Supplementary Material S4: Materials and Methods

Surface electromyographic data acquisition and processing:
Amplitude of the sEMG signal measured in microvolts was converted by the armband into a unitless signal number, a proxy for muscle activity. Normalization of the sEMG signal was performed between tasks, students, and weeks to remove the effect of variations in muscle activity by dividing all activity-related sEMG signals by the mean sEMG signal detected with the hands in the neutral position. The starting point for collecting sEMG data was based on changes in acceleration along the y-axis (perpendicular to the surgical workstation) when change in average acceleration over a window of 0.5 seconds was larger than a set threshold. This threshold was selected heuristically through trial and observation of the sEMG and acceleration signals. Subsequently, a first order Butterworth low-pass filter with a 1 Hz cutoff frequency was applied to each channel of EMG data. The 8 EMG signals were concatenated in a data matrix with 8 columns, one column for each EMG sensor, and n rows, for n sampling points, i.e. the n data points recorded by each sEMG sensor at each trial. In this work, “trial” refers to each time a student completed a task. The mean sEMG signal for each sensor was calculated for each task, student, and week.

Machine Learning:
Mean Absolute Value was calculated by averaging the absolute values of the signal’s amplitude using the following equation:

\[ MAV = \frac{1}{N} \sum_{n=1}^{N} |X_n| \]

Similar to MAV, the RMS feature represents the average of sEMG signal’s amplitude and was calculated using
where $X_n$ was the sampling point and $N$ was the number of samples in the moving window.\textsuperscript{3,8} Variance of the signal was related to the deviation of the sampling points from their average $\bar{x}$ and is the mean value of the square of these deviations,\textsuperscript{3} calculated by:

\[
VAR = \frac{1}{N-1} \sum_{n=1}^{N} (x_n - \bar{x})^2
\]

Waveform length was derived by summation over the numerical derivative of the samples\textsuperscript{3,8} and was calculated using:

\[
WL = \sum_{n=1}^{N} |X_{n+1} - X_n|
\]

The TD features were extracted on a moving window over the data from each sEMG sensor. In this work, a sliding window of size 8 samples equaling 0.16 seconds was selected. As a result, for each feature, a matrix of $m \times 8$ was formed, where $m$ was the number of extracted features for each sEMG sensor and 8 was the number of sensors. Such feature matrices were obtained for all the trials of the four tasks, or classes. For each feature, the feature matrices were concatenated in one matrix with eight columns, each containing data from one of the sensors, to compute the mean and standard deviation of the entire dataset. These parameters were calculated for each column of the concatenated matrix. The obtained mean and standard deviation values were then utilized to standardize each column of the feature matrices; elementwise subtraction of the mean and division by the standard deviation obtained from their corresponding columns of the concatenated matrix. The TD features were treated as time series as their changes over time defined the characteristics of each surgical task. Each trial of each task had a different length.
depending on task completion time for each student and task each week. For the data to be input to the classification algorithm, the feature matrices obtained from the trials were required to have the same dimensions. Therefore, the length of the longest trial among all the classes or surgical tasks was set as the reference length for each feature. To make the lengths of the rest of the trials equal to the reference length, zeros were added at the end of the trials in a zero padding process. The prepared feature matrices, obtained from each student completing each task at any week, were then input to a classifier. The classifier was composed of two identical one-dimensional layers of a convolutional neural network (CNN) and three identical layers of a long short-term memory (LSTM) network. The model was implemented in Keras, a deep learning library with TensorFlow. The hyperparameters of the model were customized for each feature to achieve the highest classification performances (Table 1). For all features, the convolution layers had kernel of size 6 with a stride of 1 and padding = “same” to ensure the output size was the same as the input size. Rectified Linear Unit (ReLU) activation function and maximum pooling function, with pool size of 30, followed by batch normalization and 20% dropout were applied at the end of each convolution layer. A dropout of 20% was also implemented after the first two LSTM layers. A fully connected layer with Softmax output activation function was implemented as the very last layer of the model. Categorical Cross entropy was the loss function, and an Adam optimizer, with adaptive learning rate, were used for training the model. Five-fold cross validation was applied to evaluate the performance of the proposed model in classifying the TD features into the four tasks. At each fold, 80% of the trials were randomly selected for training the model and the rest 20% were used for validation. Precision, recall, and F1-score were calculated using sklearn.metrics.precision recall fscore support function, by setting average = ’macro’. Accuracy was defined as true positive+ true negative instances/total instances (true
positive+ true negative+ false positive+ false negative). Precision was defined as true positive instances/total predicted positive instances (true positive + false positive). Recall, or sensitivity, was defined as true positive instances/total actually positive instances (true positive + false negative). F1 score was defined as the harmonic mean of recall (sensitivity) and precision and represented the model’s comprehensive performance.
Tables

Table 1: Hyperparameters for the Convolutional Neural Network (CNN)-Long Short-Term Memory (LSTM) classifier model

<table>
<thead>
<tr>
<th>MAV(^a)</th>
<th>Batch Size</th>
<th>Epochs</th>
<th>CNN Output Filters</th>
<th>LSTM Output Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAV(^a)</td>
<td>3</td>
<td>80</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>RMS, VAR, WL(^b)</td>
<td>3</td>
<td>100</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

\(^a\) MAV - Mean Absolute Value

\(^b\) RMS - Root Mean Square, VAR - Variance, WL - Waveform length
References


Published 2021 Feb 25. doi:10.7717/peerj-cs.379
