Deep Learning in Gesture Recognition Based on sEMG Signals

P. Tsinganos[1], A. Skodras[1], B. Cornelis[2], B. Jansen[2]
[1]Department of Electrical and Computer Engineering, University of Patras, Greece
{panagiotis.tsinganos,skodras}@ece.upatras.gr
[2]Department of Electronics and Informatics, Vrije Universiteit Brussel, Belgium
{bcorneli,bjansen}@etrovub.be

Abstract

Over the past years, Deep Learning methods have shown promising results to a wide range of research fields including image classification and natural language processing. Their increased success rates have drawn the attention of many researchers from various domains. This chapter investigates the application of Deep Learning methods to the problem of electromyography-based gesture recognition. A signal processing pipeline based on Deep Learning is presented through examples taken from the literature, whereas the details of state-of-the-art neural network architectures are discussed. In addition, this chapter illustrates a few ways adopted from image classification tasks that visualize what the neural network learns. Finally, new approaches are proposed and evaluated with publicly available datasets.

Keywords— sEMG, gesture recognition, Deep Learning, signal processing

1 Introduction

Over the last decades there has been particular interest in gesture recognition for human-computer interaction (HCI). This particular combination finds many applications, including sign language recognition, robotic equipment control, virtual reality gaming, and prosthetics control [1]. Among the various sensor modalities that have been used to capture hand gesture information, electromyography (EMG) is considered more appropriate since it captures the muscles electrical activity; the physical phenomenon that results in hand gestures. EMG data can be recorded either with invasive or non-invasive methods. Surface electromyography (sEMG) is a technique that measures muscles action potential from the surface of the skin, contrary to invasive methods that penetrate the skin to reach the muscle.

A popular approach to sEMG-based gesture recognition consists of using pattern recognition methods derived from Machine Learning (ML) [2]. Conven-

tional ML pipelines include data acquisition, feature extraction, model definition and inference. Acquisition of sEMG signals involves attaching one or more electrodes around the target muscle group. The features used for classification are usually hand-crafted by human experts and capture the temporal and frequency characteristics of the data. They serve as the input to ML classifiers, such as k-Nearest Neighbors (kNN), Support Vector Machines (SVM), Multi-Layered Perceptron (MLP), Linear Discriminant Analysis (LDA), and Random Forests (RF), where the classifiers parameters are adjusted towards accurate classification.

Deep Learning (DL) is a class of ML algorithms that has revolutionized many fields of data analysis [3]. For example, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) were successfully deployed for image classification and speech recognition tasks respectively. DL methods differ from conventional ML approaches in that feature extraction is part of the model definition, therefore obviating the need for hand-crafted features. Although these methods are not new [3], they recently gained more attention due to the increased availability of abundant data and vast improvements in computing hardware allowing these computationally demanding methods to be executed in less time.

In view of these advancements, the problem of sEMG-based gesture recognition has been formulated as a DL classification task. The main steps in a DL pipeline are shown in Figure 1. The acquisition and preprocessing methods applied in this methodology are quite similar to classical EMG signal processing techniques. Compared to other methods, DL requires many data, which in the case of sEMG can be difficult to acquire. Therefore, an augmentation step is usually employed to create synthetic data from a small dataset. The most significant part in a DL pipeline is the model definition. Two types of neural networks are mostly used, CNNs and RNNs, from which different variants can be created or improved with Transfer Learning approaches. Finally, once the model is specified, the last step is the analysis of the network performance using evaluation metrics, whereas there exist methods that try to explain how neural networks make their decision.

This chapter discusses techniques of approaching the task of gesture recognition based on sEMG signals using DL. Firstly, data acquisition methods and parameters that affect the quality of the recorded signal are briefly presented. In Section 3 common practices for data preparation, including signal processing and data augmentation are discussed. Then, state-of-the-art neural network architectures are introduced, followed by the presentation of evaluation and

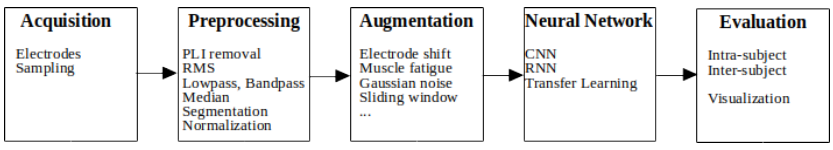


Figure 1: Flowgraph of an EMG-based gesture recognition system

network visualization techniques. Finally, future directions are discussed.

2 Data Acquisition

The detection of bioelectric signals, including EMG, is mostly based on bipolar electrodes placed over the region of interest. In the case of myoelectric signals, the electrodes are attached above the target muscle group and measure the generated signals while a movement is performed. Under this setup, the recorded signal provides information about the electrical activity of the muscle over time. However, when performing a gesture more than one muscles participate in different time instants, therefore multichannel detection is used, based on 1D or 2D electrode arrays that cover the forearm muscles. The spatially sampled signal can either be the analog surface potential or EMG features such as, Average Rectified Value (ARV), and Root Mean Square (RMS), estimated over a time interval [4]. An example of a single channel EMG surface potential and extracted features are shown in Figure 2. Since the EMG frequency content lies in the range of 5-500Hz, a sampling frequency of at least 1kHz is required − yet, most frequency power is contained between 20 and 150 Hz

In the literature of EMG gesture recognition with Deep Learning, the analog surface potential or the RMS envelope are mostly preferred. The authors of [5] record data from a 1D electrode array that reports the RMS signal, whereas in [6], [7] a high density electrode grid measures the instantaneous image of the analog myoelectric activity. Details on the acquisition practices found in literature are shown in Table 1.

From the generation in the muscle fibers to the detection by the electrodes, the EMG signal can be influenced by many factors. These are either related to the detection system or the physiological characteristics of the muscles. In the first category, the environmental noise (e.g. power line interference) and the noise induced by the electronics are the most important parameters that affect the quality of the recorded signal. To lessen their effect, proper skin prepara-

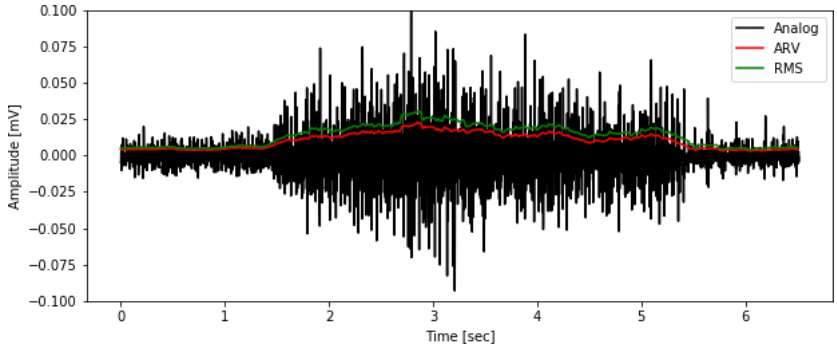


Figure 2: Example of a raw sEMG signal and its estimated amplitude envelope with ARV and RMS

Table 1: Acquisition practices in literature

Ref.	Dataset	Sampling (Hz)	Electrodes	Feature
[5]	Ninapro-DB1	100	$1 \times 8 + 2$	RMS
[5]	Ninapro-DB2	2000	$1 \times 8 + 4$	Analog
[6]	CapgMyo	1000	8×16	Analog
[7], [8]	CSL-HDEMG	2000	7×24	Analog
[9]	Myo	200	1×8	Analog

tion (e.g. rubbing with abrasive conductive paste) and careful instrumentation amplifier design are usually proposed. In addition, capacitive electrodes have shown to reduce the recorded noise in long-term monitoring applications, compared to bipolar electrodes that require a galvanic contact with the body [4]. The most important factors from the second category are the thickness of the tissue separating the muscle from the electrode, the cross-talk from other muscles and biomedical signals (e.g. ECG), as well as changes to muscle geometry due to movement [10]. Some of the induced artifacts by these inherent factors can be removed either with careful electrode arrangement (e.g. taking into consideration the possible muscle translation due to dynamic joint movement) or signal processing methods (e.g. ECG cross-talk reduction) [10].

3 Data Preparation

3.1 EMG preprocessing

Basic filtering methods applied in amplitude and spectral characteristics of muscle activity are part of the data preparation step of a DL pipeline as well. These steps are modified depending on which EMG signal characteristic is used for the classification task. Common preprocessing steps for sEMG signals are shown in Table 2.

When the EMG amplitude is used for the gesture classification, the main steps are noise reduction, rectification and smoothing. The aim of noise reduction is to eliminate additive noise, artifacts, and power-line interference that contaminate the EMG signal [4]. The next step computes the absolute value of each EMG sample, which makes the computation of amplitude parameters, like the mean, feasible (raw EMG signals have an average value of zero) [10]. Finally, smoothing is applied to extract the amplitude estimate. There are mainly two methods encountered in the literature. The first algorithm simply computes the moving average of the signal (i.e. the ARV), whereas the second is the RMS. Both methods require the selection of an appropriate window length over which the signal is considered stationary. Typically, windows of 100ms to 200ms are used [4], [10]. Smoothing can also be performed by low-pass filtering, where 2nd order or higher Butterworth filters with cutoff frequencies between 1Hz and 6Hz are common [10]. In [5], the RMS value is calculated from 200ms windows

and an additional low-pass filtering is performed with a 2nd order Butterworth filter with 1Hz cut-off, whereas the authors of [7] employ a non-linear low-pass filter (median filtering) to the instantaneous EMG maps.

Apart from EMG amplitude estimates, spectral representations based on the Short Time Fourier Transform (STFT) and Wavelet transforms have also been used. In the first case, the spectrogram of EMG segments is computed using the STFT, so that from a single EMG channel a 2D signal is generated with the two axis representing the time and the frequency. The window length and overlap used for the calculation of the STFT are important parameters that affect the time and frequency resolution. On the other hand, utilizing the Wavelet Transform allows for a multi-resolution analysis which is well suited for non-stationary signals. In that case, the Mexican hat wavelet is preferred since it best approximates the action potential generated by the muscles [11]. The authors of [12] calculate the spectrogram of each 200ms (400 samples) segment with a 256 samples window, whereas in [9] it is calculated with a 28 samples window for EMG segments of 260ms (52 samples) duration.

3.2 Data augmentation

One of the prerequisites for training a Deep Neural Network with good generalization properties is the availability of a huge dataset. Although there are a few publicly available sEMG datasets, direct use of these data is not always feasible since the acquisition system and the preprocessing steps vary. Therefore, data augmentation techniques have been applied to increase the size of the training set. In the domain of image classification, additive Gaussian noise and affine transformations, like rotation and translation, have resulted in better generalization [3]. However, these methods are not directly applicable to sEMG signals.

In order to augment time-series signals, a few data augmentation approaches have been proposed (Table 3). These include additive noise, sliding window, permutation, shift, warping, and scaling. Additive noise and sliding window have successfully been applied to sEMG data [5], [9], whereas the applicability of permutations and warping to EMG signals has not been verified. These methods are presented next and their relation to EMG data is briefly discussed. Examples of their applications to sEMG signals can be seen in Figure 3.

Table 2: Data preparation practices in literature

Ref.	**Preprocessing**	**EMG characteristic**
[5]	PLI[1] removal, RMS, low-pass	RMS
[6], [7]	PLI removal, median, normalization	raw
[9]	(not mentioned)	spectrogram
[12]	PLI removal	spectrogram

[1] PLI: Power-Line Interference

Table 3: Data augmentation practices in literature

Ref.	Augmentation	Dataset[1]
[13]	Time warping	ECG
[5]	Additive noise	EMG
[7], [6]	Electrode shift	EMG
[14]	Permutation, Scaling, Magnitude warping, Sliding window	IMU
[9]	Muscle Fatigue, Electrode Displacement	EMG

[1] ECG: Electrocardiogram, EMG: Electromyogram, IMU: Inertial Measurement Unit

Considering that training examples need to have a set duration, a *sliding window* can be applied to the EMG signal to generate these examples. Using overlapping windows (with constant or random overlap) is similar to the translation operation applied to images [9], [14].

Similarly to noise augmentation found in image classification, adding Gaussian noise with a defined signal to noise ratio has been used. This *additive noise* can simulate the noise found in EMG recordings without altering the class of the performed gesture.

Augmenting the training data by randomly shifting the channels of the EMG signal can simulate *electrode shift* that may happen between recording sessions [6]. Additionally, electrode displacement due to muscle movement can be emulated by shifting part of the power spectrum from one channel to an adjacent one [9].

Accounting for the fact that during movement repetitions the EMG amplitude and duration cannot be replicated (due to the stochastic generation of action potentials) [10], perturbing the location of sEMG samples (*time-warping*) or their value (*magnitude-warping*) is a way to augment sEMG data. To this effect, a smooth random curve can be used to either distort the time intervals between the samples or change their value through multiplication [14]. In addition, the magnitude of the sEMG samples can be altered by multiplication with a random scalar (*scaling*).

Another augmentation technique is to simulate *muscle fatigue*. The main effect of muscular fatigue on the EMG signal is the compression of the power spectrum. As a result, the amplitude of lower frequencies is increased, whereas higher frequencies are attenuated. Therefore, this method can be easily implemented in the frequency domain, where part of the power of a frequency bin is redistributed to adjacent lower frequency bins [9].

Finally, perturbations to the location of the samples can be perfromed by dividing the signal into smaller slices and then permuting them [14]. However, the *permutation* method cannot be considered appropriate for sEMG data, since the generated signal does not resemble a 'true' sEMG signal.

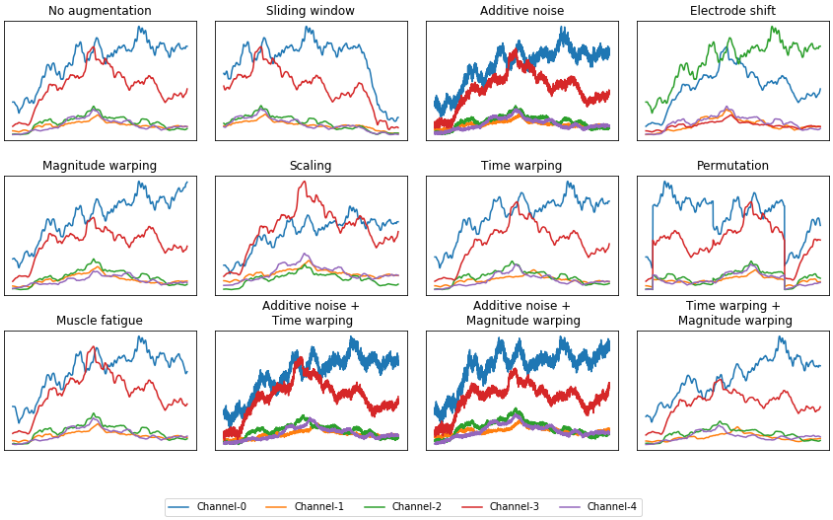


Figure 3: Examples of data augmentation techniques applied on sEMG signal

4 Network Architectures

sEMG-based hand gesture recognition can be formulated either as an image classification problem using Convolutional Neural Networks (CNNs), where the input sEMG image has a size of $H \times W \times 1$ (height×width×depth) or as a sequence classification task using Recurrent Neural Networks (RNNs). In the first case, various approaches have been employed to construct an sEMG image, among which (i) the instantaneous sEMG signals from a 2D electrode array (where the width and the height of the array match the dimensions of the image) [7], [15], [6], (ii) segments of sEMG signals using time-windows (where the width matches the number of electrodes and the height is equal to the window length) [5], and (iii) spectrograms or Wavelet transformations of sEMG segments (where for each channel of the EMG a transformation is applied resulting in a time-frequency-space representation) [12], [9]. On the other hand, there are no works utilising RNNs, probably due to the extensive use of CNNs, which makes the application of recurrent networks a possible research subject.

4.1 CNN architectures

CNN models have been widely used for image classification problems, since the neurons (filters) are arranged in three dimensions (height, width, depth) similar to images. A typical CNN architecture consists of convolutional layers followed by a non-linear activation function that are stacked many times. The operation performed by a convolutional layer (assuming zero bias) can be described by

the equation:

$$z[i,j,n] = \sum_{m=0}^{N_{in}-1} \sum_{l=\lfloor -f_h/2 \rfloor}^{\lfloor f_h/2 \rfloor} \sum_{k=\lfloor -f_w/2 \rfloor}^{\lfloor f_w/2 \rfloor} x[i-k, j-l, m] \times h_n[k,l,m] \qquad (1)$$

where $n \in [0, N_{out}-1]$, z is the output feature map, x the input, h_n the n^{th} filter of the layer, N_{in} the depth of the input, N_{out} the number of filters, and f_h/f_w the height/width dimension of the filter. Many activation functions have been proposed in the literature, yet the rectified linear unit (ReLU) and its variants are preferred. The equation for this function is given by:

$$ReLU(x) = \max\{0, x\} \qquad (2)$$

In addition, pooling layers are employed to reduce the spatial dimensions and therefore decrease the number of learned parameters. Pooling operates independently on every depth slice of the input and resizes it spatially, using either the *max* or *average* operation. Finally, a few fully connected layers compute the class scores, while a softmax activation function, $\sigma(x)$, normalizes these values into class probabilities as follows:

$$\sigma(\boldsymbol{z})_n = \frac{e^{z_n}}{\sum_{k=0}^{N-1} e^{z_k}}, \quad n \in [0, N-1] \qquad (3)$$

where $\boldsymbol{z} = [z_0, ..., z_{N-1}]$ is the output of the last fully connected layer, and N the number of the output units. CNNs are trained similar to regular neural networks. A loss function is defined and the neuron parameters are optimized using the back-propagation algorithm such that the value of the loss function is minimized. Stochastic gradient descent (SGD) is the basic algorithm used for the optimization, while many improved versions of it have been proposed. The choice of the loss function to minimize depends on the problem: for the classification between two (many) classes, the binary (categorical) cross-entropy is used, whereas for regression problems the mean squared error is preferred.

In the literature of EMG-based gesture recognition, the above principles are followed. What affects the number of the layers and the size of the filters the most is the dimensions of the input image. Usually, the first two dimensions correspond to the size of the 2D electrode grid, whereas in the case of 1D electrode array, one of the width and height corresponds to time and the other to the size of the array. Regarding the depth dimension, it is typically set to 1. Table 4 summarizes the main characteristics of these architectures. The filter sizes and parameter values refer to the publicly available Ninapro dataset [16], which contains sEMG recordings of subjects performing more than 50 gestures (Table 6). In the subsequent analysis, in case the depth dimension is omitted it is considered to be equal to 1 and the symbol $\#G$ denotes the number of class labels.

The network architecture used in [5] is comprised of five convolutional blocks, where each one contains a convolutional layer and an activation function. The

Table 4: CNN architectures used in EMG-based gesture recognition

Ref.	**Input dimensions (height×width×depth)[1]**	**Layers**	**Layer types[2]**	**Parameters**
[5]	15×10 $(t \times c)$	5	CONV, POOL	84,853[3]
[7], [6]	1×10 $(t \times c)$	8	CONV, LC, FC	555,897[3]
[15]	20×10 $(t \times c)$	44	CONV, LC, FC	2,717,877[3]
[9]	$8 \times 14 \times 4$ $(c \times f \times t)$	8	CONV, FC	71,825[4]

[1] t: time, c: electrode channels, f: frequency
[2] CONV: Convolutional, POOL: Average Pooling, FC: Fully connected, LC: Locally connected
[3] Calculated for the Ninapro DB1 with $\#G = 53$ gestures
[4] Calculated for the Ninapro DB5 with $\#G = 53$ gestures

activation function for the last block is a softmax, whereas for the rest a ReLU is used. In addition, an average pooling layer is used after the second and third blocks. The filter sizes of the convolutions are 1×10, 3×3, 5×5, 5×1 and 1×1, while the pooling is performed with 3×3 filters. Regarding the number of filters in each block, the model computes 32, 32, 64, 64 and $\#G$ filters respectively. In this architecture, the convolution at the output layer is equivalent to the dot product performed by a fully connected layer. The output of the fourth convolution has a size of $(1, 1, 64)$. Thus, substituting $N_{in} = 64$, $N_{out} = \#G$, $f_h = f_w = 1$ into Equation (1) results in:

$$y[0, 0, n] = \sum_{m=0}^{d-1} x[0, 0, m] \times h_n[0, 0, m] = \boldsymbol{x} \cdot \boldsymbol{h_n}, \quad n \in [0, N_{out} - 1]$$

which is equal to the dot product performed by a fully connected layer.

The convolutional network of [7] and [6] consists of eight blocks. The first two are convolutional layers, each of which consists of 64 3×3 filters. The next two blocks use locally connected layers with 64 filters of size 1×1. The next three are fully connected layers consisting of 512, 512 and 128 units, respectively. The network ends with a $\#G$-way fully connected layer and a softmax function. After each layer a batch normalization [17] and a ReLU non-linearity are used. In addition, dropout [18] with a probability of 0.5 is inserted after the fourth, fifth and sixth blocks to reduce overfitting. An imporovement on this architecture is presented in [15], where the input image is divided into M equally sized patches and processed separately. Each one of these segments is the input to one of M subnetworks consisting of the first four blocks of [7]. The output feature maps of these networks are concatenated into a single feature map and the last fully connected layers follow. Among the various configurations used to divide the initial image, the strategy that creates patches equal to the number of electrode channels (i.e. an $L \times 10$ $(t \times c)$ image divided into 10 patches of size $L \times 1$) performs best.

Finally, the model proposed in [9] is similar to the improvement proposed in [15] since the input is divided into smaller patches. However, the input is a time-frequency representation of sEMG segments arranged into an image

of size ($c \times f \times t$). Another difference is that a parametric ReLU (PReLU) activation function is applied to the initial input before the separation. Each patch is processed by two blocks consisting of a convolutional layer, a batch normalization [18] and a parametric exponential linear unit (PELU) activation. The convolutional layers use 12 filters with a size of 4×3 and 24 filters with a size of 3×3, respectively. The output feature maps are concatenated in an element-wise summation fashion and a convolutional block of 24 filters of size 3×3, follows. Then, three fully connected layers of 100, 100 and $\#G$ units come before the final softmax activation. This model is also augmented by Transfer Learning techniques since the trained network exploits information learned during a pre-training stage.

4.2 RNN architectures

RNNs have recently become popular due to succesful applications in problems related to sequential data [19]. Unlike feedforward networks, RNNs exhibit memory properties, which makes them suitable for processing sequence of values. Among the different variants of RNNs, Long Short Term Memory (LSTM) [20] networks are widely used as a result of their capability to learn long-term dependencies without suffering from vanishing/exploding gradient problems.

LSTMs belong to the category of gated RNNs which use gates (a sigmoid activation function followed by pointwise multiplication) to create paths through time that have derivatives that neither vanish nor explode [3]. The basic model of an LSTM cell is shown in Figure 4. Compared to an RNN, it contains three gating units ($\boldsymbol{f_t}$, $\boldsymbol{i_t}$, and $\boldsymbol{o_t}$) that control the flow of information as follows [21]: (i) the forget gate $\boldsymbol{f_t}$ decides how much of the previous cell state ($\boldsymbol{C_{t-1}}$) information to retain, (ii) the input gate $\boldsymbol{i_t}$ determines the amount of new information ($\tilde{\boldsymbol{C}}_{\boldsymbol{t}}$) to store into the cell state ($\boldsymbol{C_t}$), and (iii) the output gate $\boldsymbol{o_t}$ decides what part of the cell state will go through the output ($\boldsymbol{h_t}$).

Although the temporal nature of sEMG signals suggests that an LSTM network should be able to map a sequence of sEMG values to a gesture label, no approach in the literature exploits these properties. A possible explanation is that recurrent networks require a higher memory bandwidth compared to CNN models. In addition, it has been shown recently [22] that convolutional networks that take into account temporal information can surpass LSTMs in a wide range of tasks. Nevertheless, in this section we try to approach the problem of sEMG-based gesture recognition using a simple LSTM architecture. This consists of one LSTM layer with 128 cells followed by a fully connected layer with $\#G$ units. To investigate whether the addition of more LSTM layers improves performance, another variant of this architecture with two stacked LSTM layers, having 128 cells each, is evaluated. In addition, considering that all the training examples within a batch should have the same dimensions, the sEMG sequences are zero-padded to the longest gesture duration of the dataset. Table 5 summarizes the main characteristics of these architectures.

Similar to CNNs, after model definition, a loss function is defined and the RNN parameters are optimized using back-propagation algorithm.

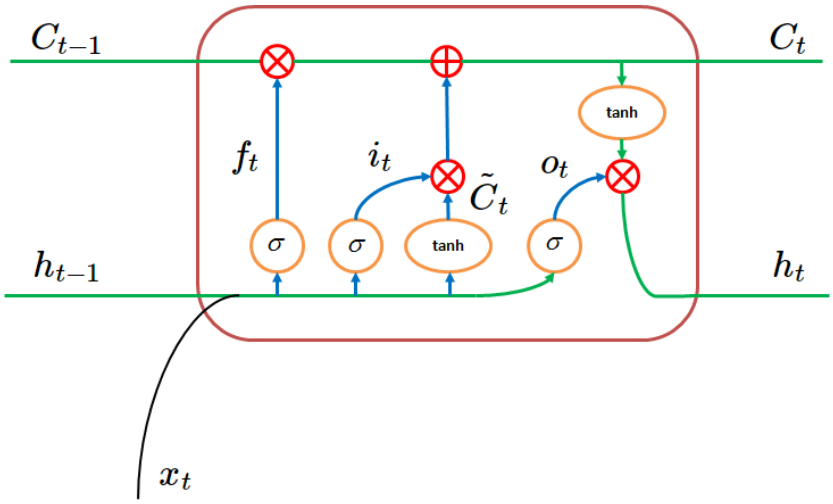


Figure 4: Basic model of an LSTM cell (taken from `https://chunml.github.io/ChunML.github.io/project/Creating-Text-Generator-Using-Recurrent-Neural-Network/`)

Table 5: Simple LSTM architecture for EMG-based gesture recognition

Input dimensions (time×channels)	**LSTM cells**	**Layer types**[1]	**Parameters**
200×10	128	LSTM, FC	78,005[2]
200×10	128, 128	LSTM, FC	209,589[2]

[1] FC: Fully connected
[2] Calculated for the Ninapro DB1 with $\#G = 53$ gestures

5 Network Evaluation

5.1 Evaluation schemes

Once the neural network is defined, the capacity of the model to learn from the data needs to be evaluated. At this step, the training dataset is partitioned into train and validation sets according to an evaluation scheme. In the literature of sEMG-based gestures recognition, two approaches are followed [6]: (i) intra-subject, and (ii) inter-subject evaluation.

In the first case, data recorded from one subject are partitioned into train and validation sets. The network is optimized on the train data and the accuracy on the validation set is measured. This procedure is repeated for all the subjects available in the dataset and an average performance is reported. Usually, during a recording session, each gesture is repeated more than once, therefore the data can be partitioned based on the gesture repetitions. For example, in [5] repetitions 2,5, and 7 comprise the validation set, while the rest (repetitions

1,3,4,6,8,9, and 10) are used for training. Additionally, a cross-validation scheme can be applied for each subject, where one repetition is used for validation and the rest for training [6]. On the other hand, the inter-subject evaluation follows a leave-one-subject-out cross-validation, where the validation set consists of all the repetitions of one subject, while for training the repetitions of all the other subjects are used [6]. This is repeated until the recordings of all the subjects have been in the validation set.

These evaluation approaches are not substitute of one another, rather they should both be applied since they provide different insights about the network. With an intra-subject evaluation, one obtains a metric about the ability of the network to generalize to unseen data from the same subject it was trained on. On the other hand, an inter-subject evaluation is useful for the assessment of methods that tackle subject variability problems (e.g. electrode shift, fatigue), as well as measure the ability of the network to generalize when the source and target distributions differ (Domain Adaptation, Transfer Learning).

In Section 5.3 an application of these schemes is presented with respect to the Ninapro-DB1 dataset.

5.2 Learning visualization

Although deep neural networks exhibit state-of-the-art performance in many tasks, it is difficult to understand why some models perform better than others or how one architecture can be improved. Recently, many researchers have made progress in opening the black box and various neural network interpretation methods have been proposed [23], [24], [25], [26]. For example, the authors of [23] create representations that show what input yields a given activation, while by occluding parts of the input they reveal which regions are important for the classification. While these methods concentrate on explaining what a CNN has learned, similar techniques can be applied to RNNs. In [19], it is demonstrated that LSTM cells can learn to count the number of characters in a sentence, whereas cells that retain their state for long time periods can be located by examining the activation of the forget gates.

Considering how valuable these insights are for image classification and language processing tasks, it is expected that similar outcomes can be achieved for the classification of sEMG recordings of hand movements. Therefore, in this section an attempt is made to explain the results of some of the aforementioned methods with regard to sEMG data and gesture recognition.

The simplest way to interpret what a neural network has learned is to visualize the network weights and particularly the filters of a CNN (Figure 5). This helps to identify interesting patterns that the network tries to match in the input. Another benefit of this visualization is that it shows whether the network has converged (there is some structure in the filters) or not (the filters look random). Considering sEMG signals, the interpretation depends on what each input dimension corresponds to. For example, if the width and height dimensions are matched to a 2D electrode grid, then a 3×3 filter may indicate the direction of muscle action potentials. In case of $t \times c$ input, a filter can detect

the electrodes' transition from negative to positive values.

Another visualization is occlusion sensitivity [23], which consists of observing classification scores while occluding (i.e. setting the values to a constant) parts of the input. As a result, important regions for the classification of a specific image, yield a lower accuracy when occluded. In the case of sEMG inputs, an occlusion map can be generated if each electrode channel ($t \times c$ input) or rows/columns of electrodes (2D grid input) are occluded one at a time. This can reveal whether for a given gesture the network assigns more weight to a specific electrode. In addition, for gestures based on the activation of a few muscles, this method can verify whether the network has learned this correlation or not. An example of occlusion map can be seen in Figure 6.

Sensitivity analysis based on saliency maps [27] has been widely used for interpreting neural networks. In this method, partial derivatives measure how much small local changes in the pixel value affect the network output. Therefore, it can be used to highlight input regions that cause the most change in the output. It is expected that these pixels correspond to the object location in the image. Regarding sEMG, this method may show what input characteristics the network thinks are important for the classification of the input sEMG as a specific gesture. The application of this method to sEMG is shown in Figure 7.

The application of all these methods to the network proposed in [5] is presented in the next section.

5.3 Application to Ninapro Dataset

This section comprises an application of the above-mentioned evaluation schemes and neural network visualizations. Specifically, these methods are applied to the CNN proposed in [5] and the simple recurrent network with LSTMs (Table 5). In addition, the augmentation techniques presented in Section 3 are compared using a Wilcoxon signed-rank test. The neural networks are implemented using the Keras [28] API for Python, whereas the CNN visualizations are made with the keras-vis [29] tool. The implementation code is available at `https://github.com/ptsinganos/deep_emg_dsp_chapter`.

The dataset used for the following experiments is the Ninapro-DB1 (Table 6). In this dataset, sEMG signals are captured with an 1×10 array of electrodes, where the first 8 electrodes are equally spaced around the forearm, and the other two are placed on the main activity spots of finger flexor and extensor muscles respectively [30]. For the intra-subject evaluation of the network proposed in [5] (AtzoriNet) and the simple LSTM of Table 5 (LstmNet), repetitions 1, 2, 4, 6, 8, and 9 are used for training, while repetitions 3, and 10 for validation. On the other hand, the inter-subject evaluation is performed only for the AtzoriNet using repetitions 4, and 6 for training, while repetitions 1, 2, 3, 4, 6, 8, 9, and 10 for validation. In addition, the following augmentation techniques are compared: additive noise (AN), time-warping (TW), magnitude-warping (MW), and electrode shift (ES).

Table 7 shows the intra-subject evaluation for the AtzoriNet. In addition to the classification accuracy of a single sEMG segment (i.e. $15 \times 10 \times 1$ image), the

Table 6: The Ninapro dataset

Dataset	Subjects	Movements ($\#G$)	Electrodes	Sampling (Hz)
DB1	27	53	$1 \times 8 + 2$	100
DB2	40	50	$1 \times 8 + 4$	2000
DB3	11	50	$1 \times 8 + 4$	2000
DB4	10	53	$1 \times 8 + 4$	2000
DB5	10	53	2×8	200

accuracy of classifying a complete gesture, measured as the majority voting of the corresponding segment predictions, is reported in parentheses. As we can see there is a big difference between the two classification metrics. The reason is that during the recording there is a gradual transition between rest, gesture and rest, in contrast to the discrete changes of the gesture labels. Consequently, accuracy is lower during the transition periods where the change in movement is not yet clearly evident from the input EMG signal [16]. In addition, the small temporal length of the input (150ms) contributes to this discrepancy, since during the transitions many of these segments look similar between different gestures.

Regarding the comparison of the augmentation schemes, a simple additive Gaussian noise improves the accuracy by almost 2%, whereas the addition of TW and MW do not provide any statistical improvement. However, when only one of the TW and MW is applied the accuracy is improved by 3% compared to training without augmentation. This indicates that augmentation used in other related tasks such as, motion recognition with IMUs, may be helpful for the classification of sEMG. Therefore, different combinations of these methods without AN should be examined.

More insight about the network is gained with the following visualizations. A filter visualization of the first layer can show how the network looks at the raw data. In AtzoriNet the first layer contains 1×10 filters, i.e. filters that look at the relation between the sEMG channels. From Figure 5, we can see that there are filters that favor specific electrode combinations, whereas others attribute

Table 7: Intra-subject evaluation for the network of Atzori [5]

Augmentation	Validation accuracy[1]	Wilcoxon test[2]
none	0.6027 (0.8637)	-
AN	0.6200 (0.8812)	$p < 0.01$
TW	0.6320 (0.8756)	$p < 0.01$
MW	0.6331 (0.8735)	$p < 0.01$
AN-TW-MW	0.6196 (0.8753)	$p < 0.01$ ($p > 0.01$, $p < 0.01$, $p < 0.01$)

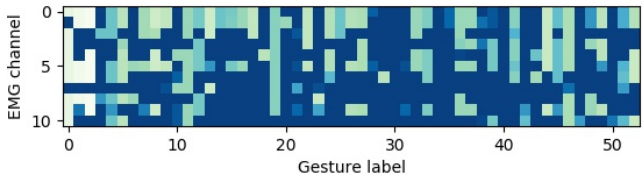
[1] The value in parentheses is the average accuracy using voting over the duration of gesture
[2] p-value of the Wilcoxon test. The values in parentheses correspond to the following comparisons: AN and AN-TW-MW, TW and AN-TW-MW, MW and AN-TW-MW.

almost equal weights to each electrode channel. In addition, the structure of the filters shows that the network has converged.

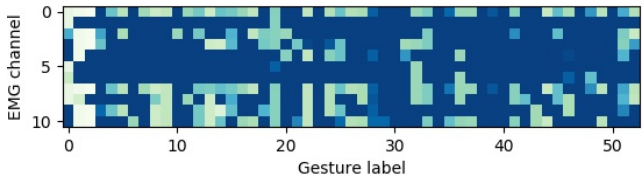
While the filter weights show what the network searches for in the input, an analysis of the occlusion sensitivity should reveal which portion of the input contains important information for a specific class. Figure 6 shows the occlusion sensitivity map generated by occluding (setting to '0' or '1') the sEMG values of each channel one at a time. Occlusion with '0' of channels 9 and 10 has a negative impact to the classification of almost all the gestures, whereas less gestures are affected when the occlusion is performed with '1'. This behavior is expected since the sEMG signal values in these regions indicate a high muscle activity for a wide range of gestures. An interesting observation can be made about the occlusion of electrodes 1-6. An occlusion with '0' of these electrodes has a smaller effect than the occlusion with '1'. This indicates that the values from these electrodes are close to zero.

The last visualization we will investigate is the saliency analysis. This method highlights the input regions that when their values is slightly changed, the class labels is affected the most. Figure 7 shows the average saliency map for the middle 1500ms segment (i.e. average of 10 segments) of each gesture.

Figure 5: The 32 learned 1×10 filters of the first convolutional layer of AtzoriNet [5]. Lighter color shades correspond to higher values. From bottom to top the values correspond to channels 1 to 10.



(a) Occlusion with '0'



(b) Occlusion with '1'

Figure 6: Occlusion sensitivity for the AtzoriNet [5]. The first row is the accuracy per gesture without occlusion. Subsequent rows correspond to the occlusion of each channel. Lighter color shades denote higher accuracy.

Only the middle segment of the gestures is used in order to avoid any misinterpretations from the analysis of transient periods. We can observe that almost all the gestures are influenced by the changes of the values of the first electrodes, whereas there are hand movements (e.g. '14') where for adjacent channels the behavior of the network is different. Furthermore, only a few gestures (those with labels '9', '21', '34', '43') are greatly affected by changes in the last two channels. This means that the electrodes at the main activity spots of the finger flexors/extensors capture useful information for the correct classification of these gestures.

The inter-subject evaluation for the AtzoriNet is given in Table 8. It is obvious from the results that training a single network with data from different subjects is indeed very difficult. Therefore, techniques based on transfer learning and domain adaptation can improve the performance of the network in this scenario. For example, in [6] the parameters of the Batch Normalization layers are adapted to each specific user during a pre-training phase with unlabeled data, whereas in [9], a source network trained with all available data shares information with user specific networks.

The evaluation of the LstmNet is shown in Table 9. The intra-subject validation accuracy for two versions of the LstmNet and with/without augmentation,

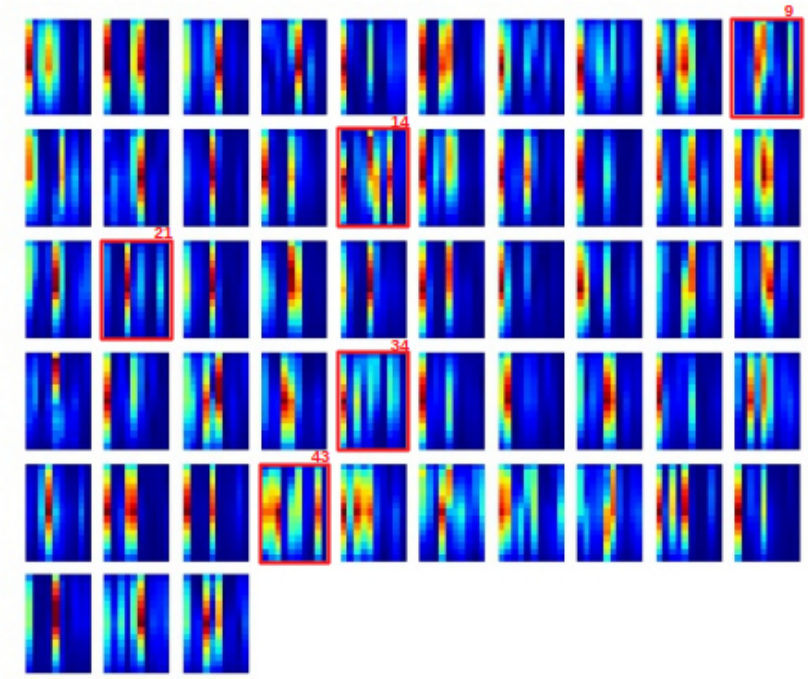


Figure 7: Saliency maps for the middle segment of each gesture. From top to bottom and left to right the images correspond to gesture labels 0 ('rest') to 52.

Table 8: Inter-subject evaluation for the network of Atzori [5]

Augmentation	**Validation accuracy**[1]	**Wilcoxon test**[2]
AN	0.0754 (0.0995)	-
AN-ES	0.0848 (0.1068)	$p < 0.01$

[1] The value in parentheses is the average accuracy using voting over the duration of gesture
[2] p-value of the Wilcoxon test.

Table 9: Intra-subject evaluation for the simple RNN of Table 5

Model	**Augmentation**	**Validation accuracy**	**Wilcoxon test**[1]
LSTM - 1 layer	none	0.0855	-
LSTM - 1 layer	AN-TW-MW	0.1879	$p < 0.05$
LSTM - 2 layers	none	0.1109	-
LSTM - 2 layers	AN-TW-MW	0.1028	$p < 0.05$

[1] p-value of the Wilcoxon test.

is given. In every case, the classification accuracy is pretty low compared to what is achieved by a CNN. The main reason why the accuracy of the LstmNet is low, is the fact that the classification is applied only after the last timestep. Therefore, architectures that take into account the output at each timestep (e.g. attention model) should perform better. Nevertheless, the augmentation has a positive impact on the single layer LSTM, by almost 10%, which should be expected considering the small size of the training dataset. In addition, there is a small performance improvement when using a deeper LSTM. However, the performance is reduced when the same augmentation is applied.

6 Conclusions

In this chapter, we have presented a basic Deep Learning methodology for sEMG-based gesture recognition. We started from the basics of sEMG acquisition and the parameters that influence the quality of the signal, to data preparation techniques, including signal processing and data augmentation. Typical neural network architectures based on convolutional and recurrent layers were presented, while their application to gesture recognition was discussed. Methods to evaluate the performance of a neural network were explained. Based on results from the domain of image classification, network visualization techniques that try to shed light on the inner-workings of neural networks were proposed. Finally, the steps of this methodology were illustrated by means of two examples applied on a publicly available dataset.

The evaluation of the CNN and the RNN models performed in Section 5 provided useful insight into the performance of the two networks. In particular, it was proved that, under the same optimization settings, CNNs are more suitable for the task of sEMG-based gesture recognition. In addition, the visualizations

for the CNN model can be proved a useful tool, since they help explain the decisions made by the network during inference. Therefore, should be an integral part of DL pipeline for hand movement classification.

References

[1] M.J. Cheok, Z. Omar, and M.H. Jaward, "A review of hand gesture and sign language recognition techniques," *International Journal of Machine Learning and Cybernetics*, vol. 0, Aug 2017.

[2] E. Scheme and K. Englehart, "Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use," *The Journal of Rehabilitation Research and Development*, vol. 48, no. 6, pp. 643–659, 2011.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, 2016.

[4] R. Merletti and D. Farina, *Surface Electromyography : Physiology, Engineering, and Applications*, John Wiley & Sons, Inc., Hoboken, New Jersey, Apr 2016.

[5] M. Atzori, M. Cognolato, and H. Müller, "Deep Learning with Convolutional Neural Networks applied to electromyography data: A resource for the classification of movements for prosthetic hands," *Frontiers in Neurorobotics*, vol. 10, Sep 2016.

[6] Y. Du, W. Jin, W. Wei, Y. Hu, and W. Geng, "Surface EMG-based inter-session gesture recognition enhanced by deep domain adaptation," *Sensors*, vol. 17, no. 3, Feb 2017.

[7] W. Geng, Y. Du, W. Jin, W. Wei, Y. Hu, and J. Li, "Gesture recognition by instantaneous surface EMG images," *Scientific Reports*, vol. 6, no. 36571, Nov 2016.

[8] C. Amma, T. Krings, J. Böer, and T. Schultz, "Advancing muscle-computer interfaces with high-density electromyography," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, New York, NY, USA, 2015, CHI '15, pp. 929–938, ACM.

[9] U. Côté-Allard, C. Latyr Fall, A. Drouin, A. Campeau-Lecours, C. Gosselin, K. Glette, F. Laviolette, and B. Gosselin, "Deep Learning for electromyographic hand gesture signal classification by leveraging transfer learning," *ArXiv e-prints*, Jan 2018.

[10] P. Konrad, "The ABC of EMG," *A practical introduction to kinesiological electromyography*, vol. 1, pp. 30–35, 2005.

[11] M. B. I. Reaz, M. S. Hussain, and F. Mohd-Yasin, "Techniques of EMG signal analysis: detection, processing, classification and applications," *Biological Procedures Online*, vol. 8, no. 1, pp. 11–35, Dec 2006.

[12] X. Zhai, B. Jelfs, R. Chan, and C. Tin, "Self-recalibrating surface EMG pattern recognition for neuroprosthesis control based on Convolutional Neural Network," *Frontiers in Neuroscience*, vol. 11, pp. 379–390, Jul 2017.

[13] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data Augmentation for Time Series Classification using Convolutional Neural Networks," in *Proceedings of 2nd ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, Riva del Garda, Italy, Sep 2016.

[14] T.T. Um, F.M.J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," in *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, New York, NY, USA, 2017, ICMI 2017, pp. 216–220, ACM.

[15] W. Wei, Y. Wong, Y. Du, Y. Hu, M. Kankanhalli, and W. Geng, "A multi-stream Convolutional Neural Network for sEMG-based gesture recognition in muscle-computer interface," *Pattern Recognition Letters*, Dec 2017.

[16] M. Atzori et al., "Characterization of a benchmark database for myoelectric movement classification," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 1, pp. 73–83, Jan 2015.

[17] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network training by reducing internal covariate shift," *ArXiv e-prints*, Feb 2015.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent Neural Networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[19] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and Understanding Recurrent Networks," *ArXiv e-prints*, Jun 2015.

[20] S. Hochreiter and Schmidhuber U., "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] C. Colah, "Understanding LSTM Networks," `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015, Accessed: 2018-06-30.

[22] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *ArXiv e-prints*, Mar 2018.

[23] R. Zeiler, M.and Fergus, "Visualizing and Understanding Convolutional Networks," *ArXiv e-prints*, Nov 2013.

[24] G. Montavon, S. Bach, A. Binder, W. Samek, and Müller. K., "Explaining NonLinear Classification Decisions with Deep Taylor Decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, May 2017.

[25] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and Müller. K., "Evaluating the Visualization of What a Deep Neural Network Has Learned," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 11, pp. 2660–2673, Nov 2017.

[26] Q. Zhang and S. Zhu, "Visual Interpretability for Deep Learning: a Survey," *ArXiv e-prints*, Feb 2018.

[27] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," *ArXiv e-prints*, Dec 2013.

[28] F. Chollet et al., "Keras," `https://keras.io`, 2015, Accessed: 2018-06-30.

[29] R. Kotikalapudi et al., "keras-vis," `https://github.com/raghakot/keras-vis`, 2017, Accessed: 2018-06-30.

[30] M. Atzori, A. Gijsberts, C. Castellini, B. Caputo, A.G.M. Hager, S. Elsig, G. Giatsidis, F. Bassetto, and H. Müller, "Electromyography data for non-invasive naturally-controlled robotic hand prostheses," *Scientific Data*, vol. 1, no. 140053, 2014.