Pattern Recognition Letters
journal homepage: www.elsevier.com

EMG-based online classification of gestures with recurrent neural networks

Miguel Simão[a], Pedro Neto[a,**], Olivier Gibaru[b]

[a] University of Coimbra, Department of Mechanical Engineering - POLO II, Coimbra, 3030-788, Portugal
[b] École National Superieure d'Arts et Métiers - ParisTech, 8, Bd Louis XIV, 59046 Lille Cedex, France

ABSTRACT

Online gesture classification can rely on unsupervised segmentation in order to divide the data stream into static and dynamic segments for individual classification. However, this process requires motion detection calibration and adds complexity to the classification, thus becoming an additional failure point. An alternative is the sequential (dynamic) classification of the data stream. In this study we propose the use of recurrent neural networks (RNNs) to improve the online classification of hand gestures with Electromyography (EMG) signals acquired from the forearm muscles. The proposed methodology was evaluated on the UC2018 DualMyo and the NinaPro DB5 data set. The performance of a Feed-Forward Neural Network (FFNN), a Recurrent Neural Network (RNN), a Long Short-Term Memory network (LSTM) and a Gated Recurrent Unit (GRU) are compared and discussed. Additionally, an alternative performance index, the gesture detection accuracy, is proposed to evaluate the performance of the model during online classification. It is demonstrated that the static model (FFNN) and the dynamic models (LSTM, RNN and GRU) achieve similar accuracy for both data sets, i.e., about 95% for the DualMyo and about 91% for the NinaPro DB5. Although both models had similar accuracies, the dynamic models (LSTM and GRU) have a third of the parameters, presenting smaller training and inference times. + + Long Short-Term Memory (LSTM)

1. Introduction

Sequential classification is the process of classification of a data stream in which each new frame of data originates a new prediction. In this way, given n new time steps, the classification model outputs n predictions. Furthermore, the classification of frame i can only use the information available in that frame and the previous, i.e., i, $i-1$, $i-2, \ldots$. As a consequence, the classification predictions are available in real-time and there is no need to wait for a gesture to end.

It is possible to achieve sequential classification in two ways: (1) using a dynamic classification model which takes as input raw or processed data in a sequential fashion; (2) a static model whose input are features determined from a window of frames ending at each time step of the input sequence, Simao et al. (2019). In both cases, the model must have some type of memory of previous events. The dynamic model has the "memory" provided by the parameters that define its state at each time step, while the static model is limited to the window of time steps that it is fed.

A dynamic model has some advantages relative to a static one. It is easier to implement online, as new data can be fed directly into the model one-by-one or by batches, without affecting its output. On the contrary, the static model's data must be split into windows of constant or variable size. Therefore, a window size must be specified, which can cause a decrease in accuracy of the model. Gestures can vary in length due to the speed of the user and/or his/her range of motion. In such a case, a small window will split the data of longer gestures and a large window will attach unimportant data to shorter gestures, Simão et al. (2016). Depending on the problem, a balance might be difficult to find, so it is worth pursuing the use of a dynamic model to avoid this problem of gesture time scale.

The classification of gesture patterns in EMG data obtained from the forearm's muscles using Convolutional Neural Networks (CNN) is proposed by Wei et al. (2017). The performance of different classifiers (CNN, LSTM, and hybrid solutions) is presented in Xie et al. (2018). It was reported that the

**Corresponding author: Tel.: +351-239-790-700; fax: +351-239-790-700;
e-mail: pedro.neto@dem.uc.pt (Pedro Neto)

hybrid solutions present the best performance. A hybrid CNN and RNN architecture that aims to better capture temporal properties of EMG signals for gesture recognition is proposed in Hu et al. (2018). The authors also propose a novel EMG image representation, allowing deep learning architectures to extract implicit correlations from EMG. Experiments indicate that the performance obtained outperforms state-of-the-art methods Hu et al. (2018). An RNN based approach that outperforms the typical window-wise approaches for hand movement classification is proposed in Koch et al. (2018). A comparative analysis between different RNN configurations for EMG-based hand gesture classification is in Samadani (2018). In particular, RNNs with recurrent units of LSTM and gated recurrent unit (GRU) are evaluated. The NinaPro2 hand gesture dataset was used, achieving a classification accuracy of 86.7%. An interesting study proposes a model and a deep-learning-based domain adaptation method, 2-Stage RNN, to approximate the domain shift for recognition accuracy enhancement Ketykó et al. (2019). The achieved classification accuracy using NinaPro (12 gestures) is about 85% and the NinaPro (8 gestures) is about 91%. The selected features have a great impact into EMG-based pattern classification. Many times the feature set is optimized for a specific application, lacking the capacity to generalize. Previous studies report that high performance in EMG classification can be achieved from simple feature sets, namely the standard deviation Wolf et al. (2013) and the root mean square Ju and Liu (2014), Paleari et al. (2015).

In this study, we propose the use of LSTM networks LeCun et al. (2015) to classify gesture patterns from EMG data. Results are compared with other classifiers. This is, to the best of our knowledge, a novel application of LSTM and RNN in general to online gesture classification, presenting advantages through the simplification of the data chain from the sensors to the classification model and online output.

2. Methodology

In this section we discuss the data pipeline for the fitting and test of the model, its architecture, training methodology and performance indicators.

2.1. Recurrent Networks

A RNN is an extension of a FFNN with loops in the hidden layers. This allows the model to take as input a sequence of samples and find time-relationships between them. However, they have been found to have issues in learning long-term relationships Bengio et al. (1994). LSTM networks solve this issue by adding parameters to the hidden node loops so that they can acquire and release states depending on the input sequence Hochreiter and Schmidhuber (1997). Therefore, states are activated according to short-term events, while the network can keep those states active indefinitely, providing long-term memory to the network. LSTM have been found to be better at learning sequences than classic RNN.

A given network can be generalized by equation (1). Input $X \in \mathbb{R}^{t \times d}$ is a sequence of t steps and d channels. Output $Y \in \mathbb{R}^{t \times n_c}$ is a sequence of the same length t and with dimensionality n_c (number of output channels – classes). The parameters of the network are represented by Θ. This form represents a one-to-one configuration in which each time step of the input data generates a time step of the output.

$$Y = \mathcal{L}_\Theta(X) \tag{1}$$

While a regular RNN node has a single weight and bias, an LSTM network has four times that value. There are four weight/bias pairs:

1. forget gate layer;
2. input gate layer;
3. output gate layer;
4. state gate layer.

The forward pass equations of an LSTM cell are described in Equation (2). The input and forget gates, i_t (2a) and f_t (2b) respectively, control how much of the previous hidden state h_{t-1} and current input x_t contribute to the cell state c_t. The forget, input and output gates' activation is scaled by a sigmoid function σ and the hidden state is the output filtered with the hyperbolic tangent function *tanh*.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2a}$$

$$f_t = \sigma\Big(W_f \cdot [h_{t-1}, x_t] + b_f\Big) \tag{2b}$$

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{2c}$$

$$\tilde{c}_t = \tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right) \tag{2d}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \tag{2e}$$

$$h_t = o_t \cdot \tanh(c_t) \tag{2f}$$

Equations (2) show that the activation value of a LSTM cell requires knowledge of the preceding value in time. Therefore, the optimization of the network's parameters, through Stochastic Gradient Descent (SGD), is done on sequences of input data time steps. Typically, the input data are split into shorter windows in order to reduce the Backpropagation (BP) time depth. The structure of the network is similar to FFNN. The input layer has a fixed time-length and number of variables. Fully connected nodes and LSTM layers compose the hidden layers, and lastly, there is an output layer with a *softmax* transfer function for classification. The time-length of the input and output layers is the same, due to the sequence-to-sequence classification configuration. The state of the LSTM cells is kept between consecutive training sequences.

An alternative to LSTM networks with less parameters is a Gated Recurrent Unit (GRU), Cho et al. (2014). An update gate (3a) with a single weight controls how much of the previous hidden state activation h_{t-1} influences the new state. The reset gate (3b) has the opposite function and its activation controls how much past states are discarded. The memory state update $\tilde{h}_t$ in given by (3c) and the final state is a weighted average of the previous state h_{t-1} and the state update $\tilde{h}_t$ (3d). GRU networks have the same advantages as LSTM networks when compared to simple RNNs, but have less parameters. Therefore, they are faster in training and inference.

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \tag{3a}$$

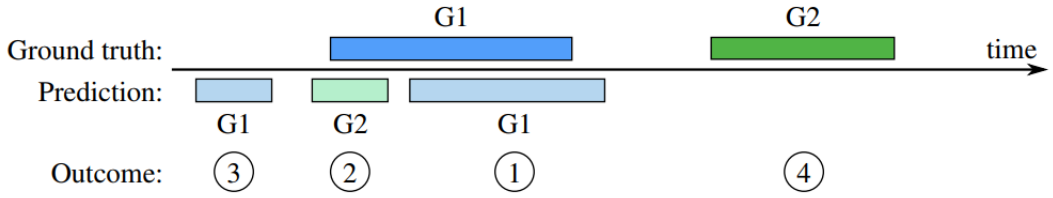

Fig. 1. Types of possible outcomes of the classification of gesture sequences. In this example, there are two gesture classes G1 and G2 interwoven by pauses. The outcome 1 represents a true positive (TP), the outcome 2 represents a misclassification (MC), the outcome 3 represents a false positive (FP) and the outcome 4 represents a false negative (FN).

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \tag{3b}$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right) \tag{3c}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{3d}$$

The networks are trained with SGD with early stopping when the classification loss stops decreasing on a non-trained data set. Early stopping prevents the model from over-fitting on the training data, but we must ensure a good local solution is reaching through hyper-parameter tuning. The hyper-parameters of interest are the architecture of the network (layers and size), the learning rate, momentum, batch size and sequence length. There is no method to define these hyper-parameters, so they must be set through manual or stochastic search. However, since LSTM networks are resource intensive careful manual search is preferred. The most important parameters to manually tune are the architecture and learning rate, while the others are set to commonly used values.

2.2. Data Pipeline

We assume that we have a labelled data set such that:

$$\mathcal{D} = \left\{\left(X^{(i)}, \iota^{(i)}\right) : \quad i = 1, 2, \ldots, n_{samples}\right\} \tag{4}$$

where $X^{(i)} \in \mathbb{R}^{t \times d}$ represents the sample data of d channels (variables) and t time steps, and $\iota^{(i)} \in \mathbb{N}_0^t$ is the vector of target class' indexes for each of the sample's time step. Index (i) corresponds to the sample's index in the data set. The data set is split into three subsets: training, validation and testing sub-sets for development and evaluation purposes.

Depending on the classifier model, features must then be extracted from the data. Generally, it is defined by $F^{(i)} = \mathcal{F}\left(X^{(i)}\right)$, where $\mathcal{F}$ represents the extraction function and $F \in \mathbb{R}^{n_f}$ are the output features, which is a matrix of length equal to the number of features per sample, n_f. In this solution the features cannot be calculated with time steps from the future. Feature vector F_t, at time step t, can only be determined with the sample time steps X_m where $m \leq t$.

Research on EMG signal processing showed us that there is not a single best solution for feature extraction Phinyomark et al.. Previous work showed that the standard deviation of the EMG signals correlates well with force being exerted by the muscle, so we chose to use the standard deviation as the single feature. It is calculated in a window of w time steps ending at step i by:

$$F_{ij} = \sqrt{\frac{1}{w-1}\sum_{k=0}^{w-1}\left(X_{(i-k)j} - \overline{X}_{ij}\right)^2}, \tag{5a}$$

$$\overline{X}_{ij} = \frac{1}{w}\sum_{k=0}^{w-1} X_{(i-k)j} \tag{5b}$$

where $j = 1, 2, \ldots n_c$ with n_c being the number of channels in the EMG data. Since a single feature is extracted, the dimensionality of F is the same as X, except the skipped time steps at the beginning of a sequence due to the features being calculated on a fixed window size. This is not an issue because most sequences begin in a rest state.

The features are then normalized, i.e., assuming the variables follow normal distributions, whose parameters are calculated in the training set. All of the subsets are normalized with these parameters and every new sample is normalized with these same parameters. Finally, the targets are one-hot encoded.

As previously mentioned, each time step in X_i has a matching target ι_i. Given that, the features are calculated on a window of frames, the choice of target label for each window is ambiguous. Therefore, we determined that it is given by the mode of the targets within that window ending at i:

$$\iota_i = \text{mode}\left(\{\iota_{i-k} : k = 0, 1, \ldots, w-2, w-1\}\right) \tag{6}$$

The output class sequences may be noisy due to noise in the source EMG data, user mistakes or class uncertainty in the transitions between gestures. These sources lead to small portions of a gesture, mainly in its boundaries, to be misclassified. We propose some post-processing steps in order to prevent errors due to these misclassifications. Firstly, gestures with length below an arbitrary value are disregarded. Then, consecutive gestures of the same class, broken by the previous filter, are merged, e.g., 1-1-2-1-1-1 is turned into 1-1-1-1-1-1.

2.3. Gesture Detection Criterion

A common performance indicator in continuous gesture classification is the Jaccard similarity index, also known as intersection over union. For a gesture spanning the true targets A and prediction B, it is defined by:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{7}$$

The co-domain is between 0 and 1, with 0 meaning that there is no overlap between the reference value and the prediction,

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

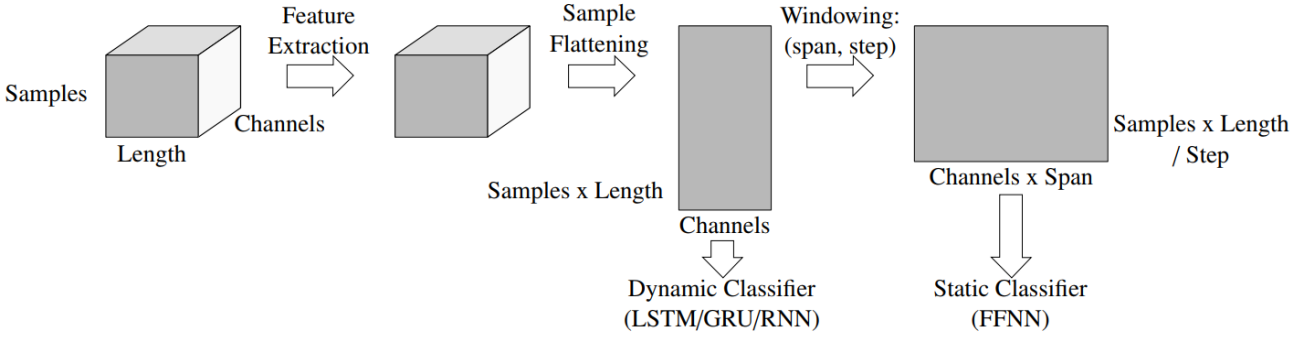

Fig. 3. Data processing chain for the RNN networks performance analysis.

while 1 means that there is perfect overlap. However, for gesture recognition, there is no difference whether there is 50% or perfect overlap. For this reason, we propose a different criterion for performance measurement of the classifier's output. We consider that there are four types of outcome in multi-class gesture detection, Fig. 1:

1. True positive (TP) – good classification;
2. Misclassification (MC) – wrong gesture detected;
3. False positive (FP) – gesture detected when there is none;
4. False negative (FN) – gesture not detected.

This list is derived from the four types of outcome in binary classification with the difference that true negatives are not considered and misclassifications (wrong gesture detected) are added. An example of each outcome is shown in Fig. 1.

We considered that good classifications are correctly predicted gestures that have an overlap of at least 50% with the ground truth. If we have an overlap inferior to 50% it can be an indication that we are in the presence of a wrongly classified gesture. Misclassification occur when there an overlap between a predicted gesture and a ground truth gesture but its classification is wrong. False positives occur when a gesture is detected during a pause (rest state) and false negatives occur when there is no overlap between a predicted and a ground truth gesture. The final score is given by:

$$\text{score} = \frac{TP}{TP + MC + FP + FN} \tag{8}$$

A score of 1.0 means that no errors occurred.

3. Experiments

3.1. Test Setup

The presented methodology was tested on the synthetic sequences of the UC2018 DualMyo data set Simão et al. (2018) and a similar subset of the NinaPro DB5 data set Pizzolato et al. (2017), both used for EMG-based hand gesture recognition. All the tests use the same fixed data split and the same data samples: 60% for the training set, 20% for validation and 20% for testing. The training set is used to train the classifier, while the validation set's performance is used to optimize the hyperparameters of the classification model, i.e., neural network structure and training hyper-parameters. Finally, the test set is used to test the generalization capability of the model and is only used when the training and validation sets provide desirable metrics. Therefore, the model is not optimized for the test set and the metrics calculated on it should provide a good measurement of the model performance in other conditions.

The UC2018 DualMyo data set acquisition setup has a total of 16 EMG channels and 8 gesture classes with 110 samples each. There are a total of 95 synthetic sequences, each one composed by 8 gesture samples, Fig. 2, each one with a normal acquisition rate of 200 Hz. While the hand shape is represented in the figures, it is not strictly defined. However, it is important that the correct muscles groups are actuated. Their actuation

is guaranteed by forcing the palm moves in the correct direction. For example, for the wave in gesture, the palm must be forced towards the body. Each sequence has a length of about 50 seconds, i.e., about 10000 time steps. We seek to compare the performance between a static model operating on windows of frames to a dynamic model that predicts on the time steps sequentially.

The NinaPro DB5 data set has the same acquisition as the DualMyo data set. It is composed by ordered sequences of 17 gesture classes, each class repeated 6 times. In order to prevent the models from learning the ordered sequence of gestures, the repetitions were randomly reorganized. Furthermore, we selected a subset of 8 out of the 17 classes, similar to the DualMyo's set of gesture classes. The resulting data set has nearly the same number of time steps, but the number of repetitions is only half of those in the DualMyo's data set. In this paper, we only present the models used for the DualMyo data set, but the description of the NinaPro models is available in the supplementary materials.

The data processing chain is slightly different for the static and dynamic classification models, as shown in Fig. 3. We start by extracting features from the 95 sequences, denominated samples. Their length is the number of time steps of the data set's sequences (10000), while the number of EMG channels is 16. The shape of the input data is maintained after feature extraction. As previously mentioned, the feature chosen is the standard deviation (σ) along time of a window of 100 time steps (0.5 seconds which is the average length of a gesture on the UC2018 DualMyo dataset), for each channel. Therefore, the number of variables is maintained. The step of the window feature extraction is a single frame, in order to keep the same time scale between the raw input data and their features.

The following step in the data processing chain is concatenating every sample into the same master sequence. Every sample starts and begins during a pause, so there are no discontinuities in the master sequence. Afterwards, this sequence can be either fed directly into the dynamic model or windowed again for the static classifier. In the latter case, the chosen window span was 200 frames for the DualMyo data set and 300 frames for the NinaPro. The rolling window has step 1 for both cases. The span of 200/300 time steps is specially chosen since it corresponds to one second of data, which is about the typical minimum length of a gesture. The windows of time steps are concatenated into a single vector, which then serves as the classification predictor for that time step. The target of that time step is the mode of targets for the same time window.

3.1.1. Static Model

The static classifier chosen is a FFNN, which has faster training and inference times for large amount of data, compared to other machine leaning methods. Considering the the DualMyo data set, there are about 933 thousand time steps, of which 561 thousand are used for training and the remaining are evenly split into the validation and testing sets. Since the selected features are a concatenated window of 200 time steps, the feature vector has length 3200.

The chosen model has an input layer with 3200 nodes, two fully-connected hidden layers with 512 units and a *softmax* output layer with 8 nodes, equal to the number of gesture classes in the data set. The transfer function of the hidden layers is the hyperbolic tangent. A representation of the network structure is shown in Fig. 4. This structure was determined by manual search but the output is not very sensitive to the hyper-parameters on this problem. There are almost 2 million trainable parameters, which were optimized with the ADAM variant of SGD Kingma and Ba (2014), a learning rate of 0.01 and a batch size of 256. The optimization is halted with the early stopping method, to prevent over-fitting, when the model loss of the validation set stops decreasing during 12 iterations. The model is then tested on all data splits (training, validation and testing) and the following metrics are calculated:

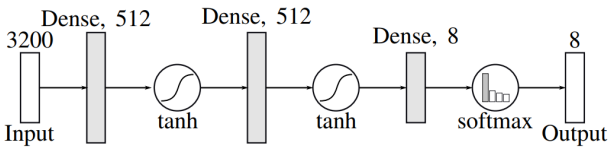

Fig. 4. Structure of the FFNN used as the static classifier model for the DualMyo data set.

1. Training time;
2. Inference time;
3. Frame-wise accuracy (FWA);
4. Gesture detection accuracy (DA).

3.1.2. Dynamic Model

The dynamic model (considering the LSTM defined in section 2.1) has 933 thousand time steps of the data set that were split in the same way as for the static model. However, in this model, the input features are a single frame of data from the 16 EMG channels of the DualMyo data set, instead of the 3200 features the static model uses. This has great advantages for the size of the model and the total memory used. The same reasoning was applied to the NinaPro DB5 data set.

The LSTM model, Fig. 5, has 16 input nodes followed by a fully-connected (dense) layer of 400 units. Afterwards, there is a recurrent layer with 256 LSTM cells, which feeds a dense layer of 8 units. Finally, a *softmax* transfer function provides the classification output probability distribution. The dense and LSTM hidden layers have the hyperbolic tangent as their activation function. This structure was also determined by manual search, with the performance having a low sensitivity to the hyper-parameters. This structure has just 683 thousand parameters, 66% less than the static model mainly due to the low dimensionality of the input layer. The optimization of the parameters is also done with ADAM's SGD, with a learning rate of 0.001, batch size of 10 and sequence length of 200 time steps. The training process is halted after 12 epochs without improvement of the validation loss.

We compared the performance of the LSTM model with a simple RNN model and a GRU RNN. These models have the same structure and number of nodes as the LSTM model and were trained in the same circumstances.

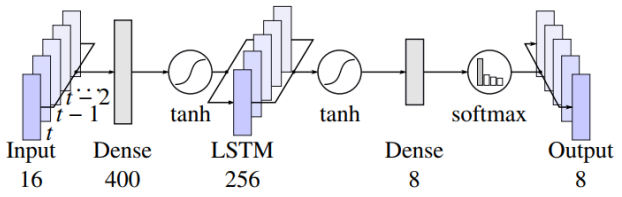

Fig. 5. Structure of the LSTM used as the dynamic classifier model, with a one-to-one input-output classification configuration.

Table 1. Training and inference times for the static and dynamic models, tested on the UC2018 DualMyo data set. Results for training and testing time are in seconds.

Model	Train Steps	Train Time (s)	Inference Steps	Test Time (s)	Frames/ second
FFNN	555,607	411.8	923,579	11.2	82k
RNN	555,600	301.9	928,000	14.9	62k
GRU	555,600	164.5	928,000	3.5	265k
LSTM	555,600	164.5	928,000	3.8	244k

3.2. Results and Discussion

The networks were defined and trained with the Keras library (python) using the Tensorflow backend Chollet et al. (2015); Abadi et al. (2015). The hardware used was a computer with a Nvidia GTX970M GPU (6GB VRAM), an Intel i7-6700HQ CPU and 32GB of RAM. The classification models were trained according to the methodology previously described. The training and inference processes are performed on the GPU. In our application target the classified gestures serve to intuitively interface with an industrial robot. In such scenario, the EMG sensors send data (wireless) to the above-mentioned external computer that in turn is connected to the robot. All data processing is performed on the external computer allowing continuous online gesture classification.

The training and inference times for the UC2018 DualMyo data set are shown in table 1. The number of training time steps changes slightly between the static and dynamic models due to the padding added to the end of the sequences in order to allow fixed-size batches of data to be fed into the dynamic model. The padding is typically filled by zeros and does not influence the training loss and the parameter optimization. The training time of the dynamic models (LSTM and GRU) is about 165 seconds, much less than the RNN (302 seconds) and the FFNN (412 seconds). The difference is mainly due to the discrepancy between the model's sizes, since the learning rate used to train the LSTM and GRU is an order of magnitude lower than that of the static model. For that reason, the LSTM and GRU are also faster in inference, classifying about 930 thousand time steps (about 1 hour and 20 minutes of data) in less than 4 seconds, while the static model FFNN takes 12 seconds and the RNN takes 15 seconds. In any case, inference time is well below the acquisition time, but the hardware requirements are high and still need optimization for embedded and portable devices. Another disadvantage of the static model is that it requires much more memory during training, since each time step requires 200 frames to be stored. This is a significant constraint for large data sets. Concerning the training and inference times the NinaPro DB5 data set presents similar behaviour.

The accuracy of the models is presented in table 2 for both the static and dynamic models. We present the frame-wise accuracy (proportion of correctly classified time steps) of the models in the 3 data splits, training, validation and testing. We also present the detection accuracy, as defined in (8). Additionally, we show what types of detection errors occurred in each data split: (1) true positives (TP), (2) misclassifications (MC), (3) false positives (FP), and (4) false negatives (FN).

In terms of frame-wise accuracy, both models present similar classification accuracy on the training and test sets, while the validation performance is slightly lower. The discrepancy between data splits is due to particularities in the sequences of each split rather than over-fitting, since the performance on the testing set is close to that of the training set.

While the frame-wise classification accuracy is an indicator of overall performance of the model, it does not tell us if there are misclassifications in the middle of gestures, causing the model to momentarily and wrongly announce a poor gesture classification. The detection accuracy measurements indicate whether gestures as a whole are correctly identified, or if the model is finding gestures when it should not (false positives). The gesture detection accuracy was determined on the post-processed output of the models. Gestures below a specific length (0.5 seconds) are disregarded in order to reduce the classification noise of the models. For the DualMyo data set the detection accuracy was generally better than the frame-wise accuracy, table 2. It exceeded 99% in the test split for all models. The validation and test splits presented no errors, but the truth/prediction overlap was lower than 50% in a few gesture samples. For the NinaPro dataset, the results are slightly different since the frame-wise accuracy was generally better than the detection accuracy in the test split, except for the FFNN. The best detection accuracy obtained was about 91% in the test split in the static model (FFNN).

Outcomes of type MC (misclassifications) are the result of a gesture being totally or partially misclassified. Examples of FP outcomes are visible in Fig. 6, where often times gesture 5 (G5) is detected in the transitions between the gesture and the surrounding rest states. This gesture (double-tap) recruits a small group of muscles and it is a burst movement so that the signal itself is weak when compared to the signals generated by other gestures. In such conditions it becomes harder to differentiate from the rest state. During the transition between rest and a gesture, the feature value increases or decreases slowly since it is the standard deviation of the signal in a window of frames. Therefore, there are some frames during the transition whose features can be similar to gesture 5. While the current feature set leads to false positives during transition periods (other gestures classified as double), these false positives have very short durations and are easily removed in post-processing, therefore not affecting the final classification performance. Despite these errors, gesture 5 is correctly classified when it appears isolated. In Fig. 7, the raw signal output of the LSTM model is shown in the same conditions and for the same sequence. There is considerably less classification noise and less false positives.

Table 2. Framewise accuracy (FWA) and detection accuracy (DA) of the classification models, as tested on the UC2018 DualMyo synthetic sequences and the NinaPro DB5 data set subset.

Model	Split	DualMyo						Ninapro DB5					
		FWA	DA	Outcomes				FWA	DA	Outcomes			
				TP	MC	FP	FN			TP	MC	FP	FN
FFNN	Train	96.94	98.51	454	4	0	1	98.76	100.00	250	0	0	0
	Val	96.33	98.55	151	1	0	2	93.75	96.47	75	2	0	0
	Test	96.65	99.34	151	0	0	1	90.82	90.82	81	5	0	1
RNN	Train	95.44	98.90	455	1	0	0	96.72	98.74	251	1	0	1
	Val	94.73	98.97	152	1	0	0	90.33	84.96	76	6	1	5
	Test	95.31	99.66	152	0	0	1	91.59	87.90	78	5	0	3
GRU	Train	97.00	99.78	456	1	0	0	96.15	98.81	250	1	0	2
	Val	95.75	99.64	152	0	0	0	91.60	91.28	83	4	1	0
	Test	96.14	100.00	152	0	0	0	92.07	87.73	78	6	0	3
LSTM	Train	95.99	99.18	455	1	0	0	95.52	98.10	250	1	0	2
	Val	94.87	98.31	152	1	0	0	90.83	88.05	78	6	0	2
	Test	95.32	99.33	152	0	0	0	90.82	86.11	76	5	0	4

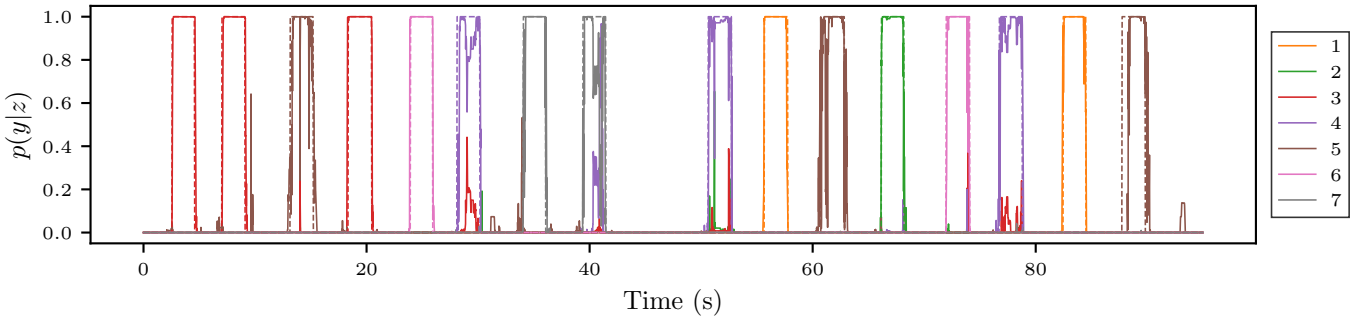

Fig. 6. Raw output of the static model on the first sequence of UC2018 DualMyo's test set and the corresponding targets in dashed lines.

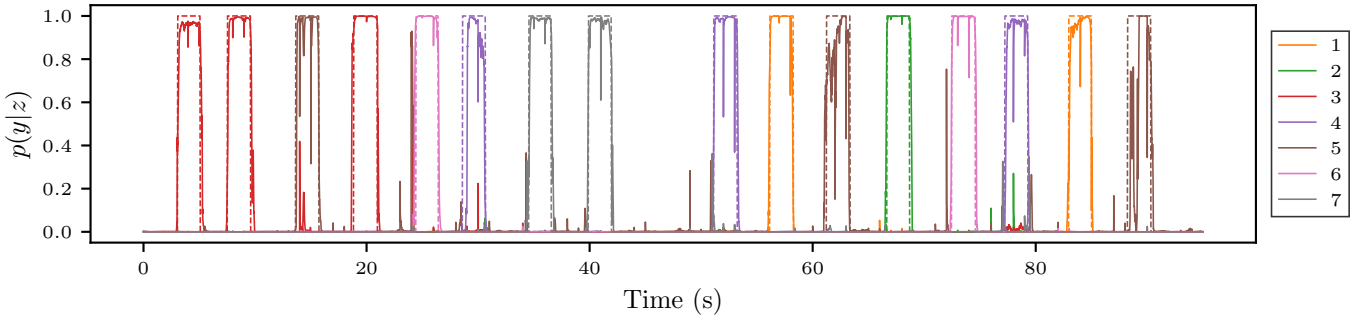

Fig. 7. Raw output of the LSTM model on the first sequence of UC2018 DualMyo's test set and the corresponding targets in dashed lines.

4. Conclusion

In this study we proposed the use of an LSTM neural network to improve the online classification of hand gestures from EMG signals acquired from the forearm muscles. The results obtained in the UC2018 DualMyo and the NinaPro DB5 data set were compared with a static model (FFNN) and other dynamic models (RNN and GRU). The proposed performance index, the gesture detection accuracy, demonstrated to be a good indicator to evaluate the performance of the model during online classification. It can be concluded that the training and inference time of the dynamic models (LSTM and GRU) is much smaller than for the RNN and the FFNN. In terms of accuracy, it can be concluded that results are similar for both static and dynamic models, with the GRU and LSTM presenting a slightly better performance.

Further optimization should be done in both models in order to decrease the number of parameters to make it better suited to embedded systems. Additionally, the methodology should be validated in larger data sets, which should also use other types of signals.

Acknowledgements

This work was supported in part by the Portuguese Foundation for Science and Technology (FCT), SFRH/BD/105252/2014, COBOTIS (PTDC/EME-EME/32595/2017) and Portugal 2020 project POCI-01-0145-FEDER-016418 by UE/FEDER through the program COMPETE2020.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. URL: https://www.tensorflow.org/. software available from tensorflow.org.

Bengio, Y., Simard, P., Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 5, 157–166. doi:10.1109/72.279181.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) URL: http://dx.doi.org/10.3115/v1/D14-1179, doi:10.3115/v1/d14-1179.

Chollet, F., et al., 2015. Keras. https://github.com/fchollet/keras.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9, 1735–1780. URL: http://dx.doi.org/10.1162/neco.1997.9.8.1735, doi:10.1162/neco.1997.9.8.1735.

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, 1–18. URL: https://doi.org/10.1371/journal.pone.0206049, doi:10.1371/journal.pone.0206049.

Ju, Z., Liu, H., 2014. Human Hand Motion Analysis With Multisensory Information 19, 456–466.

Ketykó, I., Kovács, F., Varga, K.Z., 2019. Domain adaptation for semg-based gesture recognition with recurrent neural networks. CoRR .

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. CoRR abs/1412.6980.

Koch, P., Phan, H., Maass, M., Katzberg, F., Mertins, A., 2018. Recurrent neural network based early prediction of future hand movements, in: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 4710–4713. doi:10.1109/EMBC.2018.8513145.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444. doi:10.1038/nature14539.

Paleari, M., Di Girolamo, M., Celadon, N., Favetto, A., Ariano, P., 2015. On optimal electrode configuration to estimate hand movements from forearm surface electromyography. Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS 2015-Novem, 6086–6089. doi:10.1109/EMBC.2015.7319780.

Phinyomark, A., Phukpattaranont, P., Limsakul, C., . Feature reduction and selection for EMG signal classification 39, 7420–7431. doi:10.1016/j.eswa.2012.01.102.

Pizzolato, S., Tagliapietra, L., Cognolato, M., Reggiani, M., Müller, H., Atzor, i.M., 2017. Comparison of six electromyography acquisition setups on hand movement classification tasks. Plos One doi:10.1371/journal.pone.0186132.

Samadani, A., 2018. Gated recurrent neural networks for emg-based hand gesture classification. a comparative study, in: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 1–4. doi:10.1109/EMBC.2018.8512531.

Simao, M., Gibaru, O., Neto, P., 2019. Online recognition of incomplete gesture data to interface collaborative robots. IEEE Transactions on Industrial Electronics , 1–1doi:10.1109/TIE.2019.2891449.

Simão, M., Neto, P., Gibaru, O., 2018. Uc2018 dualmyo hand gesture dataset. URL: https://doi.org/10.5281/zenodo.1320922, doi:10.5281/zenodo.1320922.

Simão, M.A., Neto, P., Gibaru, O., 2016. Unsupervised gesture segmentation by motion detection of a real-time data stream. IEEE Transactions on Industrial Informatics in press, 1–1. doi:10.1109/TII.2016.2613683.

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 Recognition Letters doi:https://doi.org/10.1016/j.patrec.2017.12.005.

Wolf, M.T., Assad, C., Stoica, A., You, K., Jethani, H., Vernacchia, M.T., Fromm, J., Iwashita, Y., 2013. Decoding static and dynamic arm and hand gestures from the JPL BioSleeve. IEEE Aerospace Conference Proceedings doi:10.1109/AERO.2013.6497171.

Xie, B., Li, B., Harland, A., 2018. Movement and gesture recognition using deep learning and wearable-sensor technology, in: Proceedings of the 2018 International Conference on Artificial Intelligence and Pattern Recognition, ACM, New York, NY, USA. pp. 26–31. URL: http://doi.acm.org/10.1145/3268866.3268890, doi:10.1145/3268866.3268890.