New Interfaces for Classifying Performance Gestures in Music

Chris Rhodes[1][0000-0002-1899-8222], Richard Allmendinger[2][0000-0003-1236-3143], Ricardo Climent[3][0000-0002-0484-8055]

[1] NOVARS Research Centre, University of Manchester, UK
Chris.rhodes@manchester.ac.uk

[2] Alliance Manchester Business School (AMBS), University of Manchester, UK
Richard.allmendinger@manchester.ac.uk

[3] NOVARS Research Centre, University of Manchester, UK
Ricardo.climent@manchester.ac.uk

Abstract. Interactive machine learning (ML) allows a music performer to digitally represent musical actions (via gestural interfaces) and affect their musical output in real-time. Processing musical actions (termed performance gestures) with ML is useful because it predicts and maps often-complex biometric data. ML models can therefore be used to create novel interactions with musical systems, game-engines, and networked analogue devices. Wekinator is a free open-source software for ML (based on the Waikato Environment for Knowledge Analysis – WEKA - framework) which has been widely used, since 2009, to build supervised predictive models when developing real-time interactive systems. This is because it is accessible in its format (i.e. a graphical user interface – GUI) and simplified approach to ML. Significantly, it allows model training via gestural interfaces through demonstration. However, Wekinator offers the user several models to build predictive systems with. This paper explores which ML models (in Wekinator) are the most useful for predicting an output in the context of interactive music composition. We use two performance gestures for piano, with opposing datasets, to train available ML models, investigate compositional outcomes and frame the investigation. Our results show ML model choice is important for mapping performance gestures because of disparate mapping accuracies and behaviours found between all Wekinator ML models.

Keywords: Interactive Machine Learning, Wekinator, Myo, HCI, Performance Gestures, Interactive Music, Gestural Interfaces

1 Motivation

Interactive music is an artform that uses gestural interfaces to build new instruments [1] – or augment existing ones [2] – to generate a musical output. A gestural interface is a device which captures physical gestures to control digital systems. Typically, they detect biometric and inertial measurement unit (IMU) data. Through their use, novel

forms of human-computer interaction (HCI) can be explored in music composition and performance.

In the concise history of interactive music, technological developments made with gestural interfaces have always evoked a creative response. In 1917, the Theremin permitted people to create music with novel actions [3]. In the 1970s, the analogue synthesiser allowed users to affect parameters of sound through interacting with modules [4]. In 1987, Jon Rose [5] developed a violin bow (K-bow), which used the Musical Instrument Digital Interface (MIDI) protocol to digitise violin performance. In 1993, Atau Tanaka et al. [6] created Sensorband; an ensemble investigating how wearable sensors could affect the digital signal processing (DSP) of real-time sound generation, showing music composition could be embodied. In recent years, electroencephalography has been used as a brain-computer interface to further investigate the embodiment of music and drive music composition [7]. However, as the interface uses several electrodes attached to the scalp, it is not ideal for interactive music composition. This is because of the limited mobility of the interface. However, in 2015, the Myo armband [8] offered a mobile interface for improved gestural control when investigating interactive music. This allowed researchers to access two forms of raw biometric data: IMUs and Electromyographic (EMG). Although access to IMU data offered by the Myo is not novel, EMG data is useful because it measures electrical activity of skeletal muscles, allowing the user to map muscular data to digital systems.

Previous literature using the Myo interface has seldom explored the application of ML to process data, but rather conditional statements [9]. Although conditional statements are a feasible way of using interface data for music composition, they are not efficient because performance gestures can be misclassified if an ML approach to gesture classification is not adopted [9]. An ML approach would therefore be more accurate when interacting with music systems and evoking a sonic output. It is possible to realise this via an interactive ML software called Wekinator [10].

Wekinator is a ML environment built on WEKA. It is useful because it allows for real-time data input from gestural interfaces and has a simplified GUI for configuring ML models, providing a better accessible tool for HCI applications. Previous research has used Wekinator for the classification of instrument articulations [11], mapping sound from colour within virtual reality [12] and duetting with an LED gestural interface [13]. However, there is a lack of study regarding the classification of performance gestures in Wekinator with an informed-model approach; only a demonstrative (trial-and-error) model training method [11]. There is also a lack of literature on model evaluation when using EMG data to train ML models in Wekinator.

In this work we provide a solution to the problem that there are different ways to process and create ML models from performance gestures using the Myo interface within Wekinator. We also address that EMG data can be used to create bespoke physical gestures to interact with music systems, instead of research limitations surrounding the use of pre-defined gestures for music research via the Myo Software Development Kit (SDK) [9, 14]. It seeks to investigate which models are the most useful for predicting a sonic output for use in interactive music performance. If solved, this will improve knowledge regarding the use the Myo interface (plus similar interfaces) with Wekinator when choosing an ML model.

Our results establish that different ML models are more effective when chosen based on performance gesture characteristics and data type for model training. Our results also show model choice has an impact on controlling a musical output. The next section of this paper will detail technical aspects of the gestural interface we are using (the Myo armband), followed by ML models offered by Wekinator, the study methodology, results and conclusions derived.

2 Gestural interfaces and machine learning

Gestural Interfaces allow us to access raw data, representing performance gestures, and build ML models. The gestural interface we use for this investigation is the Myo armband.

(a) (b)

Fig. 1. (a) The Myo armband showing 8 channels of EMG electrodes. (b) A photo of the standardized placement of the Myo armband when collecting raw IMU and EMG data.

2.1 Gestural interface: The Myo armband

Technical specification. The Myo armband is a gestural interface which allows us to access raw IMUs and 8-channels of 8-bit EMG data. It communicates raw data via Bluetooth and streams IMU data at 50Hz and EMG data at 200Hz [9]. It communicates all data to a Bluetooth dongle with a distance of <15m [15]. Data is taken from the Myo library within the SDK created by Thalmic Labs [14]. The EMG data provided by the Myo is unique to the interface. This is because a gestural interface offering access to raw EMG data is hard to acquire on the consumer market.

2.2 Raw transmitted data

The Myo allows access to two forms of biometric information: IMUs and EMG data. IMU data from the Myo SDK has three main parameters for measuring orientation: acceleration, gyroscope and quaternions. Acceleration and gyroscope data parameters use 3 axes (X, Y, Z) to measure orientation. However, quaternions use 4 axes to do so

(X, Y, Z, W). The Myo has 8 individual electrodes for measuring EMG activity around the arm, as seen in Figure 1a.

2.3 Myo usage issues

Data retrieved from the Myo is subject to issues affecting data validity. These are: The placement of the Myo armband and user calibration.

Calibration. Because muscle activity in a Myo user's arm is as unique as a fingerprint [16], the Myo must be calibrated by the user so that data is reliable. If the Myo is not calibrated before use, the data collected will not be accurate as measurement is skewed. An example of inaccuracy is not providing a point of origin or data parameter limits for IMU data.

Placement. Careful placement of the Myo must be observed and standardised. See Figure 1b. This is because incorrect placement will skew all EMG results. A change in rotation of the Myo, between future participants or users, will change the orientation of electrodes (measuring EMG data) and therefore affect data or system validity.

2.4 Machine learning: Wekinator

Wekinator was developed by Rebecca Fiebrink in 2009 [10]. It is a GUI built on the WEKA framework and contains three ML model types for input data:

1. **Continuous:** (i) Linear/Polynomial Regression, (ii) Neural Network.
2. **Classifiers**: (i) K-Nearest Neighbor, (ii) AdaBoost.M1, (iii) Decision Tree (J48), (iv) Support Vector Machine (SVM), (v) Naive Bayes, (vi) Decision Stump.
3. **Dynamic Time Warping (DTW)**

3 Methodology

The methodology behind evaluating the efficacy of Wekinator for predicting sonic output via actioning performance gestures is as follows.

3.1 Planning performance gestures for model testing

Performance gestures chosen for predicting a music output were selected based on their ability to augment piano practice. Using an instrument would provide an initial focus for the application of gesture prediction and interactive music composition. Both gestures can be considered an extended technique; meaning a non-orthodox method of playing the instrument. For ease of understanding the results, we use a single Myo worn on the right arm. To train different ML models, we use two different performance gestures with opposing datasets. This is because continuous models,

classifiers and DTW algorithms process gestural data in a very different way. For example, a performance gesture that travels between two spatial positions on the y-axis – states y_1 and y_2 – can be considered either a continuous or static gesture. This is because we may wish to measure the metric of the two states in different ways. From a data perspective, this would be represented as either a series of floating-point values (continuous position) or a single integer value (classifier position).

However, both states combined (to a single unifying value) can be considered a necessary input for a DTW model. This is because a DTW algorithm is a continuous model that predicts a performance gesture - per iteration - and not as a stream of floating-point values. Given the problem of deciding the metric, it is therefore clear that the performance context (for the application of model output) would be the principal factor when deciding a suitable model for training data. This culminated in the creation of two performance gestures for investigation:

1. **Right arm positioned above head (gesture 1).** This gesture involves extending the right arm out straight above the head. This gesture is measured using single parameter acceleration data (IMU) on the y-axis; it is also linear, meaning data variation will be low. Low variation in the data will make model prediction more accurate. As a result, model output (using this data) will closely align to model input during mapping. Therefore, this gesture is thus most apt for a continuous/DTW model. This is because continuous data analysis is integral to how each model makes an accurate prediction.

2. **Spread fingers and resting arm (gesture 2).** This gesture involves: (a) Fully extending fingers on the right-hand outwards (b) returning the arm to a limp resting position. Gesture position (b) was created as a tool to test the efficacy of the model predicting position (a) when switching between both states. As this gesture operates between two static (non-continuous) states it became clear a classifier model was most apt for this gesture. This gesture uses two EMG data parameters targeting electrodes 3 and 4 on the Myo interface (see Figure 1b).

3.2 Data acquisition and structuring

Raw data was structured from the Myo, worn on the right arm, via software built by the first author in the Max 8 development environment.[1] The software built routed the Myo C++ API information, from the Myo SDK, to Max 8. Two sets of software were built in Max 8: A program to acquire raw data from the Myo and a program to send filtered Myo data to Wekinator and then apply DSP. After receiving the raw data within Max, data was pre-processed, structured and then exported as a csv file. The first author performed all gestures for acquisition and testing.

[1] Max 8 is a GUI programming environment primarily used by artists and musicians. More information regarding Max 8 can be found here: https://cycling74.com/.

Acquisition. Raw data was recorded from the Myo SDK at 10ms per data entry. This was deemed accurate enough to see meaningful patterns in the data. Each gesture was recorded with 4 iterations over 16 seconds. A digital metronome was synchronised with the data acquisition software to keep timing when performing gestures.

Pre-processing and structuring. Data was pre-processed during acquisition by unpacking and structuring (labelling) each data type (IMU and EMG) and respectful parameters to an array (see Section 2.2 for a summary). After acquisition, the data was exported from the array to a csv file.

3.3 Data post-processing, visualisation and filtering

Data post-processing. EMG data needed to be processed via taking an average of each EMG parameter and then applying an absolute function. Using this function would make EMG data much clearer to use and visualise [17]. As IMU relies on vector movement for each axis (to show orientation), IMU data was not post-processed.

Data visualisation and analysis. The data visualisation process involved plotting the data within suitable graphs to spot any significant trends when performing gestures. Applying a preliminary analysis would show key data and begin to manually filter the data.

Data filtering. Data was manually filtered after collection to train ML models in Wekinator. This is because each physical gesture is unique. Therefore, different data parameters will best represent each gesture performance. For example, training a model to predict a music output via data detailing the placement of the arm above the head will not require EMG data but data regarding the orientation of the arm.

3.4 Training ML models in Wekinator

After filtering, training data was sent from a real-time Myo signal in Max 8 to each model input in Wekinator via Open Sound Control (OSC) and User Datagram Protocol (UDP). UDP was then used to send the OSC data packet to a specific port that Wekinator was listening to. All models were measured against performance gesture 1 and 2 (detailed in Section 3.1). Each gesture used c.8000 data entry examples to train ML models at a frequency of 10ms per example. See Table 1 for a detailed list of parameters used when training each ML model with both gestures. A model evaluation was used (provided by the Wekinator GUI) for each model type over both gestures. The evaluation method used was cross-validation, with 10 folds, and model training accuracy was also measured.

Table 1. Model types and parameters used when training all available models in Wekinator.

Model type	Model	Available model parameters in Wekinator and their settings	Data type/range
Continuous (soft limits)[2]	Neural Network	1 Hidden layer. 1 node per hidden layer.	[0,1] (float)
	Linear Regression	Linear inputs. No feature selection used. Colinear inputs removed.	
	Polynomial Regression	Polynomial exponent = 2. No feature selection used. Colinear inputs not removed.	
Classifier	K-Nearest Neighbor	Number of neighbors (k) = 1.	{1,2} (integer)
	AdaBoost.M1	Training rounds = 100. Base classifier = Decision tree.	
	Decision Tree (J48)	Model not customisable.	
	SVM	Kernel: Linear. Complexity constant: 1.	
	Naïve Bayes	Model not customisable.	
	Decision Stump	Model not customisable.	
DTW	DTW	Threshold for prediction set in GUI to be sensitive enough to predict output once per gesture iteration.	Single-fire

4 Results

4.1 Model evaluations

Continuous models. Results from training the NN, linear regression and polynomial regression models in Wekinator elucidated interesting results across both gestures.

The NN showed to be the most accurate model when mapping model prediction to input data for both gestures (see Table 2 for full list of model accuracies). An example of mapping accuracy with the NN can be seen in Figure 2. Results for continuous models also showed that NN and polynomial regression models apply curvature to the mapping of gesture input data, whereas the linear regression model applies a linear mapping. When looking at Figure 3, it is clear that the linear regression model maps best the movement of gesture 1. Observing this difference between the NN/polynomial model and regression model is significant for interactive music practice. This is because the rate at which data increases will create a disparate musical output when affecting the DSP of an audio signal.

A further observation also shows that data type used to train continuous models is more accurate than other data types. Table 2 clearly shows that IMU data (gesture 1) is more effective than using EMG data (gesture 2) to train continuous models. This is further elucidated when comparing model mapping in Figures 2 and 3. All continuous models averaged 8029 examples of input training data for both performance gestures.

[2] Meaning the maximum value set for the model range [0,1] can be exceeded.

Table 2. Accuracy of continuous models after training, measured in root mean square (where 0 is optimal).

Gesture no.	Model type	Training accuracy	Cross-validation (10 folds)
1	**NN**	**0.00**	**0.00**
	Linear Regression	0.04	0.04
	Polynomial Regression	0.04	0.04
2	**NN**	**0.12**	**0.12**
	Linear Regression	0.32	0.32
	Polynomial Regression	0.32	0.32

Classifier models. Results for the classifier models showed a varying level of accuracy when predicting gesture because of the data type used to train models. When looking at Table 3, we can see that gesture 1 is more accurate in model prediction than gesture 2. However, we can also observe that models trained with EMG data (gesture 2) show disparate levels of accuracy. In particular, the SVM and decision stump models perform the poorest, in comparison to all available classifier models. This is clear when looking at their erratic misclassification of gesture 2 in Figure 4. Results taken from performing gesture 1 reported a striking accuracy with all classifier models. They also showed that individual models display an earlier prediction than others. The decision tree and adaboost.M1 models reported (in unison) a prediction of 70ms (first instance) and 60ms (second instance) ahead of all other classifier models. All classifier models averaged 8022 examples of input training data across both performance gestures.

Table 3. Accuracy of classifier models after training, measured in percentage (where 100% is optimal).

Gesture no.	Model type	Training accuracy	Cross-validation (10 folds)
1	K-Nearest Neighbor	100%	100%
	AdaBoost.M1	100%	100%
	Decision Tree (J48)	100%	100%
	Naive Bayes	100%	100%
	SVM	100%	100%
	Decision Stump	100%	100%
2	K-Nearest Neighbor	99.36%	99.36%
	AdaBoost.M1	99.36%	99.36%
	Decision Tree (J48)	99.36%	99.36%
	Naive Bayes	98.86%	98.86%
	SVM	97.75%	97.57%
	Decision Stump	94.78%	94.78%

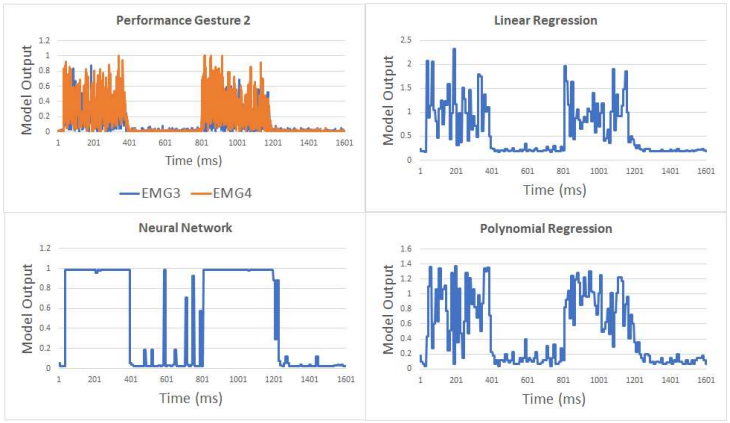

Fig. 2. A graph showing all continuous model outputs and their output activity when performing gesture 2 over a period of 16 seconds.

Dynamic Time Warping (DTW). Observations from using the DTW model indicate different measures of accuracy through different datasets. Observation of Figure 5 shows inaccuracy in predicting the onset of a performance gesture. It also shows inaccuracy when making several predictions of the same gesture iteration. This is because EMG data is a more complex dataset than IMU data. However, data taken suggests the DTW model is very effective in predicting gesture onsets with IMU data.

Weaknesses of using the DTW model are concerned with the GUI. When recording examples for training each model, the user is unable to know how long each training sample is. This is because the metric is not shown (unlike classifier and continuous models). The 'threshold' available in this mode, via the GUI, is also arbitrary. It is designed for the user to adjust model prediction sensitivity. However, the metric is inaccessible. The DTW model trained to predict gesture 1 used 67 examples of input data, whereas the model trained to predict gesture 2 used 336 examples.

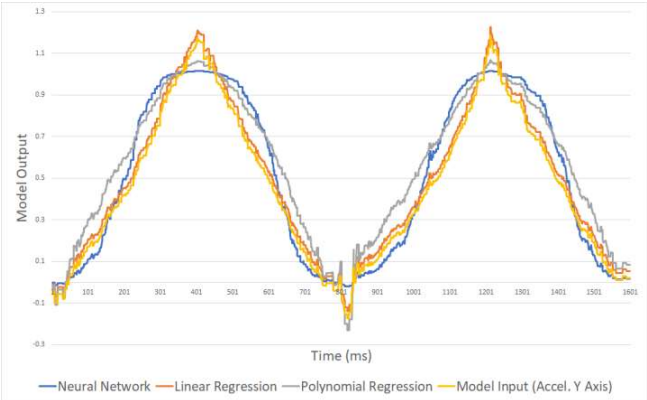

Fig. 3. A graph showing a comparison between all continuous models and their outputs vs model input over a period of 16 seconds when performing gesture 1.

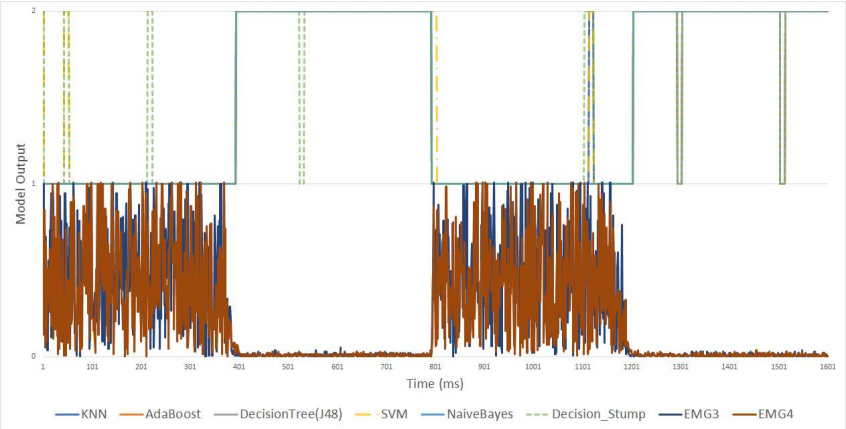

Fig. 4. A graph showing model output of all classifier models when performing gesture 2 over a period of 16 seconds.

4.2 Model output: Interactive music composition

Model results applied to interactive music composition can show: (a) Continuous models alter the data scaling (mapping) of gestures during model output (compare all model outputs in Figure 2), (b) classifier models run specific DSP algorithms by creating layers within gestures (where each layer is defined via an arbitrary number of integers), and (c) DTW models run a single DSP algorithm when predicting a gesture performed over time.

Continuous models are useful for the real-time control of music parameters. Albeit, each model is shown in this study to have differences in model output mapping.[2] This is important because interactive music is composed via applying multi-parametric control of DSP to music material [19]. The mapping (contour) from each model output will therefore affect sonic output. This is because data contour has shown to affect parametric control in two ways, smoothly and erratically. For example, creating a 1:1 mapping between a continuous model output (NN, LR, PR) and the generation of a sine wave will affect sonic output, disparately. A NN would smoothen the cycle between frequencies and the LR/PR would make the cycle behaviour erratic. However, other DSP events thrive off erratic movement in data (i.e. granular synthesis) and others a smoother movement (i.e. delay lines), due to their individual sonic aesthetic. This can be evidenced when investigating the timbral difference between a fixed music event (i.e. piano chord) and individual continuous model output.

[2] *Mapping* is an integral part of interactive music composition and computer music. It means to use a transformed parameter to control or affect another parameter [18]. Here, a model output to affect a DSP process (e.g. delay time, reverb, etc) of an audio signal.

Classifier models can be used to distinguish between stages of gesture performance. This is useful because a gesture can be used to trigger several pre-defined algorithms, at different stages of performing a gesture. This is useful for interactive music composition because it triggers a pre-defined music output, allowing the composer greater control over the music output. However, the output can be processed by using other datasets in real-time, once the algorithm (for each integer class) has been triggered. This can therefore augment music parameters of the running algorithm.

Alike classifier models, DTW models make use of a pre-defined DSP algorithm, albeit without being able to control parameters within the algorithm after triggering the event. As a single-fire event, with no parametric control over the DSP process, this model is only ideal for music composition within a system containing a rigid DSP architecture. Interactive music composition therefore benefits marginally from DTW.

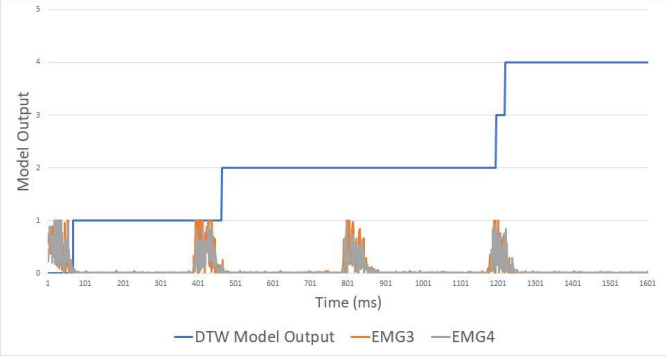

Fig. 5. Graph showing the accuracy of the DTW model when performing gesture 2 (4 times) in regular intervals over a period of 16 seconds.

5 Conclusions

It is apparent that model performance is unique to the data used to train each model. This is because data variation dictates prediction accuracy for ML models. This is evident when observing IMU data (low variation) and all model performances in this study. However, it is also clear that there are noticeable differences within ML model type; Variances in data mapping (continuous models) and model prediction strengths (classifier models). Music practitioners investigating interactive ML should therefore observe data variation when training ML models, model prediction strengths (see Tables 2 and 3) and pair model choice to desired music outcome (see Section 4.2).

Acknowledgements. This work was supported by the Engineering and Physical Sciences Research Council [2063473].

References

1. Tanaka, A.: Sensor-Based Musical Instruments and Interactive Music. In: The Oxford Handbook of Computer Music, pp. 233-257. Oxford University Press, USA (2011).
2. Miranda, E., Wanderley, M.: New Digital Musical Instruments: Control and Interaction Beyond the Keyboard. 1st edn. A-R Editions, USA (2006).
3. Hayward, P.: Danger! Retro-Affectivity! The Cultural Career of the Theremin. Convergence 3(4), 28-53 (1997).
4. Finamore, E.: A Tribute to the Synth: How Synthesisers Revolutionised Modern Music. https://www.bbc.co.uk/programmes/articles/3ryZCdlXtpkNG3yRl3Y7pnh/a-tribute-to-the-synth-how-synthesisers-revolutionised-modern-music, last accessed 28/7/2019.
5. Rose, J.: K-bow: The Palimpolin Project. http://www.jonroseweb.com/e_vworld_k-bow.html, last accessed 28/7/2019.
6. Bongers, B.: An Interview with Sensorband. Computer Music Journal 22(1), 13-24 (1998).
7. Wu, D. et al.: Music Composition from the Brain Signal: Representing the Mental State by Music. Computational Intelligence and Neuroscience 2010, 1-6 (2010).
8. Jackson, B.: Thalmic Labs Myo Armband Hits Consumer Release, for Sale on Amazon. https://www.itbusiness.ca/news/thalmic-labs-myo-armband-hits-consumer-release-for-sale-on-amazon/54056, last accessed 26/7/2019.
9. Nymoen, K. et al.: MuMyo – Evaluating and Exploring the MYO Armband for Musical Exploration. In: Proceedings of the International Conference on New Interfaces for Musical Expression, vol. 2015, pp. 215-218. NIME, USA (2015).
10. Fiebrink, R, Trueman, D, Cook, P, R.: A Meta-Instrument for Interactive, On-the-fly Machine Learning. In: Proceedings of the International Conference on New Interfaces for Musical Expression, vol. 2009, pp. 280-285. NIME, USA (2009).
11. Fiebrink, R, Schedel, M.: A Demonstration of Bow Articulation Recognition with Wekinator and K-Bow. In: Proceedings of the International Computer Music Conference (ICMC), vol. 2011, pp. 272-275. ICMC, UK (2011).
12. Santini, G.: Synesthesizer: Physical Modelling and Machine Learning for a Color-Based Synthesizer in Virtual Reality. In: Mathematics and Computation in Music, vol. 11502, pp. 229-235. Springer, Spain (2019).
13. Hantrakul, L, Kondak, Z.: GestureRNN: A Neural Gesture System for the Roli Lightpad Block. In: Proceedings of the International Conference on New Interfaces for Musical Expression, vol. 2018, pp. 132-137. NIME, USA (2018).
14. North. Myo Connect, SDK and Firmware Downloads. https://support.getmyo.com/hc/en-us/articles/360018409792-Myo-Connect-SDK-and-firmware-downloads, last accessed 23/07/2019.
15. North. What is the Wireless Range of the Myo?. https://support.getmyo.com/hc/en-us/articles/202668603-What-is-the-wireless-range-of-Myo-, last accessed 23/07/2019.
16. North, Creating a Custom Calibration Profile for Your Myo Armband. https://support.getmyo.com/hc/en-us/articles/203829315-Creating-a-custom-calibration-profile-for-your-Myo-armband, last accessed 26/07/19.
17. Soares, S. B. et al.: The Use of Cross Correlation Function in Onset Detection of Electromyographic Signals. In: ISSNIP Biosignals and Biorobotics Conference, vol 2013, pp. 1-5. IEEE, Brazil (2013).
18. Hunt, A., Wanderley, M. M.: Mapping Performer Parameters to Synthesis Engines. Organised Sound 7(2), 97-108 (2002).
19. Winkler, T.: Composing Interactive Music: Techniques and Ideas Using Max. 1st edn. MIT Press, USA (1998).