17th Int. Conf. on Acc. and Large Exp. Physics Control Systems ICALEPCS2019, New York, NY, USA JACoW Publishing
ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA011

IMPROVING GESTURE RECOGNITION WITH MACHINE LEARNING: A COMPARISON OF TRADITIONAL MACHINE LEARNING AND DEEP LEARNING

R. Bacher, Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

Abstract

Meaningful gesturing is important for an intuitive human-machine communication. This paper deals with methods suitable for identifying different finger, hand and head movements using supervised machine learning algorithms. On the one hand it discusses an implementation based on the k-nearest neighbour classification algorithm (traditional machine learning approach). On the other hand it demonstrates the classification potential of a convolutional neural network (deep learning approach). Both methods are capable of distinguishing between fast and slow, short and long, up and down, or right and left linear as well as clockwise and counter-clockwise circular movements. The details of the different methods with respect to recognition accuracy and performance will be presented.

INTRODUCTION

Controlling video games through a gaming console or acting in a virtual or mixed reality environment recognizing arm, head and body motions all lead to popular and intuitive interface features currently in common use. Even in the case of industrial applications, novel interaction technologies are gaining in importance, e.g. to simplify quality assurance of manufacturing processes.

In the field of accelerators, hardware commissioning and maintenance use cases might profit from such novel interaction capabilities. For instance, wearing rough and dirty working gloves during cooling-water maintenance work is not adequate for touch sensitive devices. Interacting via hand or arm gestures might be a better choice.

Today's users of accelerator control applications have developed intuitions based on click-like interactions through a mouse or a touch-sensitive device. Both interfaces provide high reliability, a very accurate pointing capability and standardized user actions normally associated with graphical widgets. Therefore, any new interaction capability such as gesture recognition will only be accepted by the users if it provides comparable or even better handiness, ease-of-use, and reliability to click-like interactions.

This paper discusses methods and their implementations aiming at improving the quality and reliability in recognizing gestures based on different finger, hand and head movements using machine learning algorithms. It compares a traditional machine learning (TML) and a deep learning approach (CNN) including a non-linear regression for extracting the features of the movement and a k-nearest neighbour method for movement classification using memorized training data (traditional machine learning) and a trained convolutional neural network for classification (deep learning).

GESTURE TYPES

The consumer market provides various devices capable of recognizing 2D/3D spatial gestures including hand-gestures, hand- or arm-gestures and 3-axis (yaw, pitch, and roll) head movements (smart glasses). The native or device-specific gestures such as "Closed-Hand", "Fingers-Spread" or "Nodding" can be combined with preceding or following linear movements or rotations including

- Horizontal: left, right
- Vertical: upward, downward
- Diagonal: upward-left, upward-right, downward-left, downward-right
- Circular: clockwise, counter clockwise

In addition, each of these enriched gestures can be performed as a long-or-short and slow-or-fast linear movement or rotation which is projected to a virtual plane in front of the user.

GESTURE RECOGNITION WORKFLOW

The workflow to predict a movement associated with an enriched gesture consists of two distinct phases: a training phase and a real-time prediction phase. It is important that both the training and the real-time prediction phase use the same algorithms.

The input for both phases is continuously recorded position data of the user's input device in Cartesian (X, Y) coordinates resulting from finger, hand, arm or head linear movements or rotations. The continuous stream of sensor data is pre-processed to determine the position of the signal within a certain time window. Optionally the noise floor of the measurement can be reduced or obvious outliers can be removed. If properly centred within the time window, the cleansed data array is fed into the machine learning algorithms.

Traditional Machine Learning

The basis for the traditional approach is a well-engineered mathematical model (Eq. (1)) describing the movements by a set of representative parameters (features).

$$\begin{aligned}
&\mathbf{t \le t_{start}}: && \mathbf{f(t) = s_{start}} \\
&\mathbf{t_{start} < t < t_{end}}: && \mathbf{f(t) = s_{start} + \left(\left(\frac{s_{end} - s_{start}}{t_{end} - t_{start}}\right) * (t - t_{start})\right)} \quad (1) \\
&\mathbf{t \ge t_{end}}: && \mathbf{f(t) = s_{end}}
\end{aligned}$$

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA011

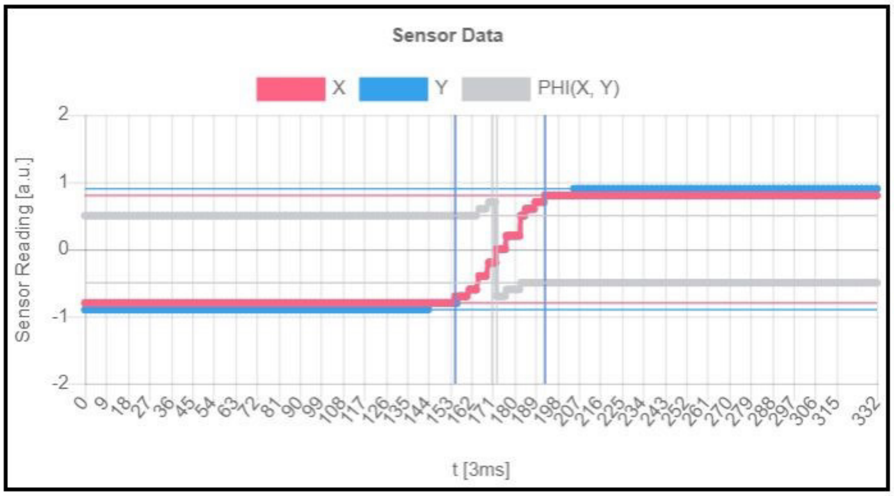

Figure 1: Recorded position sensor data (i.e. time series of 333 data points at intervals of 3 ms) associated with a Diagonal Upward-Right-Long-Fast movement.

Here, the parameters t_{start} and t_{end} mark the start and end time of the recognized gesture within the analysed user's finger, arm or head movement, respectively. Similar, s_{start} and s_{end} are indicating the start and end position of the gesture.

The features of a movement are extracted using a non-linear regression (Nelder-Mead method) to fit consecutive time series of sensor data (Fig. 1) to the pre-defined model.

The regression reduces the dimension of the gesture recognition task by two orders of magnitude and is performed for each Cartesian coordinate orientation (linear horizontal movement, linear vertical movement) separately. Fitting the polar angle PHI allows the identification of circular movements. If the regression algorithm converges and the parameters are confined within reasonable limits, the duration ($t_{end} - t_{start}$) and the length ($s_{end} - s_{start}$) of the gesture are calculated.

Comparing s_{start} and s_{end} of both horizontal and vertical movement allows distinguishing between linear (left / right horizontal, up / down vertical, left / right / up / down diagonal) and circular (clockwise / counter clockwise) movements.

During the training phase the extracted features of a sufficiently-sized set of training data are used to generate sets or clusters of learned data representing valid movement types (Long-Slow, Long-Fast, Short-Slow, Short-Fast). The resulting prediction quality of the trained model is verified with a verification data set.

The duration and length of the movement are used as input for a k-nearest neighbour analysis. This classification algorithm calculates the Euclidean distance between the predicted data point (duration / length) and each memorized learned data point. It searches the k-nearest neighbours to identify the learned data cluster the predicted data point belongs to (Fig. 2).

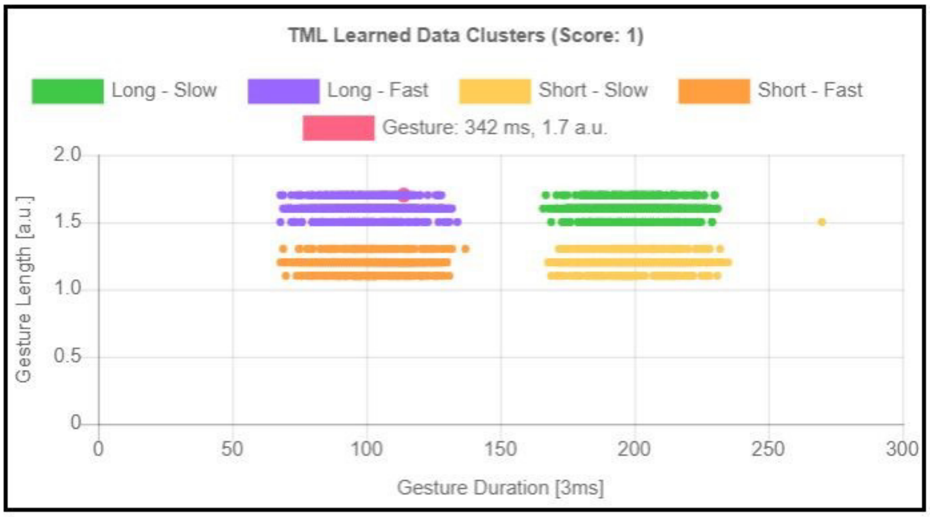

Figure 2: Predicted gesture duration / length and learned data clusters. The predicted movement (red dot) is classified as a Long-Fast movement.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA011

Table 1: Convolutional Neural Network

Layer (Filters)	Array Size
Input	1 x 333 x 2
Convolution (18)	1 x 331 x 18
ReLU Activation	1 x 331 x 18
Pooling	1 x 165 x 18
Convolution (36)	1 x 163 x 36
ReLU Activation	1 x 163 x 36
Pooling	1 x 81 x 36
Convolution (72)	1 x 79 x 72
ReLU Activation	1 x 79 x 72
Pooling	1 x 39 x 72
Fully Connected	1 x 1 x 1120
ReLU Activation	1 x 1 x 1120
Fully Connected	1 x 1 x 560
ReLU Activation	1 x 1 x 560
Softmax Activation	1 x 1 x 280
Output	1 x 1 x 40

Deep Learning

In contrast to the traditional approach deep learning does not require any preceding feature engineering or model building.

Data classification is performed by a multi-layer convolutional neural network well suited to classify time series data. Consecutive time sequences of sensor data (X, Y) are injected into the input layer. The input layer is followed by 3 convolution layers (CONV). Each layer extracts characteristic data features by moving small-sized two-dimensional filter arrays across the data arrays of the preceding layer all the while performing a convolution calculation providing corresponding feature maps. Applying a rectified linear activation function (ReLU) provides contrast enhancement of the data and pooling of adjacent neurons halves the size of the resulting data arrays but retains the most important information. The output of the last CONV layer is passed and flattened through 3 subsequent fully connected layers (FC) connecting every neuron of the previous layer to every neuron of the next. The FC layers provide binary classification. ReLU activation is subsequently applied one more time. Furthermore, the array size is reduced step-by-step. Finally, the output of the last FC layer is transformed into a probability distribution by applying a special activation function (Softmax) delivering the output predictions.

During the training phase a sufficiently-sized set of training data are used to train the convolutional neural network. The prediction quality is verified with verification data sets.

IMPLEMENTATION DETAILS

Both the implemented traditional machine learning and the deep learning algorithm are part of the Web2cHMI library [1]. Web2cHMI is a Web-based native user interface implementation for accelerator operations and maintenance applications in the context of the Web2cToolkit Web service collection. Web2cHMI provides gesture recognition and speech recognition capability. It is entirely coded in JavaScript capable of being processed locally by a standard Web-browser and does not rely on any server-side logic.

Algorithms

The k-nearest neighbour classification algorithm used in the traditional machine learning approach searches the 20 nearest data points (k = 20).

The convolutional neural network capability is based on the ConvNetJS JavaScript library [2]. Table 1 summarizes the parameters of the network layers.

Both the traditional machine learning and the deep learning algorithm are capable of distinguishing between 32 linear and 8 circular movements:

- Diagonal: Upward/Downward – Right/Left – Long/Short – Slow/Fast
- Horizontal: Right/Left – Long/Short – Slow/Fast
- Vertical: Upward/Downward – Long/Short – Slow/Fast
- Circular: Clockwise/Counterclockwise – Long/Short – Slow/Fast

Table 2: Movement Parameters

Parameter	Value
s_{start} (long signal)	-0.8
s_{end} (long signal)	0.8
s_{start} (short signal)	-0.6
s_{end} (short signal)	0.6
t_{start} (fast signal)	0.35 * n_{points}
t_{end} (fast signal)	0.65 * n_{points}
t_{start} (slow signal)	0.2 * n_{points}
t_{end} (slow signal)	0.8 * n_{points}

Data Sets

The analysis discussed in this paper is based on generated data sets. Each set contains all sorts of horizontal, vertical, diagonal and circular finger, hand or arm movements according to Eq. (1) parameterized by t_{start}, t_{end}, s_{start} and s_{end} (Fig. 1). Each movement is a 1s long time series of data points at intervals of 3 ms and consists of 333 data points (n_{points}). The data points are normalized ranging from -1.0 to 1.0. The characteristic parameters of the simulated movement are summarized in Table 2.

Duration ($t_{end} - t_{start}$) and length ($s_{end} - s_{start}$) of each gesture type are randomly (t_{jitter}, s_{jitter}) distributed values around a given mean value as given in Table 2. In addition, random noise (f_{noise}) is added to each individual function value. Besides data sets for training and verification, test data sets with different jitter and noise values have been generated for comparison purposes. The parameters of the generated test data sets are summarized in Table 3.

Table 3: Data Sets

Data Set Type	t_{jitter}	s_{jitter}	f_{noise}
Training / Verification	5%	5%	10%
Test 0 (Reference set)	5%	5%	10%
Test 1 (f_{noise} fixed)	2.5%	2.5%	10%
Test 2 (f_{noise} fixed)	7.5%	7.5%	10%
Test 3 (t_{jitter}, s_{jitter} fixed)	5%	5%	5%
Test 4 (t_{jitter}, s_{jitter} fixed)	5%	5%	15%

TRAINING AND VERIFICATION

Due to the inherent complexity of the deep learning algorithm a training step of the convolutional neural network requires considerably more time to be performed as in the case of the traditional machine learning algorithm.

To explore the training progress of both machine learning algorithms training has been performed with different numbers of training steps ($n_{training}$) and the prediction quality of the classification algorithms after training has been measured using 100 verification data sets.

The achieved training quality is summarized in Table 4. It lists the mean prediction probability and the best and worst individual prediction probability out of the 40 gesture types to be distinguished.

In all cases the deep learning approach performs better than the traditional machine learning algorithm.

Table 4: Training Quality

$n_{training}$	**Algorithm**	**Mean**	**Min**	**Max**
500	TML	0.936±0.017	0.50	1.00
	CNN	0.997±0.002	0.93	1.00
1000	TML	0.945±0.015	0.58	1.00
	CNN	1.000±0.000	1.00	1.00
1500	TML	0.949±0.015	0.61	1.00
	CNN	1.000±0.000	0.99	1.00

PREDICTION

Individual gestures are never identical. In order to explore the prediction power for similar but slightly differently performed gestures, two series of tests have been executed:

- Varying the jitter of the duration and length of the simulated gesture but keeping the noise floor of the position sensor data constant (f_{noise} = 10%),
- Varying the noise floor of the simulated position sensor data but keeping the jitter of the duration and length of the gesture constant (t_{jitter} = 5%, s_{jitter} = 5%).

As reference, the training and verification data set type has been used. Both types (TML, CNN) of algorithms have been trained through 1000 training steps. Each individual test includes 100 test steps. Table 5 summarizes the test results.

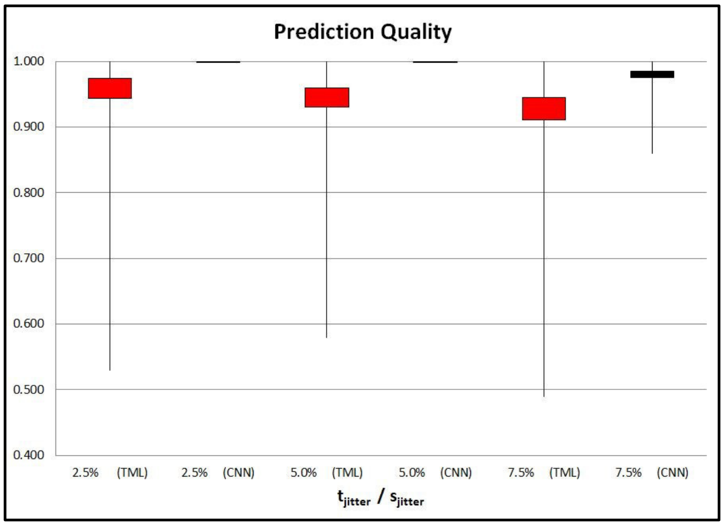

Figure 3: Prediction quality (see Table 5) as function of the jitter of the duration and length of the simulated gesture. The noise floor of the simulated position sensor data is kept constant.

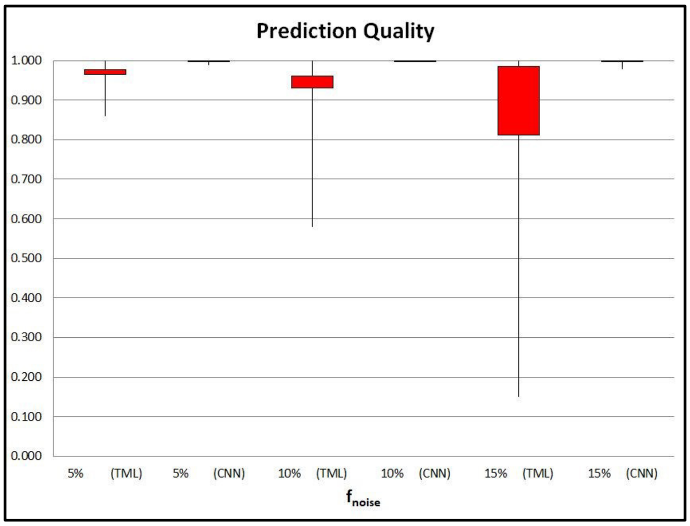

Figure 4: Prediction quality (see Table 5) as function of the noise floor of the simulated position sensor data. The jitter of the duration and length of the simulated gesture is kept constant.

Table 5: Prediction Quality

Test	Algorithm	Mean	Min	Max
Test 0	TML	0.945±0.015	0.58	1.00
	CNN	1.000±0.000	1.00	1.00
Test 1	TML	0.959±0.015	0.53	1.00
	CNN	1.000±0.000	1.0	1.0
Test 2	TML	0.928±0.017	0.49	1.0
	CNN	0.980±0.005	0.86	1.0
Test 3	TML	0.971±0.007	0.86	1.0
	CNN	1.000±0.000	0.99	1.0
Test 4	TML	0.849±0.037	0.15	1.0
	CNN	1.000±0.000	0.98	1.0

Compared to the traditional machine learning approach the deep learning algorithm is less sensitive to deviations from the standard movement data set (reference data set Test 0) used for training and provides in all tested cases a higher prediction quality (Fig. 3). In particular, the prediction power of the traditional machine learning algorithm degrades rapidly with increasing noise floor (Fig. 4).

CONCLUSIONS

The obvious advantage of the traditional machine learning approach is the short period which is needed to train a model. However, proper feature engineering has to be performed prior to the training and real time prediction phase. A too simple or insufficiently engineered model may result in a minor prediction quality and the algorithm may only accurately work if the data to be analysed do not differ too much from the data used for training. This deficit may be overcome by user-specific, individually trained models.

Similarly, the obvious disadvantage of the deep learning approach is its slow training behaviour. However, if the neural network is finally trained, the algorithm seems to be more robust with respect to deviations from the data sets used for training which makes individual training unnecessary. Feature engineering is not required. As a consequence the features of the gestures remain largely hidden. Finally, the deep learning approach provides a higher prediction quality compared to the traditional machine learning approach.

REFERENCES

[1] R. Bacher, “Augmented User Interaction”, in *Proc. PCaPAC’16*, Campinas, Brazil, Oct. 2016, pp. 16-20. doi:10.18429/JACoW-PCAPAC2016-WEUIPLIO01

[2] ConvNetJS, https://cs.stanford.edu/people/karpathy/convnetjs/