



Hilbert sEMG data scanning for hand gesture recognition based on deep learning

Panagiotis Tsinganos^{1,2} · Bruno Cornelis^{2,3} · Jan Cornelis² · Bart Jansen^{2,3} · Athanassios Skodras¹

Received: 28 February 2020 / Accepted: 16 June 2020 / Published online: 7 July 2020
© The Author(s) 2020

Abstract

Deep learning has transformed the field of data analysis by dramatically improving the state of the art in various classification and prediction tasks, especially in the area of computer vision. In biomedical engineering, a lot of new work is directed toward surface electromyography (sEMG)-based gesture recognition, often addressed as an image classification problem using convolutional neural networks (CNNs). In this paper, we utilize the Hilbert space-filling curve for the generation of image representations of sEMG signals, which allows the application of typical image processing pipelines such as CNNs on sequence data. The proposed method is evaluated on different state-of-the-art network architectures and yields a significant classification improvement over the approach without the Hilbert curve. Additionally, we develop a new network architecture (MSHilbNet) that takes advantage of multiple scales of an initial Hilbert curve representation and achieves equal performance with fewer convolutional layers.

Keywords Hilbert curve · Hand gesture recognition · sEMG · Electromyography · Classification · CNN · Deep learning · Multi-scale

1 Introduction

The problem of gesture recognition is encountered in many applications including human computer interaction [38], sign language recognition [10], prosthesis control [9] and rehabilitation gaming [8, 34]. Signals generated from the electrical activity of the forearm muscles, which can be

recorded with surface electromyography (sEMG) sensors, contain useful information for decoding muscle activity and hand motion [16].

Machine learning (ML) classifiers have been used extensively for determining the type of hand motion from sEMG data. A complete pattern recognition system based on ML consists of data acquisition, feature extraction, classifier definition and inference from new data. For the classification of gestures from sEMG data, electrodes attached to the arm and/or forearm acquire the sEMG signals, and features such as mean absolute value (MAV), root mean square (RMS), variance, zero crossings and frequency coefficients are extracted and then fed as input to classifiers like k-nearest neighbors (k-NNs), support vector machine (SVM), multilayer perceptron (MLP) or random forests [40].

Over the past years, deep learning (DL) models have been applied with great success in sEMG-based gesture recognition. In these approaches, sEMG data are represented as images and a convolutional neural network (CNN) is used to determine the type of gesture. A typical CNN architecture consists of a stack of convolutional and pooling layers followed by fully connected (i.e., dense)

✉ Panagiotis Tsinganos
panagiotis.tsinganos@ece.upatras.gr

Bruno Cornelis
bcorneli@etrovub.be

Jan Cornelis
jpcornel@etrovub.be

Bart Jansen
bjansen@etrovub.be

Athanassios Skodras
skodras@upatras.gr

¹ Department of Electrical and Computer Engineering, University of Patras, 26504 Patras, Greece

² Department of Electronics and Informatics, Vrije Universiteit Brussel, 1050 Brussels, Belgium

³ IMEC, 3001 Leuven, Belgium

layers and a softmax output. In this way, CNNs transform the input image layer by layer, from the pixel values to the final classification label.

The application of DL methods can also favor the performance of sEMG interfaces used in rehabilitation. There is a large body of literature about the utilization of sEMG devices in rehabilitation and myoelectric control [17, 28, 36, 37]. A common limitation presented is the lack of classification robustness. To address this, recent studies provide evidence for significant performance improvement (with respect to classification accuracy and latency) achieved with DL approaches [6, 41, 47].

CNNs have made breakthroughs in feature extraction and image classification tasks in 2D problems. Yet, choosing a proper method to convert time-series into images that can be used as inputs to CNN models is not obvious. Among the methods proposed in the literature are the segmentation of multi-channel signals using windows and the application of 1D transformations such as the Fourier and wavelet transforms.

In this work, we extend our previous research [49] about the application of the Hilbert space-filling curve to represent sEMG signals as images. This type of curve is useful because it provides a mapping between 1D and d-dimensional spaces while preserving locality. This approach enables the application of image processing techniques on sequence data such as biomedical signals. In this case, CNN models are used to classify Hilbert curve images of sEMG signals for the problem of hand gesture recognition. The overview of the current work is given in Fig. 1. The main contributions presented in this paper are:

- a detailed performance comparison of the proposed locality-preserving Hilbert curve representation over well-known CNN models
- a comparison with the state-of-the-art (WeiNet [53]) in hand gesture recognition
- the development of a new lightweight model (MSHilb-Net) based on multiple scale Hilbert curves.

The remaining of the paper is organized as follows. Section 2 provides a literature review on gesture recognition approaches, as well as on applications of the Hilbert curve to classification tasks. In Sect. 3, the details of the proposed method and of the CNN architectures used for experimentation are given. Section 4 describes the experiments performed for the evaluation of the models, while the results followed by a discussion are presented in Sect. 5–6. Finally, a summary of the outcomes is given in Sect. 7 and Appendix 1 contains additional figures and tables.

2 Related work

Both typical ML approaches and DL practices have been employed to study the problem of sEMG-based hand gesture recognition. The first ML approach is presented in [25], where for the classification of four gestures a set of time-domain features is extracted from sEMG signals recorded with two electrodes. The authors of [7] achieve 97% accuracy in classifying three types of grasps using the RMS feature extracted from seven electrodes as input to an SVM classifier. A comparison of different types of EMG features and classifiers for the classification of 52 gestures from the Ninapro reference dataset [3, 4] is provided by

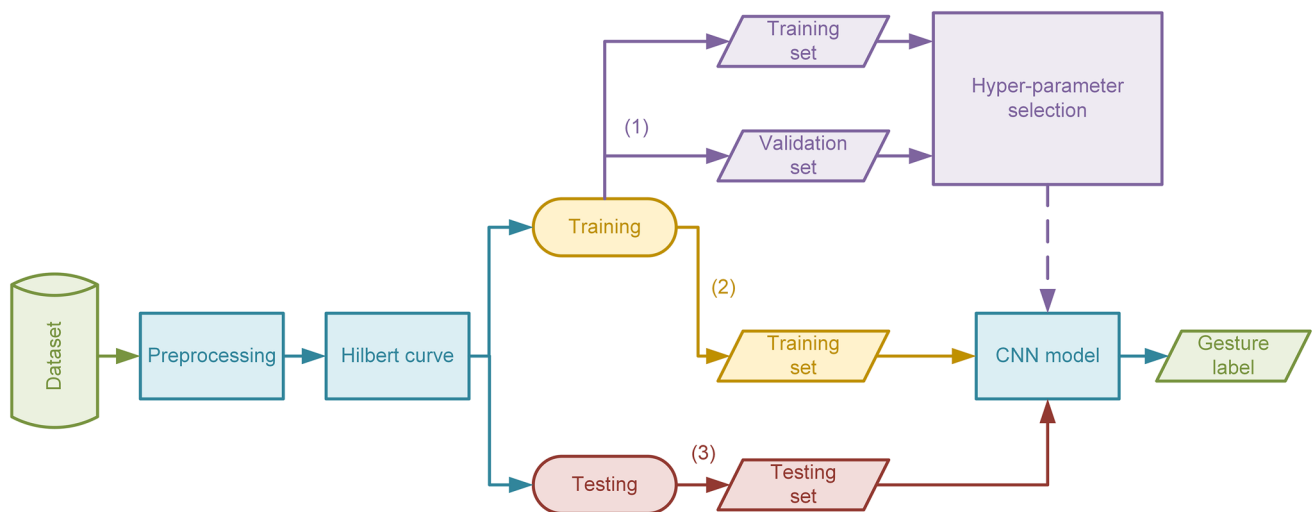


Fig. 1 The flowchart of the current work. sEMG data are preprocessed as in previous studies, and then the Hilbert curve mapping (Figs. 3, 4) is applied. During the training session (1), a set of hyper-parameters (Tables 3, 4, 5) is selected based on the performance on

the validation set, before optimizing (2) the weights of the CNN model (Figs. 6, 8, 9, 10, 11, 12), which is evaluated (3) on the testing data (Figs. 7, 12, 13, 14, Tables 6, 7, 8, 9, 10)

[3, 19, 30]. A random forest classifier and a combination of statistical and frequency domain features, i.e., MAV, histogram, wavelet and Fourier transform features, yield the best performance, an accuracy of 75%.

The literature for DL methods related to sEMG gesture recognition has been continuously increasing over the last years. In [39], the authors evaluate different configurations of RNNs and their results show that a classifier with a bidirectional recurrent layer composed of long short-term memory (LSTM) units followed by attention mechanism performs best in an application classifying 18 gestures from the Ninapro database. The authors of [46] use an unsupervised generative flow model to learn comprehensible features classified by a softmax layer that achieves about 64% accuracy on classifying 53 gestures. RNNs are important for sequence problems where successive inputs are dependent on each other. However, this is not totally true for EMG signals since they are inherently stochastic.

CNN models are the most commonly used DL approach for the task of gesture recognition based on sEMG. In [35], the authors develop a CNN for the categorization of six common gestures that improves the classification accuracy compared to SVM. The model of [2] consisting of convolutional and average pooling layers results in comparable performance to what was achieved using typical ML approaches. The results of our previous work [48] indicate that the use of max pooling rather than average pooling and the addition of dropout [44] layers between the convolutions can produce a 3% increase in accuracy (from 67 to 70%). The works of [18, 53] propose a few novelties compared to previous works not only in network structure but also in the way EMG signals are acquired. This is based on a high-density electrode array, which is considered an effective approach in myoelectric control [27, 33, 45]. Using instantaneous EMG images, the CNN model of [18] correctly classifies a set of eight hand movements with a rate of 89%, whereas the multi-stream CNN described in [53] achieves 85% accuracy on the Ninapro database. In their later works [22, 52], the authors of [53] propose a multi-view approach combining various sEMG representations, including FFT and traditional feature vectors, that achieves a classification improvement of about 3%.

Methods that deal with the adaptation of a pretrained network to new users have been developed as well. The work of [14] utilizes adaptive batch normalization (AdaBN) [31] to distinguish between subject-specific knowledge (normalization layers) and gesture-specific knowledge (convolutional layer's weights), whereas in [12] a method based on weighted connections between a network trained on one subject (source domain) and a network trained on a different subject (candidate target network) is presented. In addition, [12] compares methods of data augmentation for sEMG signals.

The properties of the Hilbert curve are well known and have been exploited in the past for diverse applications. The authors of [13, 29] employ the Hilbert curve to represent mammographic images as 1D vectors from which a combination of features is extracted in order to detect breast cancer. Similarly, the work of [11] transforms volumetric data into 2D and 1D representations, which are then processed efficiently by typical CNNs. Compared to processing the raw data directly, the method described in [11] reduces training time and can be used on data with an arbitrary number of channels. The performance of recurrent models, such as long short-term memory (LSTM) networks, in the detection of image forgeries depends on the sequence of the extracted image patches. In [5], the order by which image patches are fed into an LSTM is determined by the Hilbert curve in order to better preserve their spatial locality. In our work, the Hilbert curve is not applied for dimensionality reduction, rather it is utilized for presenting 1D sEMG signals as 2D images in a way similar to what is done in a few other studies. In the work of [54] that deals with the problem of DNA sequence classification, it was determined that long-term interactions between regions of the sequence are important for high classification accuracy. Instead of using very deep networks or larger filters, the Hilbert curve was used to map the DNA sequence into an image such that proximal elements stay close, while the distance between distant elements is reduced. In addition, the authors of [1] employ the same representation to improve the performance of a deep neural network that detects regions in the DNA sequence that are important for gene transcription.

3 Methods

3.1 Hilbert curve

A Hilbert curve (also known as a Hilbert space-filling curve), first described by the German mathematician David Hilbert in 1891, is a continuous fractal space-filling curve, i.e., a curve that passes through all the points of a d-dimensional space sequentially. Space-filling curves have been widely applied to tasks in data organization and compression. The Hilbert curve is known for being superior in preserving locality compared to alternatives [20, 32], such as the z-order and Peano curves. This means when two points that lie on a 1D line at a specific distance are mapped into 2D space with a space-filling curve, their new distance will be smaller if the Hilbert curve is used.

Following the notation found in [20], a discrete d-dimensional Hilbert curve of order k , denoted as H_k^d of length L^d , where $L = 2^k$, is a bijective mapping:

$$H_k^d : [L^d] \rightarrow [L]^d \quad (1)$$

where $[L] = \{0, \dots, L-1\}$. For the 2D case ($d=2$), a sequence x of length L^2 , $\{x\} = \{x_0, x_1, \dots, x_{L^2-1}\}$, is mapped into a 2D image y with dimensions $L \times L$, such that $y_{i,j} = x_l, \forall l \in \{0, \dots, L^2-1\}$, where $(i,j) = H_k^2(l)$. For the rest of the paper, we simply denote H_k^2 as H_k .

The Hilbert curve can be easily constructed in a recursive manner. Initially, the 2D plane is divided into four quadrants traversed according to a fundamental pattern, as shown in Fig. 2, that constitutes the first-order representation H_1 . Higher-order curves are produced by dividing existing sub-squares into four smaller ones that are connected by a pattern obtained by rotation and/or reflection of the fundamental pattern. A visualization of Hilbert curve traversals of the 2D space for orders $k = \{1, 2, 3\}$ is shown in Fig. 2, where the numbers correspond to the index within the sequence that is mapped to a specific pixel.

3.2 sEMG representation

In this work, the Hilbert curve is employed to transform multi-channel sEMG signals into 2D image representations. Firstly, the sEMG data of a hand gesture recorded by M electrodes are organized into small segments of length N . Therefore, the dimensions of the data are $N \times M$. Then, the mapping can be used in two ways: (1) across the time dimension, i.e., for each sEMG channel, map the time sequence into a 2D image, or (2) across the sEMG channels, i.e., for each time instant, map the values of the channels into a 2D image. Examples of these representations are shown in Fig. 3.

Usually in image applications, the input of a CNN is either an image with one channel (grayscale) or an image with three channels (RGB). In our approach, the image depth corresponds to either the number of sEMG electrodes or the duration of the sEMG segment.

What our approach actually does, is to reshuffle the spatiotemporal samples of a 1D image (Fig. 3d) in a multi-channel image with lower dimensions (Fig. 3e, f). The result of constructing each channel through the Hilbert

scanning is to extend either the time neighborhoods (e) or the electrode neighborhoods (f). So, only one of the domains benefits from the Hilbert curve: the time domain in (e) where the Hilbert curve is applied on the rows of (d), and the electrode (spatial) domain in (f) where the Hilbert curve is applied on the columns of (d).

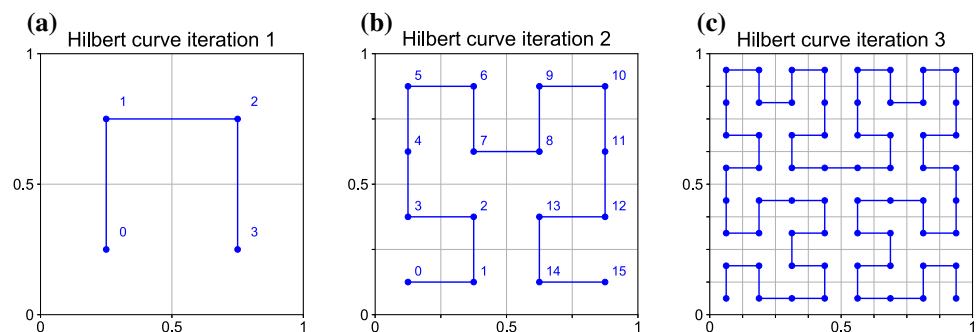
The computation of the proposed representations using the Hilbert space-filling curve requires only bitwise operations performed in constant time [21]. Then, the algorithm that computes the mapping between 1D and 2D takes $\mathcal{O}(\log_2 K)$, where $K = N$ (Sect. 3.2.1) or $K = M$ (Sect. 3.2.2) is the length of the projected sequence. In addition, since the mapping from 1D to 2D is the same for all the generated images, it is computed only once and then used as a look-up table. Thus, considering that K is limited to $K_{max} = 64$, the computational overhead is negligible compared to training the CNN.

We evaluate the Hilbert curve representations of sEMG signals across five CNNs and compare their performance to the baseline approach (i.e., Hilbert mapping is not applied).

3.2.1 Hilbert in time

The application of the Hilbert mapping across the time dimension (HilbTime) consists of the following steps (Fig. 4). Given a single electrode sEMG sequence of length N , a 2D Hilbert representation is achieved with maximum dimensions $L \times L$, where $N \leq L^2$ and L is a power of two, i.e., $L = 2^k$. If there are M sEMG electrodes, this process is repeated for every electrode, and the outputs are stacked into a M -channel image, i.e., an image with dimensions $L \times L \times M$. For example, an sEMG segment of 10 electrodes with 64 samples is mapped into an $8 \times 8 \times 10$ image. The output image, y , is initialized with zeros, and then for every electrode m and every timestep n , image coordinates (i, j) are generated from the timestep n such that the image value at position (i, j, m) equals the signal value at timestep n of electrode m . It is important to note that when sequence segments of length smaller than L^2 are used, the final image can be cropped in order to remove rows and columns with only zeros.

Fig. 2 The first three iterations of the Hilbert curve



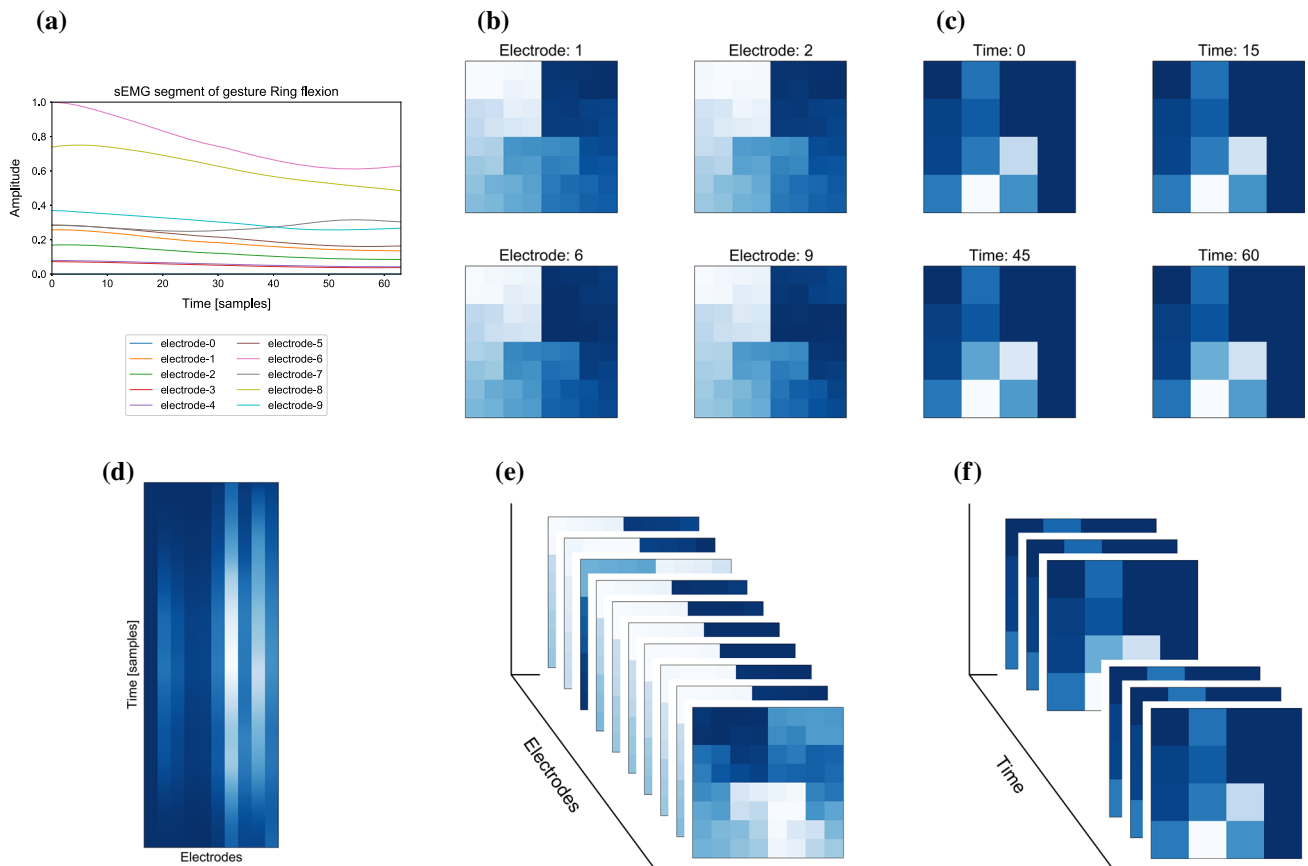


Fig. 3 The application of the Hilbert curve mapping to sEMG data. **a** A 64-samples segment of sEMG signal, **b** the ‘HilbTime’ (Hilbert across time) representation of electrodes $m = \{1, 2, 6, 9\}$ (image size 8×8), **c** the ‘HilbElect’ (Hilbert across electrodes) representation at time instants $n = \{0, 15, 45, 60\}$ (image size 4×4), **d** the ‘Baseline’

image representation (image size 64×10), **e** the ‘HilbTime’ 3D sEMG representation (image size $8 \times 8 \times 10$) and **f** the ‘HilbElect’ 3D sEMG representation (image size $4 \times 4 \times 64$). In (c, f), there are less electrodes than pixel dimensions; thus, the last six pixels on the right of the images are zeros

(a)

Algorithm: Time dimension (HilbTime)

Input: x , 2D array of size $N \times M$, where $N \leq L^2$
Output: y , 3D array of size $L \times L \times M$
 $hc \leftarrow H_{\log_2 L}$;
 $y \leftarrow \text{zeros}(L, L, M)$;
for $m \leftarrow 0$ **to** $M - 1$ **do**
 for $n \leftarrow 0$ **to** $N - 1$ **do**
 $i, j \leftarrow hc(n)$;
 $y[i, j, m] \leftarrow x[n, m]$;
 end
end

(b)

Algorithm: Electrode dimension (HilbElect)

Input: x , 2D array of size $N \times M$, where $M \leq L^2$
Output: y , 3D array of size $L \times L \times N$
 $hc \leftarrow H_{\log_2 L}$;
 $y \leftarrow \text{zeros}(L, L, N)$;
for $n \leftarrow 0$ **to** $N - 1$ **do**
 for $m \leftarrow 0$ **to** $M - 1$ **do**
 $i, j \leftarrow hc(m)$;
 $y[i, j, n] \leftarrow x[n, m]$;
 end
end

Fig. 4 The steps of the two approaches: **a** Hilbert in time and **b** Hilbert in electrodes

3.2.2 Hilbert in electrodes

The Hilbert mapping is applied across the sEMG electrodes (HilbElect) (Fig. 4). Specifically, if the number of sEMG electrodes is M , then $M \leq L^2$, where $L = 2^k$. The Hilbert mapping is applied at every time instant of the sequence

resulting in an image with dimensions $L \times L \times N$. For example, an sEMG segment of 16 electrodes with 20 samples is mapped into a $4 \times 4 \times 20$ image. The output image, y , is initialized with zeros, and then for every electrode m and every timestep n , image coordinates (i, j) are generated from the electrode index m such that the

image value at position (i, j, n) equals the sEMG value at timestep n of electrode m . As in the previous case, if the number of electrodes is less than L^2 the final image can be cropped.

3.3 Network architectures

In this work, we evaluate our method on the state-of-the-art CNN architecture for the problem of hand gesture recognition (WeiNet [53]), as well as models based on architectures typically found in image tasks, such as VGGNet [42], DenseNet [24] and SqueezeNet [26]. Since these models were defined for image tasks where the dimensionality of the data is bigger, modifications were needed to adapt these architectures to sEMG data. In addition, inspired by the multi-scale dense networks (MSDNet) [23], we propose the application of a similar architecture, which we name multi-scale Hilbert network (MSHilbNet), where the generation of Hilbert curve representations of multiple orders is an inherent feature of the topology. The number of parameters for the model architectures is presented in Table 1, while details of each architecture are given next.

3.3.1 WeiNet [53]

In [53], the state-of-the-art model in hand gesture recognition, the authors propose a shallow network that consists of two convolutional and two locally connected layers that do not decrease the spatial resolution of the input. This architecture is applied for every EMG channel, and the corresponding feature maps are merged into a single feature vector via concatenation. Then, this feature vector passes through a series of fully connected layers followed by a softmax classifier. We include this CNN model in our investigation since it achieves the highest accuracy among the works presented in Sect. 2 on the benchmark dataset (i.e., Ninapro [3]). A graphical representation of the model is shown in Fig. 8.

3.3.2 VGGNet [42]

The initial VGGNet [42], invented by VGG (Visual Geometry Group) from University of Oxford, won the localization task in ILSVRC 2014. It consists of 16

convolutional layers and follows a very uniform architecture: It uses 3×3 convolutions only, while every third convolutional layer is followed by a pooling operation. In this work, we use convolutional blocks (d) that consist of two 3×3 convolutional layers with rectified linear unit (ReLU) activations and a pooling layer (pl). The output class label is obtained through a global pooling operation followed by a dense layer with softmax activation. A graphical representation of the model is shown in Fig. 9.

3.3.3 DenseNet [24]

The DenseNet [24] was invented by Cornwell University, Tsinghua University and Facebook AI Research. The architecture is based on the idea of creating short paths from early layers to later layers, thus ensuring maximum information flow between the network layers. An advantage of this is that the network can be compact with less parameters, since each layer receives feature maps from all preceding layers. In this work, the building block (d) of the architecture contains three densely connected groups of batch normalization (BN), ReLU and 3×3 convolutional layer, while a transition layer made of BN, ReLU, 1×1 convolution and 2×2 pooling (pl) connects two successive dense blocks. The classifier contains a 1×1 convolution, ReLU, global pooling layer and a softmax activation. A graphical representation of the model is shown in Fig. 10.

3.3.4 SqueezeNet [26]

The SqueezeNet [26] was invented by UC Berkeley and Stanford University in an effort to achieve equivalent accuracy with smaller CNN architectures. The main component of this model is the ‘fire module.’ This is based on 1×1 convolutions (squeeze layer) to reduce the depth of the feature maps before the application of 3×3 convolutions (expand layer) that increase the feature depth. In this work, three fire modules followed by a 2×2 pooling layer (pl) comprise the main building block (d) of the architecture, while the classifier contains a 1×1 convolution, ReLU, global pooling layer and a softmax activation. A graphical representation of the model is shown in Fig. 11.

3.3.5 Multi-scale Hilbert network (MSHilbNet)

Inspired by the MSDNet [23], we propose the multi-scale Hilbert network (MSHilbNet), a new multi-scale Hilbert curve approach for the problem of gesture recognition. With this model, we investigate the utilization of Hilbert curve representations of multiple scales (i.e., multiple orders of the Hilbert curve), where coarser scales can be constructed via downsampling. We show here that downsampling with a pooling layer of non-overlapping 2×2

Table 1 Number of parameters for the model architectures

| Model | Parameters |
|-----------------|------------|
| WeiNet [53] | 7,277,625 |
| VGGNet [42] | 302,389 |
| DenseNet [24] | 126,618 |
| SqueezeNet [26] | 58,853 |
| MSHilbNet | 62,453 |

kernels ($\text{kernel size} = \text{stride} = 2 \times 2$) retains the pattern of the Hilbert curve (Fig. 5), i.e., the output of this operation is the Hilbert curve representation of the subsampled signal.

Given an image y that is the Hilbert representation of order k of the 1D sequence x of length L^2 (i.e., $y_{ij} = x_l$, where $\{x\} = \{x_0, x_1, \dots, x_{L^2-1}\}$, and $(i, j) = H_k(l), \forall l \in \{0, \dots, L^2 - 1\}$), then the following holds for the subsampled representation $z = \text{pool}_2^2(y)$:

$$x' = \text{pool}_4^1(x) \quad (2)$$

$$z_{i',j'} = x'_l \quad (3)$$

$$(i', j') = H_{k-1}(l), \forall l \in \{0, \dots, L^2/4 - 1\} \quad (4)$$

where $\text{pool}_2^2()$ is the 2D pooling operation with kernel size and stride equal to 2×2 , $\text{pool}_4^1()$ is the 1D pooling with kernel size and stride equal to 4, $\{x'\} = \{x'_0, x'_1, \dots, x'_{L^2/4-1}\}$.

With this tool, we can create a network architecture, shown in Fig. 6, that exploits multiple resolutions of the Hilbert representation obtained via a pooling (pl) operation. As in MSDNet, regular convolutions increase the depth (d) of the architecture along a resolution, while we adopt strided convolutions to carry information from a higher resolution to a lower resolution (s). Regular convolutions consist of 3×3 convolutions and ReLU activations, whereas the strided convolutions use 2×2 filters. Finally, a single output label is obtained by merging the outputs (o) of the intermediate classifiers that consist of 1×1 convolution, ReLU, global pooling layer and a softmax activation. The implementation code will be available at <https://github.com/DSIP-UPatras/sEMG-hilbert-curve>.

4 Experiments

4.1 Dataset

The evaluation of the models was performed on the first dataset of the Ninapro database [3]. This consists of sEMG

recordings of 27 healthy subjects that repeat all 52 gestures 10 times with a relax period between repetitions. The hand movements, which cover the majority of movements found in activities of daily living and rehabilitation exercises, can be grouped into three categories: (1) basic finger movements, (2) isometric, isotonic hand configurations and wrist movements and (3) grasps and functional movements. EMG signals are measured using 10 electrodes, of which eight are equally spaced around the forearm and the remaining are attached on main activity spots of the large flexor and extensor muscles of the forearm [3].

The sEMG signals are preprocessed with a low-pass filter as in previous studies that involve the Ninapro database [2, 18, 48]. Then, training data are augmented by adding Gaussian noise with a signal-to-noise ratio (SNR) equal to 25 dB. In addition, sEMG signals are augmented with the magnitude-warping method described in [50, 51]. As a last step, sEMG signals from the 10 channels are segmented into overlapping windows of length N with a step of 10ms (1 sample) and organized into $N \times 10$ arrays.

In the following experiments, window segments of 16, 32 and 64 samples (160 ms, 320 ms and 640 ms, respectively) were used. A validation experiment showed that a window size of 64 samples performs the best, while we included the results for shorter segments considering the guidelines for real-time myoelectric control applications [15, 43]. In the case of $N = 16$, the training set (Fig. 1 path (2)) contains on average 266K instances with a standard deviation of 17K, while the corresponding testing set (Fig. 1 path (3)) is $3.6K \pm 250$. We attribute this variance across the subjects to the fact that in the Ninapro dataset the duration of the gesture repetitions is not the same among the subjects. The train and test sizes for the remaining input configurations are shown in Table 2. Similar to other classification problems, the gesture labels are encoded with one-hot vectors of dimension equal to 53, i.e., the 52 hand gestures and the relax periods.

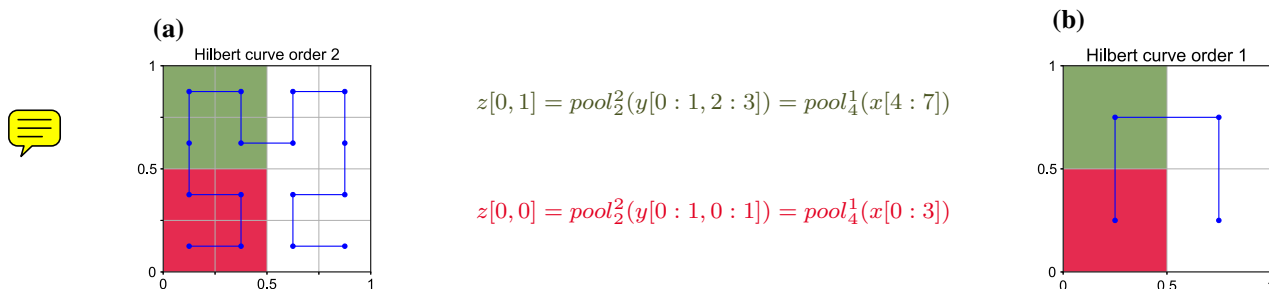


Fig. 5 Application of pooling operator to obtain lower-order Hilbert curve representations. In **b**, z is a first-order representation generated from a second-order representation y , shown in **a**

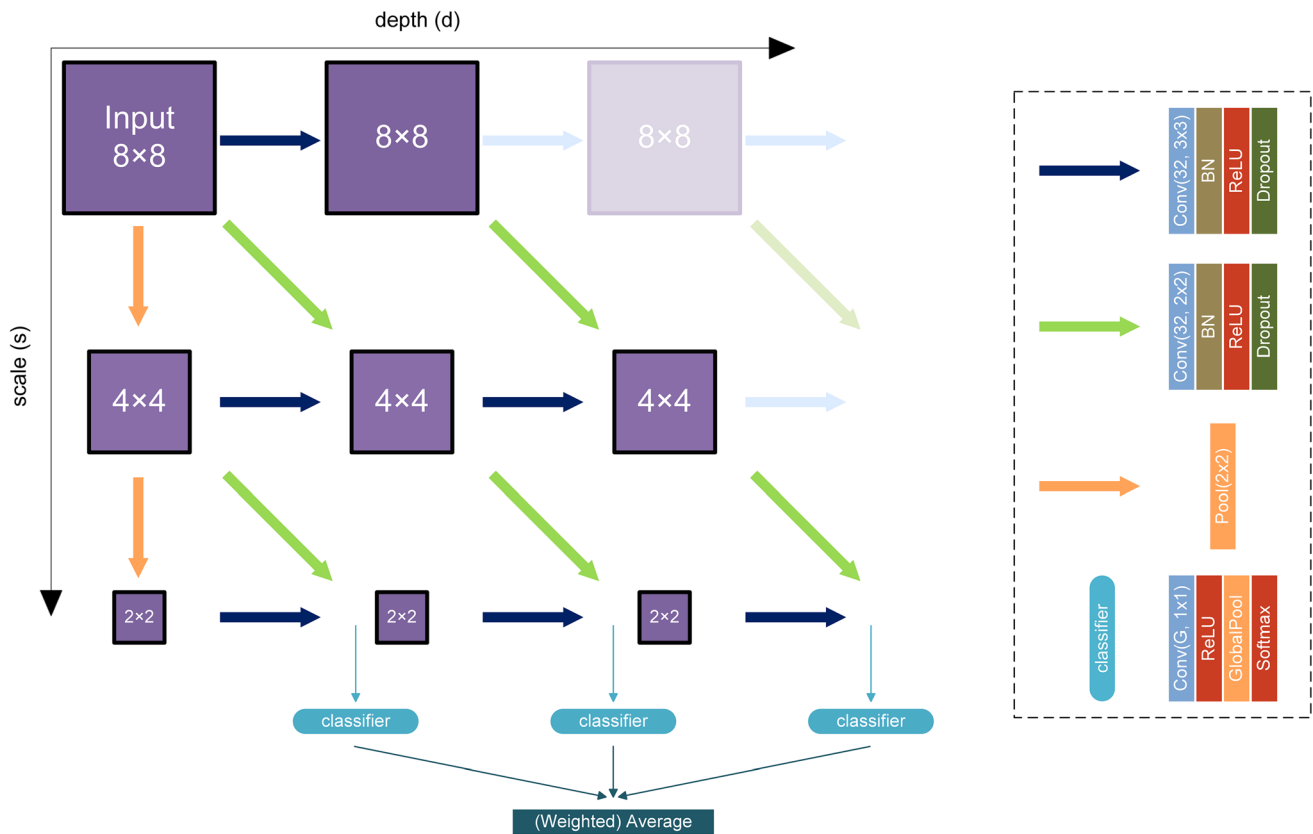


Fig. 6 The MSHilbNet model. Horizontal arrows correspond to regular convolutions between features of the same scale, while the diagonal arrows are strided convolutions that convey information to coarse scales. The initial scales are generated by a pooling operation

shown with vertical arrows. The dark shaded blocks correspond to a model with $s = [8 \times 8, 4 \times 4, 2 \times 2]$, $d = 3$, $o = [3, 2, 1]$. A detailed description of the operations that correspond to each type of arrow is given on the right of the figure

4.2 Evaluation of methods

The evaluation of the models is identical to existing works that have used the Ninapro dataset [2, 18, 48, 53]. Specifically, a new model is randomly initialized for each subject and trained (Fig. 1 path (2)) on data from seven repetitions (1, 3, 4, 6, 8, 9 and 10) and tested (Fig. 1 path (3)) on the remaining three (2, 5 and 7). As performance metrics, we use the accuracy, precision and recall averaged over all the subjects. An exception was made for the WeiNet which was trained and evaluated as in the original paper [53].

Prior to performing the evaluation, a model selection step (Fig. 1 path (1)) is required to determine the appropriate hyper-parameter values of the models. In the case of VGGNet, DenseNet, SqueezeNet and MSHilbNet, a validation set determined the depth (d) of the network and the type of pooling (pl), as well as the number of Hilbert scales (s) and output layers (o) of the MSHilbNet. The exact parameter space and the selected values are reported in Table 3. A set of ten randomly selected subjects was used. From the training set of each subject (Fig. 1 path (2)),

repetition number 6 is held out as a validation set (Fig. 1 path (1)). Then, a grid search is performed for the hyper-parameters in Table 3 where models are trained for every subject using six repetitions and evaluated on the validation data. The models with the best performance (i.e., highest accuracy for the validation set) are selected for further evaluation. At this stage, we consider input sEMG segments of length $N = 16$ (i.e., input dimensions 16×10) for the VGGNet, DenseNet and SqueezeNet, while for the MSHilbNet we use input segments of $N = 64$ (i.e., input dimensions $8 \times 8 \times 10$) since it allows to evaluate the performance of 1–3 scales.

The depth hyper-parameter, d , corresponds to the number of basic blocks that the network consists of. Therefore, in the case of VGGNet, the basic block is made of two convolutions, whereas in the DenseNet block there are three convolutions. The basic block of the SqueezeNet contains three fire modules, each one with two levels of convolutions. Finally, in MSHilbNet the number of convolutions depends on both the depth and the number of scales (e.g., for a number of scales equal to 3, every depth $d > 1$ adds to the network graph three regular and two strided convolutions).

Next, we describe the experimentation we followed to compare the performance of the proposed Hilbert curve mapping.

4.2.1 Baseline

As our baseline, we follow the approach where the Hilbert mapping is not used. Therefore, the $N \times 10$ arrays are fed into the CNN models as single-channel images ($N \times 10 \times 1$). For the window length N , we experiment with three values: 16, 32 and 64 samples using the models in Table 1.

4.2.2 Hilbert in time

In the case of the Hilbert mapping across the time dimension (HilbTime), the $N \times 10$ segments are organized into $M \times M \times 10$ images. For N values equal to 16, 32 and 64, the resulting image sizes are $4 \times 4 \times 10$, $8 \times 4 \times 10$ (8×8 cropped into 8×4 to remove zero columns) and $8 \times 8 \times 10$, respectively. The models in Table 1 were used.

4.2.3 Hilbert in electrodes

The Hilbert mapping across the sEMG channel dimension (HilbElect) is performed in a similar fashion. Given the number of channels $M = 10$, the $N \times 10$ segments are organized into $4 \times 4 \times N$ images. The pixels corresponding to the last six positions that the Hilbert curve traverses are set to zero. In this approach, we retain the spatial resolution constant due to the small number of available electrodes. For the window length, we experimented with $N = 16$, $N = 32$, and $N = 64$. As in the previous case, the models in Table 1 were used.

4.3 Model optimization

In the hyper-parameter selection step (Fig. 1 path (1)), the networks were trained using stochastic gradient descent (SGD) for 30 epochs with an initial learning rate of 0.1, halved every 10 epochs and a batch size of 1024. Due to convergence problems of the SqueezeNet with this learning schedule, it was trained with SGD optimizer and an initial learning rate of 0.1 that was reduced when the validation loss stopped decreasing. To avoid overfitting the networks due to the small training set, dropout layers were appended after convolutional layers with a forget rate of 0.3. In addition, weight decay regularization with a value of $l_2 = 0.0005$ was applied to all convolutional layers.

The final models were trained (Fig. 1 path (2)) with SGD for 60 epochs as in [49]. The WeiNet model was trained following the procedure described in the original paper [53].

The WeiNet was trained on a workstation with an Intel Xeon, 2.40 GHz (E5-2630v3) processor, 16 GB RAM and a Nvidia GTX1080, 8GB GPU using the MxNet tools for Python. The rest of the models were trained on a workstation with an Intel i9-7920X, 2.90 GHz processor, 128 GB RAM and a Nvidia RTX2080 Ti, 12GB GPU using the Keras and Tensorflow libraries for Python.

5 Results

The optimal hyper-parameter selection (Fig. 1 path (1)) for the different CNN models is based on the evaluations shown in Tables 4 and 5. The search space along with the selected values for each of the network architectures is presented in Table 3. The number of scales (s) and the output layers (o) hyper-parameters are used only by the MSHilbNet model. In Tables 4 and 5, we show the average accuracy and the standard deviation on the validation set (i.e., repetition 6) of 10 random subjects for every parameter combination across the four architectures, while a bold font denotes the best (i.e., higher accuracy) combination of parameters.

With the optimal values for the hyper-parameters, the networks and the Hilbert representations are evaluated next (Fig. 1 path (3)). The evaluation results of the models and the proposed Hilbert representations are presented in Tables 6, 7, 8, 9 and 10 for different window lengths, N . The metrics' values are given by the average and the standard deviation over all subjects in the dataset evaluated on the test set (i.e., repetitions 2, 5, and 7). For the VGGNet, DenseNet, SqueezeNet and MSHilbNet, the accuracy, precision and recall are calculated, whereas for the WeiNet, only the accuracy is shown since the code provided by the authors of [53] does not compute the other metrics. The columns correspond to the combination of a representation method (Baseline, HilbTime, HilbElect) and a window length N (16, 32, 64). It should be noted that for the WeiNet, the 'HilbElect' representation is not evaluated since this model architecture assumes that the last dimension of the input equals the sEMG electrode dimension. The accuracy curves during training and testing (Fig. 1 paths (2–3)) are shown in Fig. 12. Further, a radar chart in Fig. 7 evaluates other aspects of the investigated architectures when the Hilbert scanning is used.

Comparisons between the models and representation methods are statistically evaluated. A repeated measures analysis of variance (ANOVA) with the Greenhouse–Geisser correction to account for the violation of sphericity (Mauchly's Test of Sphericity indicated that the assumption of sphericity had been violated) is performed. This is followed by post hoc pairwise tests using the Bonferroni correction for multiple comparisons. Significance level was

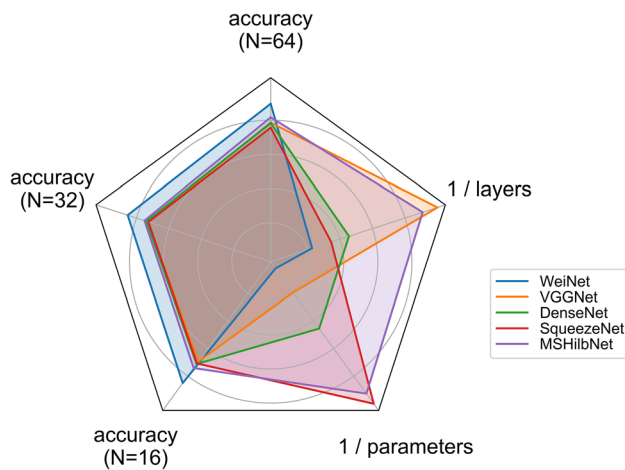


Fig. 7 Radar chart for the comparison of WeiNet, VGGNet, DenseNet, SqueezeNet and MSHilbNet with the Hilbert curve mapping approach. The axes correspond to classification accuracies for window lengths $N = 16$, $N = 32$ and $N = 64$, the inverse of the number of trainable layers and the inverse of the amount of parameters. The first three constitute a metric of the performance, while the last relate to the models' complexity

set to $\alpha = 0.05$ for all tests (Tables 11, 12, 13, 14, 15, reftable:rmanovaspsmshilbnet and 17). In Tables 11 and 12, the results are analyzed across three variables, i.e., classification model, representation method and window length. Differences in the classifier and the input method were significant ($p \ll 0.05$), which was not the case for the window length (Table 11). Pairwise comparisons revealed meaningful differences; thus, the combination with the higher performance is a DenseNet model with the 'Hilb-Elect' approach (Table 8). When the DenseNet, VGGNet, SqueezeNet and MSHilbNet are compared for the 'Hilb-Time' representation (Tables 13 and 14), considerable differences are found between the MSHilbNet and the other models ($p \ll 0.05$), while DenseNet and VGGNet have almost similar performance ($p = 0.059$). Regarding the window length, there is no difference between $N = 16$ and $N = 32$ ($p = 1.0$), while any variation between $N = 16$ and $N = 64$ is significant ($p = 0.002$). Consequently, for the HilbTime representation the MSHilbNet at window length $N = 64$ performs the best (Table 10).

Finally, a further analysis of the MSHilbNet is shown in Figs. 13 and 14 and Tables 15, 16 and 17. In Fig. 13, highly activated feature maps are shown for the single-scale (Fig. 13c) and the multi-scale (Fig. 13d, e) approaches when a 'Thumb up' gesture is performed. In addition, the softmax distributions of the intermediate classifiers and the final output calculated as their average are shown in Fig. 14. Three types of output layers (o) and two model depths (d) are evaluated on the validation set of 10 random subjects (Fig. 1 path (1)). Significant differences were found (Table 16) between the two variables ($p = 0.018$ and $p \ll 0.05$ for depth and output layer, respectively). Further,

the pairwise comparisons (Table 17) suggested that there is a marginal difference between the two depths (i.e., $d = 3$ and $d = 5$), whereas for the output layer the performance measured as the 'weighted average' of all the layers is considered similar to that of the deepest layer ($p = 0.220$).

6 Discussion

6.1 Hyper-parameter selection

For the hyper-parameter selection (Fig. 1 path (1), Table 4), we see that finding good parameters for the SqueezeNet is rather difficult, since the classification accuracy is low, and the models for some of the subjects did not converge. In the VGGNet and DenseNet, we observe that the 'max' pooling operation is inferior to 'average' pooling, whereas 'max' pooling provides better results for the SqueezeNet. Also, increasing the depth of the model has a smaller effect ($< 1\%$) to the VGGNet compared to the other models (e.g., 1% gain in DenseNet from $d = 3$ to $d = 5$). This is probably due to the fact that the VGGNet has many parameters even for shallower models that makes the optimization difficult. The optimal hyper-parameter values correspond to the highest achieved classification accuracy. Thus, for the VGGNet we select pooling $pl = 'average'$ and depth $d = 4$, for the DenseNet $pl = 'average'$ and $d = 5$, and finally for the SqueezeNet $pl = 'max'$ and $d = 3$.

In the case of MSHilbNet (Table 5), we see that in general adding more layers along the depth dimension does not yield any performance improvements. On the contrary, increasing the number of scales increases the accuracy by up to 9% (e.g., in the case where depth $d = 5$ and classifiers are attached to all depths $o = [5, 4, \dots, 1]$ the accuracy is improved from 0.6211 to 0.7127). In addition, averaging the outputs of the intermediate classifiers does not provide a higher classification accuracy (i.e., given the depth, the accuracy decreases the more the intermediate classifiers are). However, the performance of the intermediate classifiers improves when the network deepens, but it still remains lower than the corresponding single classifier case. Eventually, the best performance is achieved for depth $d = 3$, three scales $s = [8 \times 8, 4 \times 4, 2 \times 2]$ and a single classifier from the deepest layer $o = [3]$.

6.2 Main results

The main results of this work can be summarized in Tables 6, 7, 8, 9 and 10 which show the performance on the testing set (Fig. 1 path (3)). What we see across all networks is that the performance improves when the window length N increases. This is expected since a wider window

contains more temporal information which is useful for the classification. In addition, *generally the Hilbert representations improve ($p < 0.05$) the classification accuracy compared to the corresponding baseline*, except for one case, namely VGGNet and $N = 16$. The reason for the improvement is the locality preservation property of the Hilbert curve [20, 32], which, given a model architecture, allows learning correlations between distant points using less convolutional layers. More specifically, for window lengths $N = 64$ there is a 3% classification gain for ‘HilbTime’ (Hilbert curve mapping across time) in the VGGNet, 5% in the DenseNet and almost 8% for the ‘HilbElect’ (Hilbert curve mapping across electrodes) in the SqueezeNet. Since for a CNN classifier the inference time is negligible compared to the acquisition time, the size of the window length N accounts for most of the latency, an important aspect of real-time applications. Although the use of a 640-ms window is not considered appropriate in such cases [15, 43], shorter windows can be applied when a long window inhibits real-time performance. However, that results in a significant reduction of the accuracy gain obtained by the Hilbert curve mapping, but without the accuracy becoming worse than for the baseline case as shown in Tables 6, 7, 8, 9 and 10.

Furthermore, we can infer from the results that a model with many parameters benefits less from the Hilbert curve representation compared to one with few parameters (e.g., there are <1% gain in the WeiNet and about 3% increase in accuracy in the VGGNet). *Comparing the performance between the MSHilbNet and the VGGNet, DenseNet and SqueezeNet for segments of size $N = 64$ and the ‘HilbTime’ method, we can see that the MSHilbNet is always superior ($p < 0.05$)*. This can be attributed to the fact that at every depth the model has access to both fine- and coarse-level features, while in the other topologies coarser feature maps can only be achieved after a sufficient depth.

Apart from the classification metrics, other aspects of the investigated models can be compared. For example, the radar chart in Fig. 7 shows the differences between the models with respect to classification accuracy and model complexity. What we observe is that the higher accuracy of the WeiNet can be partly attributed to its huge model size. On the other hand, although the rest of the models have different sizes, they perform at the same but lower than the WeiNet accuracy level. Yet, this is achieved with only a fraction of the WeiNet’s size. Overall, we believe that the proposed MSHilbNet achieves a good balance between high accuracy and small model size.

6.3 Evaluation of the MSHilbNet

Next, we further analyze the performance of the MSHilbNet. *The main advantage of this model is the use of*

multiple scales of the input image which allows to capture both fine and coarse features at every depth. In Fig. 13, we show the feature maps with the highest activation generated from models with depth $d = 3$ and scales $s = \{8 \times 8\}$ (Fig. 13c), $s = \{8 \times 8, 4 \times 4\}$ (Fig. 13d), $s = \{8 \times 8, 4 \times 4, 2 \times 2\}$ (Fig. 13e), when using as input the middle segment of ‘Thumb up’ gesture (Fig. 13a, b). Comparing the corresponding features between the three models, we see that when using only one scale the model needs more layers to locate the region with useful information. In contrast, more scales of the input allow the model to locate significant features faster. For example, the feature map of the first convolutional layer has high amplitude activations in a larger area when one scale is used (Fig. 13c ‘b1_regular_0/5’), whereas in the case of three scales (Fig. 13e ‘b1_regular_0/28’) high values are limited to a smaller region. In addition, there is a correspondence to the activation patterns between subsequent layers as well as coarse scales (e.g., low right highly activated), while in the single-scale case subsequent activation maps activate different regions. Finally, the convolutions of the single-scale network fail to extract other important features (e.g., low left region) that more scales can identify (e.g., Fig. 13e ‘b2_strided_0/5’).

The second component of the MSHilbNet is the use of multiple classifiers. Figure 14 shows the softmax output of the intermediate and final classifier for two models with depth $d = 3$ (Fig. 14a) and $d = 5$ (Fig. 14b) for the first 12 gestures of the Ninapro dataset (basic finger movements). We can see that classifiers at the last layer are in general more accurate (i.e., high confidence for the correct label), while earlier classifiers tend to misclassify. To further evaluate whether a multi-classifier approach is helpful, we substituted the final average classifier with weighted classifier with learnable weights. In particular, the weights are learned during training through an attention mechanism. The classification results on the validation set (Fig. 1 path (1)) are shown in Table 15. Clearly, there is a great improvement from using the ‘weighted average’ since the classification accuracy is improved by 11% and 7% for $d = 3$ and $d = 5$, respectively, compared to the case of the ‘average’ classifier. *However, the performance is not significantly better than a single classifier ($p < 0.05$)*.

6.4 Future work

In the proposed methodology, the Hilbert curve provides a better locality preserving representation in only a single dimension, i.e., either the time (HilbTime) or the electrodes (HilbElect) dimension. It would be in our interests to investigate the application of higher-dimensional curves that combine both dimensions for the generation of the sEMG image. Particularly, this would be beneficial to the

case of electrode grids, where the input of the corresponding baseline approach is already a multi-channel image. Another drawback of the current approach might be the requirement for squared images with dimensions equal to powers of two, which limits the duration of the sequence to multiples of four. Though smaller sequences can be zero-padded or interpolated to match the required sequence length, experimentation is needed to evaluate the effect of these modifications. Future work will also investigate the generalization of the proposed Hilbert curve mapping to datasets with different configuration of sensors and set of gestures.

7 Conclusions

This paper investigated the generation of image representations of sEMG using the Hilbert fractal curve. The proposed methodology offers an alternative for the classification of sEMG patterns using image processing methods, while using the Hilbert curve offers the advantage of locality preservation. Two methods (HilbTime and HilbElect) were evaluated and showed superior performance across various networks compared to the window segmentation method (baseline). However, the benefit was smaller for models with many parameters. Then, we presented a model (MSHilbNet) with few trainable parameters that utilizes multiple scales of the initial Hilbert curve representation. The evaluation of this multi-scale topology

suggested that in every case it performed better than regular topologies based on VGGNet, DenseNet and SqueezeNet. Finally, an analysis provided insights into the performance of the MSHilbNet architecture.

Acknowledgements The work is supported by the ‘Andreas Mentzelopoulos Scholarships for the University of Patras’ and the VUB-UPatras International Joint Research Group on ICT (JICT).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

A Appendix

See Figs. 8, 9, 10, 11, 12, 13 and 14.

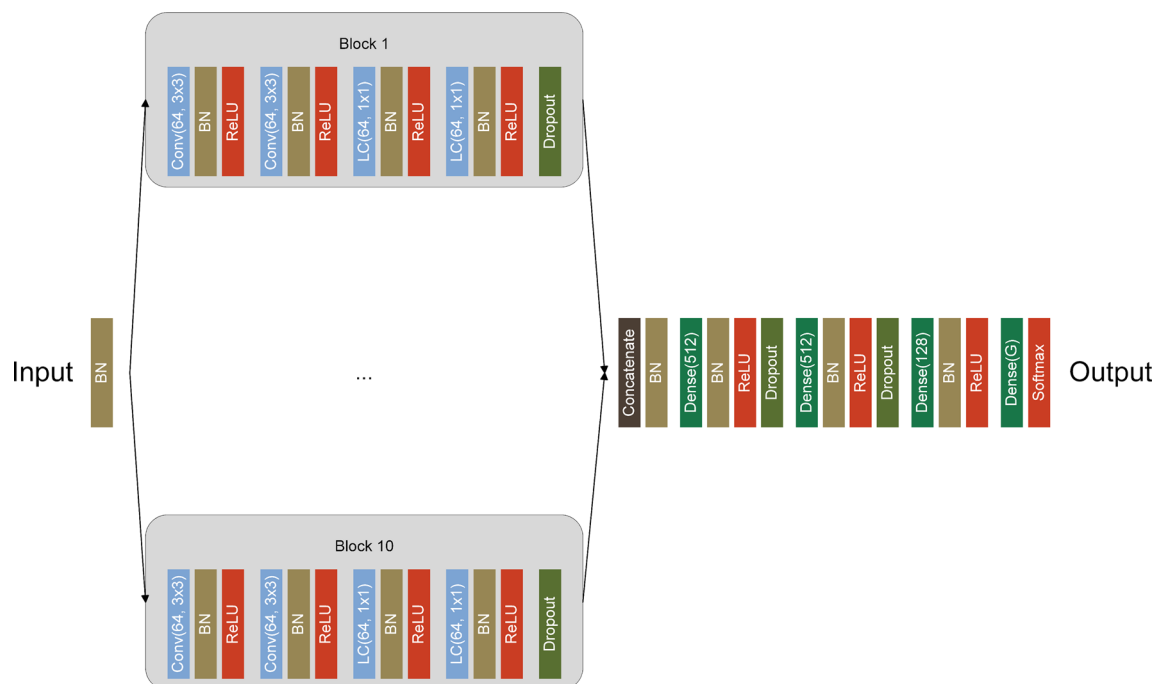


Fig. 8 The WeiNet [53] model follows a multi-stream approach. The multi-stream blocks consist of two 1×1 convolutions followed by two locally connected layers. The outputs from each stream are merged and further processed by dense layers

Fig. 9 A typical VGGNet [42] model architecture. It consists of successive blocks of convolutional and pooling layers. The classification label is obtained via a dense layer followed by softmax activation

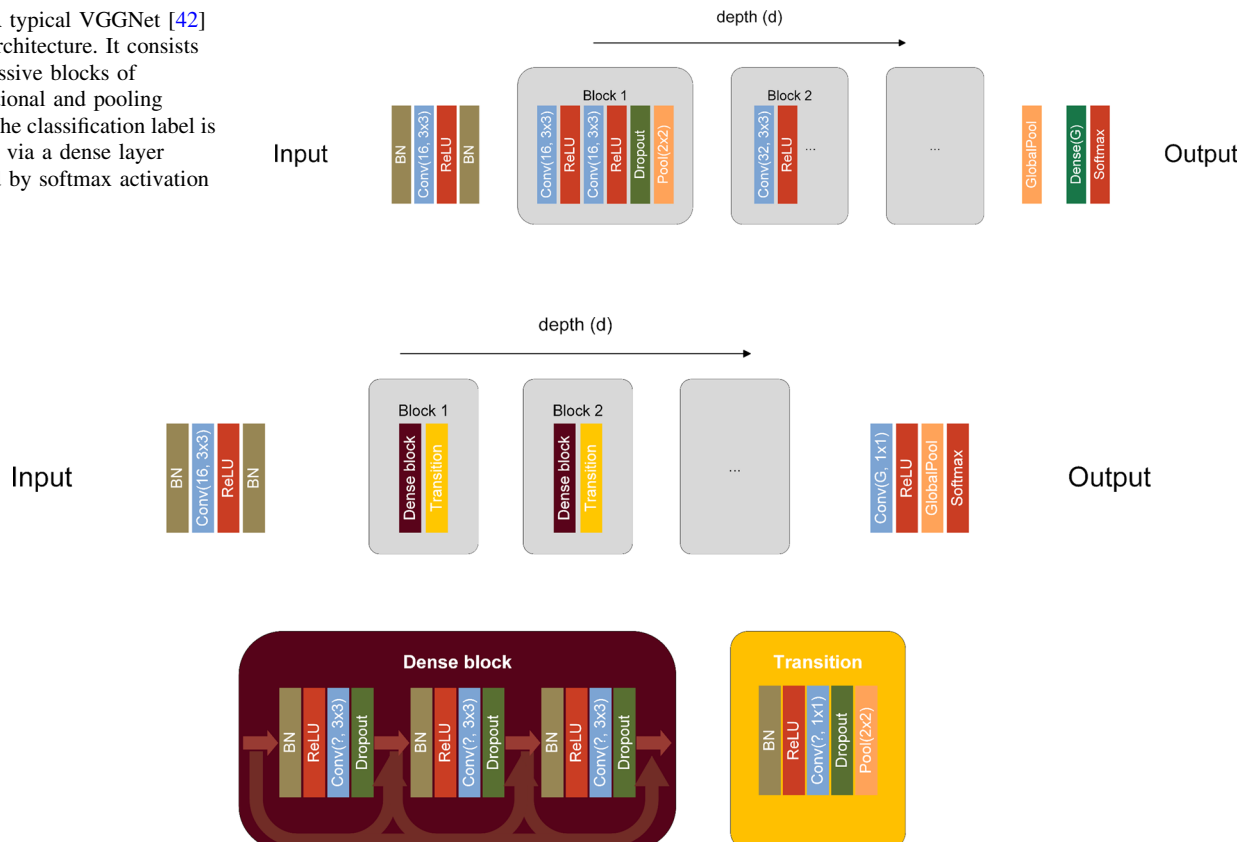


Fig. 10 The DenseNet [24] model consists of blocks with dense connections. These dense blocks are followed by a transition layer that performs a pooling operation

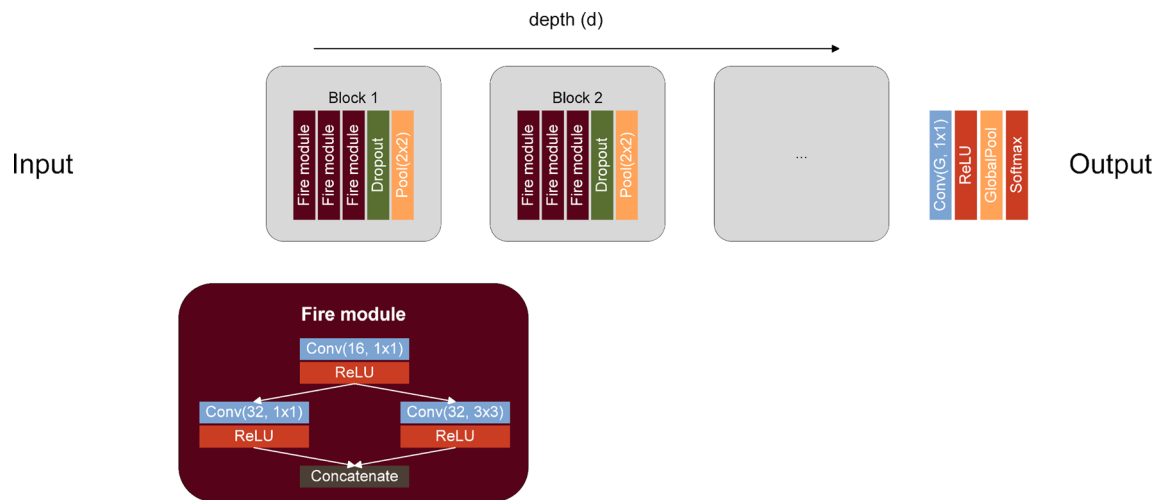


Fig. 11 The main block of the SqueezeNet [26] model uses the fire module to compress the feature map's dimensionality. The fire module performs a squeezing convolution followed by two parallel

expanding convolutions the outputs of which are merged into a single feature map. Between two successive blocks, a pooling operation is applied

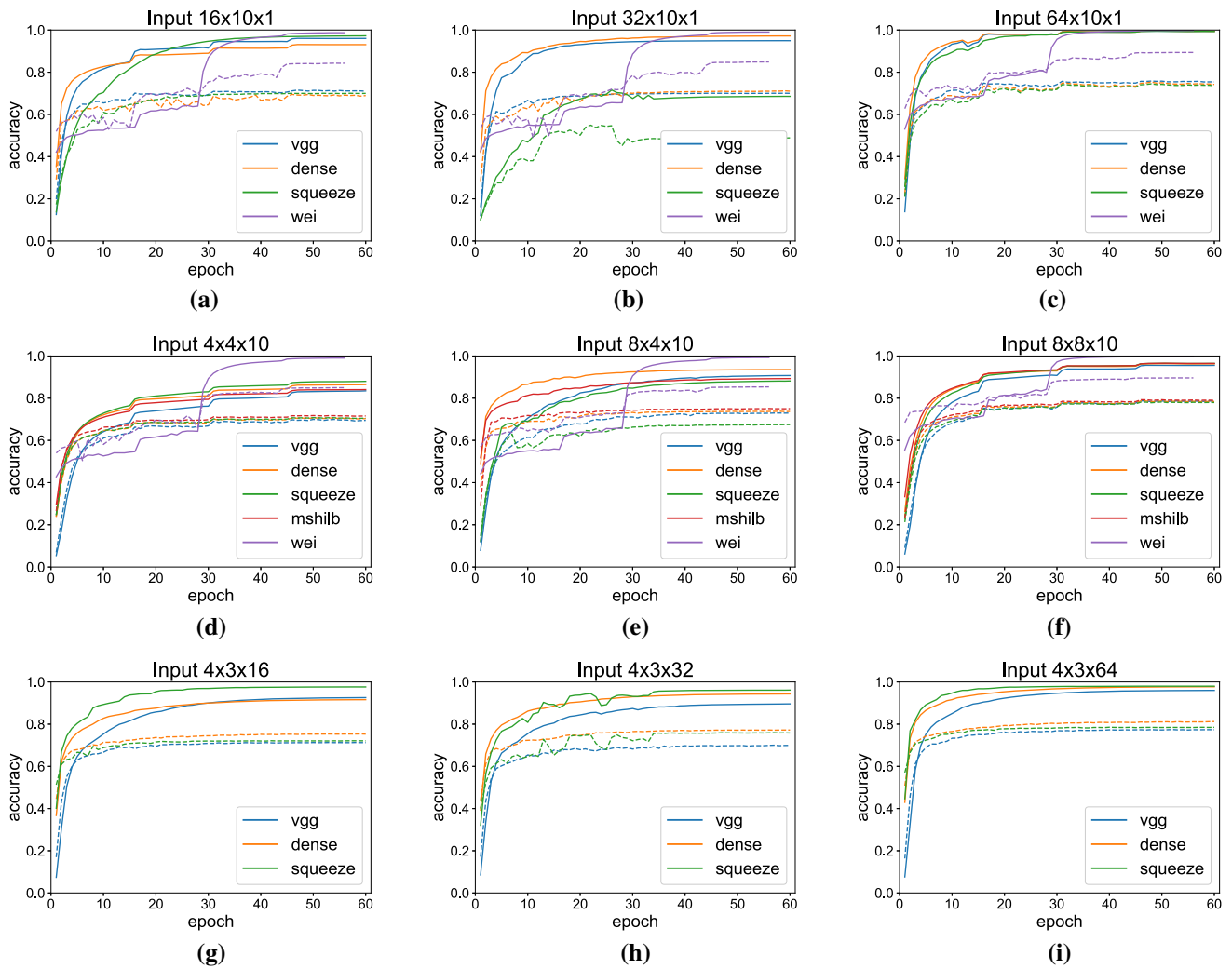


Fig. 12 Average accuracy curves for training (solid lines) and testing (dashed lines)

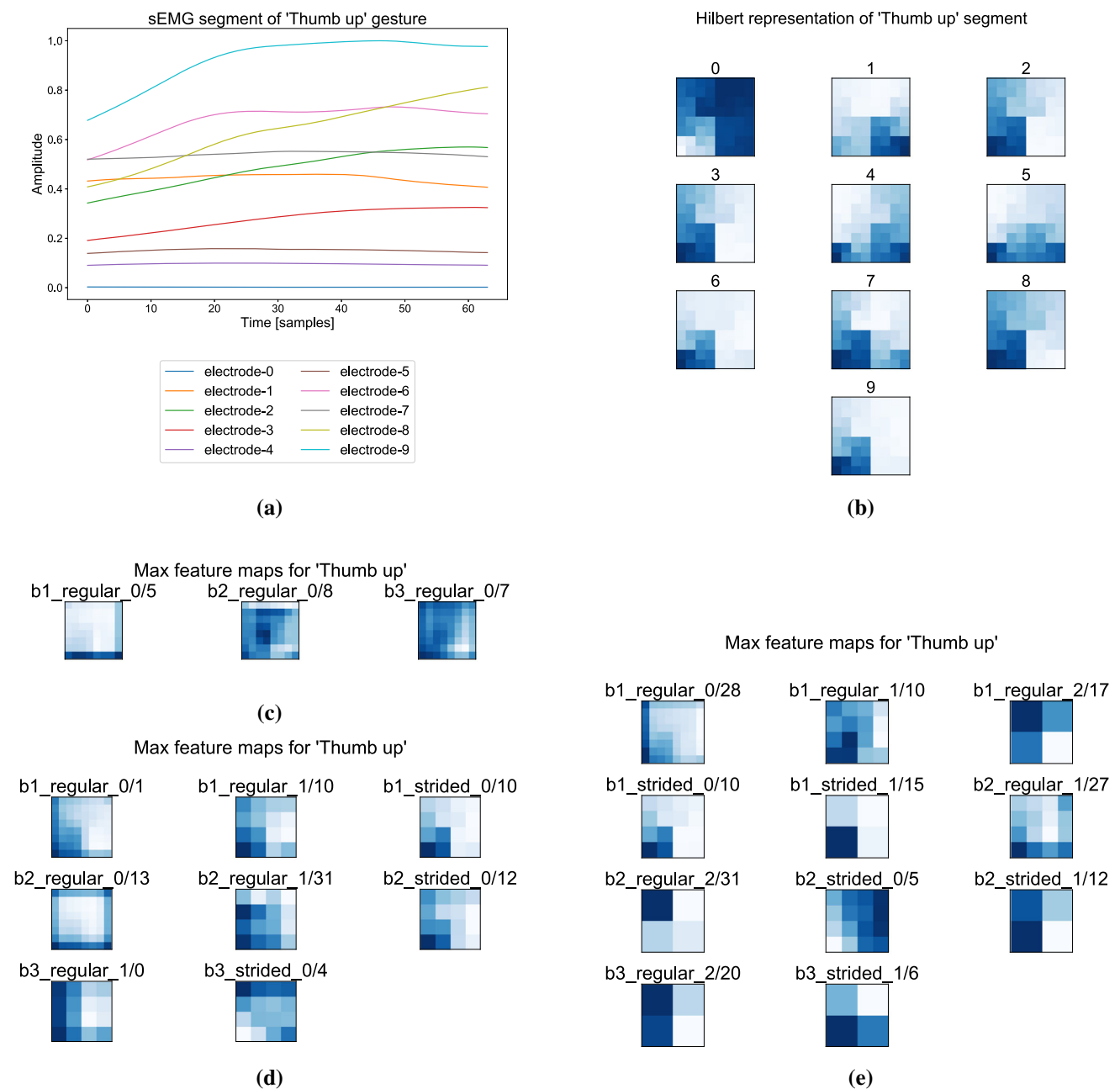


Fig. 13 Feature maps of highest activations generated from models with depth $d=3$ and scales $s=[8 \times 8]$ (c), $s=[8 \times 8, 4 \times 4]$ (d), $s=[8 \times 8, 4 \times 4, 2 \times 2]$ (e), when using as input the middle segment of 'Thumb up' gesture (a, b). The naming convention of the feature

maps is 'b_{level}__{type}__{scale}__{feature map index}', where level={1,2,3} is the depth level, type={regular, strided} is the type of convolution, and scale={0: 8×8 , 1: 4×4 , 2: 2×2 } is the resolution used as input

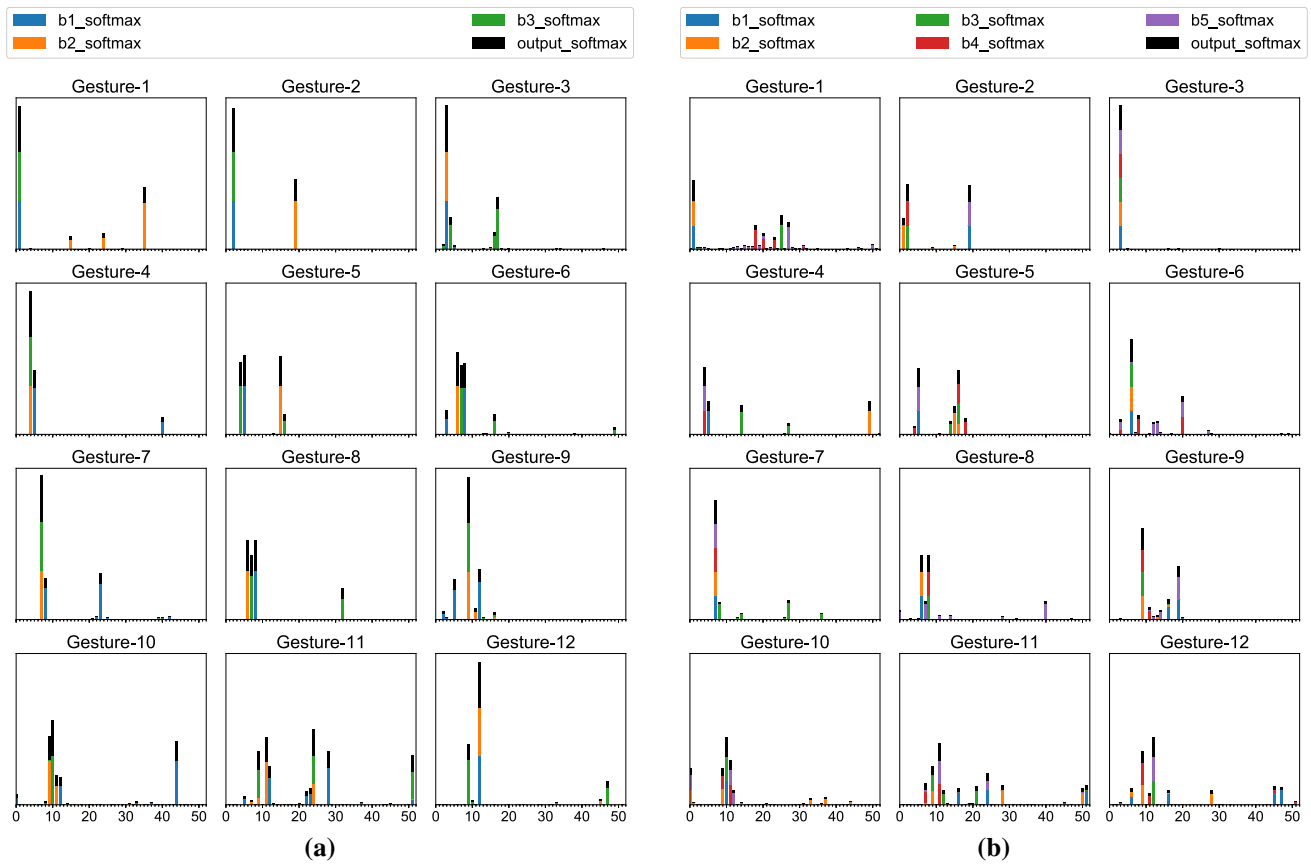


Fig. 14 Softmax distributions of intermediate and final classifiers for the cases of $d = 3$ and $d = 5$, using a simple average final classifier

See Tables 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 and 17.

Table 2 Average size of train and test sets. Standard deviations are reported in parentheses

| Configuration | Train | Test |
|---------------|------------------|-------------|
| $N = 16$ | 266,916 (17,301) | 3,667 (251) |
| $N = 32$ | 256,671 (17,272) | 1,795 (125) |
| $N = 64$ | 231,300 (17,301) | 857 (63) |

Table 3 Hyper-parameter selection

| Hyper-parameter | Search space | Optimal values | | | |
|------------------------|---|----------------|-----------|------------|-----------|
| | | VGGNet | DenseNet | SqueezeNet | MSHilbNet |
| Depth (d) | {3, 4, 5} | 4 | 5 | 3 | 3 |
| Pool (pl) | {‘max’, ‘average’} | ‘average’ | ‘average’ | ‘max’ | ‘max’ |
| Scales* (s) | {3, 2, 1}** | – | – | – | 3** |
| Output layers* (o) | { $[d]$, $[d, d - 2]$, $[d, d - 1, \dots, 1]$ } | – | – | – | $[d]$ |

*Parameters related to MSHilbNet only

The values {3, 2, 1} for s correspond to $\{[8 \times 8, 4 \times 4, 2 \times 2], [8 \times 8, 4 \times 4], [8 \times 8]\}$, respectivelyTable 4** Average accuracy of VGGNet, DenseNet and SqueezeNet models for different hyper-parameters with input $16 \times 10 \times 1$ ($N = 16$, no Hilbert) using the validation set from 10 randomly selected subjects. Standard deviations are reported in parentheses. The best performing model is indicated in bold

| d | pl | VGGNet | DenseNet | SqueezeNet |
|-----|-----------|------------------------|------------------------|------------------------|
| 3 | ‘max’ | 0.6532 (0.0765) | 0.4656 (0.0547) | 0.5750 (0.0644) |
| | ‘average’ | 0.6695 (0.0679) | 0.6185 (0.0730) | 0.5356 (0.0455) |
| 4 | ‘max’ | 0.6510 (0.0735) | 0.4142 (0.0716) | 0.5383 (0.0676) |
| | ‘average’ | 0.6720 (0.0673) | 0.6215 (0.0862) | 0.4787 (0.0491) |
| 5 | ‘max’ | 0.6442 (0.0793) | 0.4277 (0.0717) | * |
| | ‘average’ | 0.6640 (0.0699) | 0.6285 (0.0739) | * |

*The network failed to converge

Table 5 Average accuracy of MSHilbNet model for different hyper-parameters with input $8 \times 8 \times 10$ ($N = 64$) using the validation set from 10 randomly selected subjects. Standard deviations are reported in parentheses. The best performing model is indicated in bold

| s | o | $d = 3$ | $d = 4$ | $d = 5$ |
|--|------------------------|------------------------|-----------------|-----------------|
| $[8 \times 8]$ | $[d]$ | 0.7496 (0.0694) | 0.7465 (0.0711) | 0.7471 (0.0693) |
| | $[d, d - 2]$ | 0.6779 (0.0579) | 0.6890 (0.0689) | 0.6999 (0.0645) |
| | $[d, d - 1, \dots, 1]$ | 0.6061 (0.0416) | 0.6121 (0.0610) | 0.6211 (0.0631) |
| $[8 \times 8, 4 \times 4]$ | $[d]$ | 0.7692 (0.0716) | 0.7683 (0.0680) | 0.7639 (0.0661) |
| | $[d, d - 2]$ | 0.7057 (0.0614) | 0.7238 (0.0653) | 0.7215 (0.0694) |
| | $[d, d - 1, \dots, 1]$ | 0.6409 (0.0521) | 0.6546 (0.0571) | 0.6879 (0.0660) |
| $[8 \times 8, 4 \times 4, 2 \times 2]$ | $[d]$ | 0.7862 (0.0648) | 0.7826 (0.0671) | 0.7767 (0.0664) |
| | $[d, d - 2]$ | 0.7335 (0.0576) | 0.7382 (0.0599) | 0.7502 (0.0713) |
| | $[d, d - 1, \dots, 1]$ | 0.6718 (0.0494) | 0.6864 (0.0669) | 0.7127 (0.0619) |

Table 6 Average values of evaluation metrics for optimal models using full dataset. Standard deviations are reported in parentheses

| | WeiNet | | | | | |
|----------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| | BASELINE(16) | HILBTime(16) | BASELINE(32) | HILBTime(32) | BASELINE(64) | HILBTime(64) |
| Accuracy | 0.8427 (0.0496) | 0.8498 (0.0475) | 0.8487 (0.0477) | 0.8536 (0.0462) | 0.8937 (0.0374) | 0.8963 (0.0376) |

Table 7 Average values of evaluation metrics for optimal models using full dataset. Standard deviations are reported in parentheses. The depth of the model is denoted by d , and pl corresponds to the pooling method

| VGGNet | | | | | | | | | |
|----------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| $(d = 4, pl = \text{'average'})$ | | | | | | | | | |
| | BASELINE(16) | HILBTime(16) | HILBElect(16) | BASELINE(32) | HILBTime(32) | HILBElect(32) | BASELINE(64) | HILBTime(64) | HILBElect(64) |
| Accuracy | 0.7111 (0.0663) | 0.6946 (0.0591) | 0.7126 (0.0687) | 0.7261 (0.0654) | 0.7293 (0.0653) | 0.7252 (0.0678) | 0.7551 (0.0654) | 0.7858 (0.0549) | 0.7736 (0.0631) |
| Precision | 0.7159 (0.0664) | 0.7095 (0.0564) | 0.7219 (0.0671) | 0.7305 (0.0667) | 0.7385 (0.0612) | 0.7353 (0.0643) | 0.7633 (0.0661) | 0.7975 (0.0552) | 0.7844 (0.0618) |
| Recall | 0.7099 (0.0666) | 0.6948 (0.0597) | 0.7127 (0.0687) | 0.7259 (0.0654) | 0.7295 (0.0647) | 0.7253 (0.0669) | 0.7543 (0.0666) | 0.7857 (0.0563) | 0.7736 (0.0625) |

Table 8 Average values of evaluation metrics for optimal models using full dataset. Standard deviations are reported in parentheses. The depth of the model is denoted by d , and pl corresponds to the pooling method

| DenseNet | | | | | | | | | |
|----------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| $(d = 5, pl = \text{'average'})$ | | | | | | | | | |
| | BASELINE(16) | HILBTime(16) | HILBELECT(16) | BASELINE(32) | HILBTime(32) | HILBELECT(32) | BASELINE(64) | HILBTime(64) | HILBELECT(64) |
| Accuracy | 0.6895 (0.0719) | 0.7119 (0.0699) | 0.7532 (0.0618) | 0.7105 (0.0682) | 0.7372 (0.0627) | 0.7720 (0.0574) | 0.7329 (0.0699) | 0.7829 (0.0555) | 0.8112 (0.0578) |
| Precision | 0.7052 (0.0626) | 0.7194 (0.0673) | 0.7603 (0.0639) | 0.7162 (0.0647) | 0.7419 (0.0618) | 0.7793 (0.0573) | 0.7475 (0.0692) | 0.7918 (0.0521) | 0.8200 (0.0571) |
| Recall | 0.6885 (0.0692) | 0.7106 (0.0724) | 0.7524 (0.0623) | 0.7079 (0.0684) | 0.7357 (0.0627) | 0.7719 (0.0573) | 0.7294 (0.0741) | 0.7806 (0.0559) | 0.8107 (0.0578) |

Table 9 Average values of evaluation metrics for optimal models using full dataset. Standard deviations are reported in parentheses. The depth of the model is denoted by d , and pl corresponds to the pooling method

| SqueezeNet* | | | | | | | | | |
|------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| $(d = 3, pl = \text{'max'})$ | | | | | | | | | |
| | BASELINE(16) | HILBTIME(16) | HILBELECT(16) | BASELINE(32) | HILBTIME(32) | HILBELECT(32) | BASELINE(64) | HILBTIME(64) | HILBELECT(64) |
| Accuracy | 0.6931 (0.0611) | 0.7079 (0.0653) | 0.7142 (0.0644) | 0.6847 (0.0574) | 0.7258 (0.0581) | 0.7410 (0.0601) | 0.7077 (0.0564) | 0.7556 (0.0610) | 0.7788 (0.0681) |
| Precision | 0.6730 (0.0545) | 0.7053 (0.0651) | 0.7100 (0.0596) | 0.6848 (0.0562) | 0.7222 (0.0545) | 0.7038 (0.0835) | 0.7085 (0.0673) | 0.7127 (0.0587) | 0.7553 (0.0837) |
| Recall | 0.6795 (0.0557) | 0.7049 (0.0641) | 0.7105 (0.0617) | 0.6776 (0.0617) | 0.7209 (0.0564) | 0.7168 (0.0727) | 0.7008 (0.0629) | 0.7182 (0.0570) | 0.7632 (0.0740) |

* After removing results from subjects where training did not converge

Table 10 Average values of evaluation metrics for optimal models using full dataset. Standard deviations are reported in parentheses. The depth of the model is denoted by d , pl corresponds to the pooling method, s is the number of scales, and o the location of the output classifier

| MSHilbNet | | | |
|--|-----------------|-----------------|-----------------|
| $(s = [8 \times 8, 4 \times 4, 2 \times 2], d = 3, o = [3])$ | | | |
| | HILBTIME(16) | HILBTIME(32) | HILBTIME(64) |
| Accuracy | 0.7410 (0.0564) | 0.7502 (0.0683) | 0.8168 (0.0525) |
| Precision | 0.7601 (0.0509) | 0.7662 (0.0610) | 0.8258 (0.0492) |
| Recall | 0.7416 (0.0566) | 0.7503 (0.0687) | 0.8149 (0.0526) |

Table 11 Repeated measures ANOVA for evaluating the significance of differences between CNN model (Model: {DenseNet, VGGNet, SqueezeNet}), representation method (Method: {Baseline, HilbTime, HilbElect}) and window size ($N : \{16, 32, 64\}$). An ‘*’ denotes a significant difference ($\alpha = 0.05$)

| Source | df | F | Sig. |
|--------|-------|--------|--------------|
| Model | 1.211 | 32.757 | $\ll 0.05^*$ |
| Method | 1.615 | 45.403 | $\ll 0.05^*$ |
| N | 1.799 | 2.408 | 0.106 |

Table 12 Pairwise comparisons (Bonferonni correction) for the repeated measures ANOVA shown in Table 11. Values correspond to the p-value of the comparison between the quantities on the corresponding row and column. An ‘*’ denotes a significant difference ($\alpha = 0.05$)

| Model | DenseNet | VGGNet | SqueezeNet | Method | Baseline | HilbTime | HilbElect | N | 16 | 32 | 64 |
|------------|----------|--------|--------------|-----------|----------|--------------|--------------|----|----|-------|-------|
| DenseNet | - | 0.024* | $\ll 0.05^*$ | Baseline | - | $\ll 0.05^*$ | $\ll 0.05^*$ | 16 | - | 1.000 | 0.191 |
| VGGNet | | - | $\ll 0.05^*$ | HilbTime | | - | 0.001* | 32 | | - | 0.262 |
| SqueezeNet | | | - | HilbElect | | | - | 64 | | | - |

Table 13 Repeated measures ANOVA for evaluating the significance of differences between CNN model (Model: {DenseNet, VGGNet, SqueezeNet, MSHilbNet}) and window size (N: {16, 32, 64}) for the HilbTime representation method. An ‘*’ denotes a significant difference ($\alpha = 0.05$)

| Source | df | F | Sig. |
|--------|-------|--------|--------------|
| Model | 1.033 | 16.453 | $\ll 0.05^*$ |
| N | 1.462 | 7.222 | 0.005* |

Table 16 Repeated measures ANOVA for evaluating the significance of differences between the depth ($d : \{3, 5\}$) and output ($o : \{[d], [d, \dots, 1]$ ‘average,’ $[d, \dots, 1]$ ‘weighted average’}) hyper-parameters of the MSHilbNet model. An ‘*’ denotes a significant difference ($\alpha = 0.05$)

| Source | df | F | Sig. |
|--------|-------|---------|--------------|
| d | 1.000 | 8.345 | 0.018* |
| o | 1.116 | 112.354 | $\ll 0.05^*$ |

Table 14 Pairwise comparisons (Bonferonni correction) for the repeated measures ANOVA shown in Table 13. Values correspond to the p-value of the comparison between the quantities on the corresponding row and column. An ‘*’ denotes a significant difference ($\alpha = 0.05$)

| Model | DenseNet | VGGNet | SqueezeNet | MSHilbNet | N | 16 | 32 | 64 |
|------------|----------|--------|------------|--------------|----|----|-------|--------|
| DenseNet | - | 0.059 | 0.012* | $\ll 0.05^*$ | 16 | - | 1.000 | 0.002* |
| VGGNet | | - | 0.023* | $\ll 0.05^*$ | 32 | | - | 0.051 |
| SqueezeNet | | | - | $\ll 0.05^*$ | 64 | | | - |
| MSHilbNet | | | | - | | | | - |

Table 15 Comparison of average accuracy between MSHilbNet classifiers at depths $d = 3$ and $d = 5$ ($s = [8 \times 8, 4 \times 4, 2 \times 2]$ and $N = 64$). Standard deviations reported in parentheses

| | $d = 3$ | | | $d = 5$ | | |
|----------|--------------------|-----------------------------|--------------------------------------|--------------------|-----------------------------------|--|
| | $o = [3]$ | $o = [3, 2, 1]$, ‘average’ | $o = [3, 2, 1]$, ‘weighted average’ | $o = [5]$ | $o = [5, 4, 3, 2, 1]$, ‘average’ | $o = [5, 4, 3, 2, 1]$, ‘weighted average’ |
| accuracy | 0.7862 (0.0648) | 0.6718 (0.0494) | 0.7850 (0.0642) | 0.7767 (0.0664) | 0.7127 (0.0619) | 0.7875 (0.0624) |

Table 17 Pairwise comparisons (Bonferonni correction) for the repeated measures ANOVA shown in Table 16. Values correspond to the p -value of the comparison between the quantities on the corresponding row and column. An “*” denotes a significant difference ($\alpha = 0.05$)

| d | 3 | 5 | o | $[d]$ | $[d, \dots, 1]$ ‘average’ | $[d, \dots, 1]$ ‘weighted average’ |
|-----|---|--------|---|-------|---------------------------|------------------------------------|
| 3 | - | 0.040* | $[d]$ | - | $\ll 0.05^*$ | 0.220 |
| 5 | | - | $[d, \dots, 1]$ ‘average’ $[d, \dots, 1]$ ‘weighted average’ | | - | $\ll 0.05^*$ - |

References

- Anjum MM, Tahmid IA, Rahman MS (2019) CNN model with Hilbert curve representation of DNA sequence for enhancer prediction. *bioRxiv*
- Atzori M, Cognolato M, Müller H (2016) Deep learning with convolutional neural networks applied to electromyography data: a resource for the classification of movements for prosthetic hands. *Front Neurobot* 10:9
- Atzori M, Gijsberts A, Castellini C, Caputo B, Hager AGM, Elsig S, Giatsidis G, Bassetto F, Müller H (2014) Electromyography data for non-invasive naturally-controlled robotic hand prostheses. *Scientific Data* 1:140053
- Atzori M, Gijsberts A, Heynen S, Hager AGM, Deriaz O, Van Der Smagt P, Castellini C, Caputo B, Muller H (2012) Building the Ninapro database: a resource for the biorobotics community. In: *Proc IEEE RAS EMBS Int Conf Biomed Robot Biomechatron* pp 1258–1265
- Bappy JH, Simons C, Nataraj L, Manjunath B, Roy-Chowdhury AK (2019) Hybrid LSTM and encoder–decoder architecture for detection of image forgeries. *IEEE Trans Image Process* 28:3286
- Batista TVV, Machado LdS, Valença AMG, Moraes RMD (2019) FarMyo: a serious game for hand and wrist rehabilitation using a low-cost electromyography device. *Int J Serious Games* 6(2):3–19
- Castellini C, Fiorilla AE, Sandini G (2009) Multi-subject/daily-life activity EMG-based control of mechanical hands. *J Neuroeng Rehabil* 6(1):41
- Chang YJ, Chen SF, Huang JD (2011) A Kinect-based system for physical rehabilitation: a pilot study for young adults with motor disabilities. *Res Dev Disabil* 32(6):2566–2570
- Chen X, Zhang X, Zhao ZY, Yang JH, Lantz V, Wang KQ (2007) Hand gesture recognition research based on surface EMG sensors and 2D-accelerometers. In: *Proc Int Symp Wearable Comput*, pp. 1–4. IEEE
- Cheok MJ, Omar Z, Jaward MH (2019) A review of hand gesture and sign language recognition techniques. *Int J Mach Learn Cybern* 10(1):131–153
- Corcoran T, Zamora-Resendiz R, Liu X, Crivelli S (2018) A spatial mapping algorithm with applications in Deep Learning-based structure classification. *ArXiv e-prints*
- Côté-Allard U, Fall CL, Drouin A, Campeau-Lecours A, Gosselin C, Glette K, Laviolette F, Gosselin B (2018) Deep learning for electromyographic hand gesture signal classification using transfer learning. *ArXiv e-prints*
- Dhahbi S, Barhoumi W, Kurek J, Swiderski B, Kruk M, Zagrouba E (2018) False-positive reduction in computer-aided mass detection using mammographic texture analysis and classification. *Comput Methods Programs Biomed* 160:75–83
- Du Y, Jin W, Wei W, Hu Y, Geng W (2017) Surface EMG-based inter-session gesture recognition enhanced by deep domain adaptation. *Sensors* 17(3):458
- Englehart K, Hudgins B (2003) A robust, real-time control scheme for multifunction myoelectric control. *IEEE Trans Biomed Eng* 50(7):848–854
- Farina D, Merletti R, Enoka RM (2014) The extraction of neural strategies from the surface EMG: an update. *J Appl Physiol* 117(11):1215–1230
- Farina D, Jiang Ning, Rehbaum H, Holobar A, Graimann B, Dietl H, Aszmann OC (2014) The extraction of neural information from the surface EMG for the control of upper-limb prostheses: emerging avenues and challenges. *IEEE Trans Neural Syst Rehabil Eng* 22(4):797–809
- Geng W, Du Y, Jin W, Wei W, Hu Y, Li J (2016) Gesture recognition by instantaneous surface EMG images. *Scientific Rep.* 6(1):36571
- Gijsberts A, Atzori M, Castellini C, Muller H, Caputo B (2014) Movement error rate for evaluation of machine learning methods for sEMG-based hand movement classification. *IEEE Trans Neural Syst Rehabil Eng* 22(4):735–744
- Gotsman C, Lindenbaum M (1996) On the metric properties of discrete space-filling curves. *IEEE Trans Image Process* 5(5):794–797
- Holzmüller D (2017) Efficient neighbor-finding on space-filling curves. *Universität Stuttgart, Bachelor*
- Hu Y, Wong Y, Wei W, Du Y, Kankanhalli M, Geng W (2018) A novel attention-based hybrid CNN-RNN architecture for sEMG-based gesture recognition. *PLoS ONE* 13(10):e0206049
- Huang G, Chen D, Li T, Wu F, van der Maaten L, Weinberger KQ (2017) Multi-scale dense networks for resource efficient image classification. *ArXiv e-prints*
- Huang G, Liu Z, van der Maaten L, Weinberger KQ (2016) Densely connected convolutional networks. *ArXiv e-prints*
- Hudgins B, Parker P, Scott R (1993) A new strategy for multifunction myoelectric control. *IEEE Trans Biomed Eng* 40(1):82–94
- Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016) SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5mb model size. *ArXiv e-prints*
- Jiang N, Vujaklija I, Rehbaum H, Graimann B, Farina D (2014) Is accurate mapping of EMG signals on kinematics needed for precise online myoelectric control? *IEEE Trans Neural Syst Rehabil Eng* 22(3):549–558
- Krasoulis A, Kyranou I, Erden MS, Nazarpour K, Vijayakumar S (2017) Improved prosthetic hand control with concurrent use of myoelectric and inertial measurements. *J Neuroeng Rehabil* 14(1):71
- Kurek J, Swiderski B, Osowski S, Kruk M, Barhoumi W (2018) Deep learning versus classical neural approach to mammogram recognition. *Bull Pol Acad Sci Tech.* <https://doi.org/10.24425/bpas.2018.125930>
- Kuzborskij I, Gijsberts A, Caputo B (2012) On the challenge of classifying 52 hand movements from surface electromyography. In: *Annu int conf IEEE eng med biol soc*, pp. 4931–4937. IEEE

31. Li Y, Wang N, Shi J, Liu J, Hou X (2016) Revisiting batch normalization for practical domain adaptation. *ArXiv e-prints*
32. Moon B, Jagadish H, Faloutsos C, Saltz J (2001) Analysis of the clustering properties of the Hilbert space-filling curve. *IEEE Trans Knowl Data Eng* 13(1):124–141
33. Muceli S, Jiang N, Farina D (2014) Extracting signals robust to electrode number and shift for online simultaneous and proportional myoelectric control by factorization algorithms. *IEEE Trans Neural Syst Rehabil Eng* 22(3):623–633
34. Omelina L, Jansen B, Bonnechère B, Van Sint Jan S, Cornelis J (2012) Serious games for physical rehabilitation: designing highly configurable and adaptable games. In: *Proc int conf on disabil, virtual real assoc technol*, pp. 195–201
35. Park KH, Lee SW (2016) Movement intention decoding based on Deep Learning for multiuser myoelectric interfaces. In: *Int winter conf brain comput interface*, pp. 1–2. IEEE
36. Phinyomark A, Scheme E (2018) EMG pattern recognition in the era of big data and deep learning. *Big Data Cognitive Comput* 2(3):21
37. Prahm C, Vujaklija I, Kayali F, Purgathofer P, Aszmann OC (2017) Game-based rehabilitation for myoelectric prosthesis control. *JMIR Serious Games* 5(1):e3
38. Rautaray SS, Agrawal A (2015) Vision based hand gesture recognition for human computer interaction: a survey. *Artif Intell Rev* 43(1):1–54
39. Samadani A (2018) Gated recurrent neural networks for emg-based hand gesture classification. A comparative study. In: *Annu int conf ieee eng med biol soc*, pp. 1–4. IEEE
40. Scheme E, Englehart K (2011) Electromyogram pattern recognition for control of powered upper-limb prostheses: state of the art and challenges for clinical use. *J Rehab Res Dev* 48(6):643
41. Shim H, An H, Lee S, Lee E, Hk Min, Lee S (2016) EMG pattern classification by split and merge deep belief network. *Symmetry* 8(12):148
42. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. *ArXiv e-prints*
43. Smith LH, Hargrove LJ, Lock BA, Kuiken TA (2011) Determining the optimal window length for pattern recognition-based myoelectric control: balancing the competing effects of classification error and controller delay. *IEEE Trans Neural Syst Rehabil Eng* 19(2):186–192
44. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15:1929–1958
45. Stango A, Negro F, Farina D (2015) Spatial correlation of high density EMG signals provides features robust to electrode number and shift in pattern recognition for myoelectric control. *IEEE Trans Neural Syst Rehabil Eng* 23(2):189–198
46. Sun W, Liu H, Tang R, Lang Y, He J, Huang Q (2019) sEMG-based hand-gesture classification using a generative flow model. *Sensors* 19(8):1952
47. Tabor A, Bateman S, Scheme E, Flatla DR, Gerling K (2017) Designing game-based myoelectric prosthesis training. In: *Proc SIGCHI conf hum factor comput syst*, pp. 1352–1363. ACM Press, New York, New York, USA
48. Tsinganos P, Cornelis B, Cornelis J, Jansen B, Skodras A (2018) Deep learning in EMG-based gesture recognition. In: *Proc int conf physiol comput syst*, pp. 107–114. Scitepress, Seville, Spain
49. Tsinganos P, Cornelis B, Cornelis J, Jansen B, Skodras A (2019) A Hilbert curve based representation of sEMG signals for gesture recognition. In: *Int conf syst signals image process*. Osijek, Croatia
50. Tsinganos P, Skodras A, Cornelis B, Jansen B (2018) Deep Learning in gesture recognition based on sEMG signals. In: F. Ring, W.C. Siu, L.P. Chau, L. Wang, T. Tang (eds.) *Learn approaches signal process*, 1 edn., chap. 13, p. 471. Pan Stanford Publishing
51. Um TT, Pfister FMJ, Pichler D, Endo S, Lang M, Hirche S, Fietzek U, Kulić D (2017) Data augmentation of wearable sensor data for parkinson's disease monitoring using Convolutional Neural Networks. In: *Proc ACM int conf multimodal interact*, vol. 517, pp. 216–220. ACM Press, New York, New York, USA
52. Wei W, Dai Q, Wong Y, Hu Y, Kankanhalli M, Geng W (2019) Surface electromyography-based gesture recognition by multi-view deep learning. *IEEE Trans Biomed Eng* 66:2964–2973
53. Wei W, Wong Y, Du Y, Hu Y, Kankanhalli M, Geng W (2017) A multi-stream convolutional neural network for sEMG-based gesture recognition in muscle-computer interface. *Pattern Recognit Lett* 119:131–138
54. Yin B, Balvert M, Zambrano D, Schönhuth A, Bohte S (2018) An image representation based convolutional network for DNA classification. *ArXiv e-prints*

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.