, M. Du, L. Xiao, X. Liu, T. Young, Image-based quantitative analysis of gold immunochromatographic strip via cellular neural network approach, IEEE transactions on medical imaging 33 (5) (2014) 1129–1136.

[20] A. Kalantari, A. Kamsin, S. Shamshirband, A. Gani, H. Alinejad-Rokny, A. T. Chronopoulos, Computational intelligence approaches for classification of medical data: State-of-the-art, future challenges and research directions, Neurocomputing 276 (2018) 2–22.

[21] U. C. Allard, F. Nougarou, C. L. Fall, P. Giguère, C. Gosselin, F. Laviolette, B. Gosselin, A convolutional neural network for robotic arm guidance using semg based frequency-features, in: Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2016, pp. 2464–2470.

[22] K.-H. Park, S.-W. Lee, Movement intention decoding based on deep learning for multiuser myoelectric interfaces, in: Proceedings of the 2016 4th International Winter Conference on Brain-Computer Interface (BCI), IEEE, 2016, pp. 1–2.

[23] X. Zhai, B. Jelfs, R. H. Chan, C. Tin, Self-recalibrating surface EMG pattern recognition for neuroprosthesis control based on convolutional neural network, Frontiers in neuroscience 11 (2017) 379.

[24] S. Benatti, F. Montagna, V. Kartsch, A. Rahimi, D. Rossi, L. Benini, Online learning and classification of EMG-based gestures on a parallel ultra-low power platform using hyperdimensional computing, IEEE transactions on biomedical circuits and systems 13 (3) (2019) 516–528.

[25] A. Waris, I. K. Niazi, M. Jamil, K. Englehart, W. Jensen, E. N. Kamavuako, Multiday evaluation of techniques for EMG based classification of hand motions, IEEE journal of biomedical and health informatics (2018) 1–1.

[26] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F. E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.

[27] N. Zeng, H. Zhang, B. Song, W. Liu, Y. Li, A. M. Dobaie, Facial expression recognition via learning deep sparse autoencoders, Neurocomputing 273 (2018) 643–649.

[28] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, T. Ball, Deep learning with convolutional neural networks for brain mapping and decoding of movement-related information from the human EEG, arXiv preprint arXiv:1703.05051.

[29] M. Längkvist, L. Karlsson, A. Loutfi, A review of unsupervised feature learning and deep learning for time-series modeling, Pattern Recognition Letters 42 (2014) 11–24.

[30] J. Bouvrie, Notes on convolutional neural networks (2006).
URL `http://cogprints.org/5869/`

[31] J. Masci, U. Meier, D. Cireşan, J. Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction, in: Proceedings of the International Conference on Artificial Neural Networks, Springer, 2011, pp. 52–59.

[32] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.

[33] T. G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization, Machine learning 40 (2) (2000) 139–157.

[34] N. S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, The American Statistician 46 (3) (1992) 175–185.

[35] H. Yu, B. M. Wilamowski, Levenberg-marquardt training, Industrial electronics handbook 5 (12) (2011) 1–1.

[36] F. Tenore, A. Ramos, A. Fahmy, S. Acharya, R. Etienne-Cummings, N. V. Thakor, Towards the control of individual fingers of a prosthetic hand using surface EMG signals, in: Proceedings of the 2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE, 2007, pp. 6145–6148.

[37] F. V. Tenore, A. Ramos, A. Fahmy, S. Acharya, R. Etienne-Cummings, N. V. Thakor, Decoding of individuated finger movements using surface electromyography, IEEE transactions on biomedical engineering 56 (5) (2009) 1427–1434.

[38] R. J. Smith, F. Tenore, D. Huberdeau, R. Etienne-Cummings, N. V. Thakor, Continuous decoding of finger position from surface EMG signals for the control of powered prostheses, in: Proceedings of the 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE, 2008, pp. 197–200.

[39] A. Andrews, E. Morin, L. McLean, Optimal electrode configurations for finger movement classification using EMG, in: Proceedings of the 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE, 2009, pp. 2987–2990.

[40] G. Tsenov, A. Zeghbib, F. Palis, N. Shoylev, V. Mladenov, Neural networks for online classification of hand and finger movements using surface EMG signals, in: Neural Network Applications in Electrical Engineering. NEUREL 2006. 8th Seminar on, IEEE, 2006, pp. 167–171.

[41] D. Peleg, E. Braiman, E. Yom-Tov, G. F. Inbar, Classification of finger activation for use in a robotic prosthesis arm, IEEE Transactions on Neural Systems and Rehabilitation Engineering 10 (4) (2002) 290–293.

[42] A. Phinyomark, P. Phukpattaranont, C. Limsakul, Feature reduction and selection for EMG signal classification, Expert Systems with Applications 39 (8) (2012) 7420–7431.

[43] M. A. Oskoei, H. Hu, Myoelectric control systems—a survey, Biomedical Signal Processing and Control 2 (4) (2007) 275–294.

[44] M. Zecca, S. Micera, M. Carrozza, P. Dario, Control of multifunctional prosthetic hands by processing the electromyographic signal, Critical Reviews^{TM} in Biomedical Engineering 45 (2017) 1–6.

[45] S. Moein, Medical diagnosis using artificial neural networks, IGI global, 2014.

[46] K. Englehart, B. Hudgins, A robust, real-time control scheme for multifunction myoelectric control, IEEE transactions on biomedical engineering 50 (7) (2003) 848–854.

[47] A. G. Lalkhen, A. McCluskey, Clinical tests: sensitivity and specificity, Continuing Education in Anaesthesia Critical Care & Pain 8 (6) (2008) 221–223.

[48] D. M. Powers, Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation (2011).

[49] C. Goutte, E. Gaussier, A probabilistic interpretation of precision, recall and f-score, with implication for evaluation, in: Proceedings of the European Conference on Information Retrieval, Springer, 2005, pp. 345–359.

View publication stats