Classifying Human Manual Control Behavior Using LSTM Recurrent Neural Networks

Versteeg, Rogier; Pool, Daan M.; Mulder, Max

DOI
10.1109/THMS.2023.3327145

Publication date
2024

Document Version
Final published version

Published in
IEEE Transactions on Human-Machine Systems

Classifying Human Manual Control Behavior Using LSTM Recurrent Neural Networks

Rogier Versteeg, Daan M. Pool, *Member, IEEE*, and Max Mulder, *Senior Member, IEEE*

***Abstract*—This article discusses a long short-term memory (LSTM) recurrent neural network that uses raw time-domain data obtained in compensatory tracking tasks as input features for classifying (the adaptation of) human manual control with single- and double-integrator controlled element dynamics. Data from two different experiments were used to train and validate the LSTM classifier, including investigating effects of several key data preprocessing settings. The model correctly classifies human control behavior (cross-experiment validation accuracy 96%) using short 1.6-s data windows. To achieve this accuracy, it is found crucial to scale/standardize the input feature data and use a combination of input signals that includes the tracking error and human control output. A possible online application of the classifier was tested on data from a third experiment with time-varying and slightly different controlled element dynamics. The results show that the LSTM classification is still successful, which makes it a promising online technique to rapidly detect adaptations in human control behavior.**

***Index Terms*—Classification, cybernetics, human–machine systems, manual control, neural networks.**

I. Introduction

CYBERNETICS theory describes the systematic adaptation of a human controller (HC) to task variables such as the display and controlled element (CE) dynamics [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11]. Gaining understanding of human manual control behavior is challenging due to the nonlinear, adaptive, and time-varying characteristics of the HC. Although cybernetics is widely used in studying manual control and tuning a plethora of human–machine systems [12], [13], the theory and modeling approaches lag behind the rapid advancements in technology (e.g., haptics and multimodal interfaces) and are limited in capturing the full range of human manual control [7].

Current models and tools are often restricted to using linear time-invariant model structures, have been validated mainly for compensatory tracking, are designed for offline use, and require large amounts of tracking data from many subjects. HC control behavior is mostly analyzed in the frequency domain, lumping all nonlinearities in a remnant signal, which is ill understood. State-of-the-art techniques to identify time-varying adaptive HC behavior either offline [14] or in real time, e.g., using wavelets [15], ARX models [16], or extended Kalman filters [17], are slow to converge and highly variable in detecting changes in behavior. For these techniques to lag, 10 s or more behind the actual change in HC behavior is not uncommon [16], [17]; making them faster to respond comes at the cost of more (faulty) modeled variations.

This all hampers attempts to create cooperative human–machine systems, such as in human–automation teaming and shared control systems [18]. Ideally, to safeguard stability, performance, and safety, these systems include components that monitor the dynamics of *both* the system to be controlled (the CE) and the human controlling (HC). Whereas techniques to identify changing CE dynamics, for instance to detect failures, is an active field of research (see [19] for an overview in aviation), the lack of progress in monitoring real-time human behavior is becoming a problem. This is partly due to the fact that the CE dynamics are more easily defined, and with that determined, than the HC dynamics.

Whereas the HC output can be easily measured (the control manipulator), the HC *input*(s) is not [20]. Especially in real-life manual control, such as driving a car, the HC can have multiple inputs (visual, somatosensory, etc.) and close an abundance of loops [21], and human dynamics quickly become extremely difficult to define and determine. This makes monitoring human behavior and the design and tuning of adaptive human–machine systems to support humans such a challenge: we do not know enough about what they are actually doing. There is a clear need for alternative methods to determine (the adaptation of) human manual control behavior [7].

Time-domain black box "pattern recognition" models have rarely been used in this field [3], [13]. Such techniques require little or no a priori information, assume no form of the HC dynamics, and can include nonlinearities which linear models cannot. When such a black box model would be able to capture unique patterns of HC behavior, it may be able to classify this behavior in a variety of tasks, may require less data, and may quickly detect time-varying behavior, possibly even in real time. To avoid well-known risks of black box models and trivial classification outcomes that do not generalize, knowledge from cybernetics theory is required to assess any black box models of HC behavior.

Most human behavior pattern recognition models use hand-picked statistical features, calculated from raw time-domain signals and then used as inputs to a classifier [22]. Many applications already exist, such as classification of driver style and distraction [23], [24], [25]. A disadvantage of using statistical

Manuscript received 20 February 2023; revised 4 August 2023 and 7 October 2023; accepted 20 October 2023. Date of publication 29 November 2023; date of current version 26 January 2024. This article was recommended by Associate Editor J. A. Adams. *(Corresponding author: Max Mulder.)*

The authors are with the Control and Simulation Section, Faculty of Aerospace Engineering, Delft University of Technology, 2629 HS Delft, The Netherlands (e-mail: rogierversteeg@hotmail.com; d.m.pool@tudelft.nl; m.mulder@tudelft.nl).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/THMS.2023.3327145.

Digital Object Identifier 10.1109/THMS.2023.3327145

properties, however, is that these properties are task specific and time consuming to tune [26]. Automated feature extraction techniques are developed [27], but require specific knowledge on the relevance of different statistical properties.

Research into neural network models capable of recognizing patterns from *raw* signals is a growing area of interest in time-series classification [22]. Especially, long short-term memory (LSTM) networks—designed specifically for sequence prediction problems—achieve superior performance in raw signal time-series classification, as used for driver activity recognition [26], [28], [29] and disease diagnosis [30], [31]. Hence, this particular technique may also be suitable for the classification of HC behavior based on raw manual control task signals.

This article describes the capability of a time-domain pattern recognition technique in HC behavior classification. The goal is not to model the HC dynamic behavior, but rather to determine what behavior the HC exhibits with the smallest possible amount of data. An LSTM recurrent neural network is applied to data from compensatory tracking tasks, classifying (adaptation of) HC behavior with two different CE dynamics. The study is performed by first training an LSTM network through supervised learning, mapping HC input–output relations by providing a labeled training dataset.

We selected a dataset that consists of human control input and output signals measured during compensatory tracking tasks with single-integrator (SI) or double-integrator (DI) CE dynamics [32]. With a compensatory display, the input to the HC is known. SI and DI dynamics are the most relevant in manual control, as many dynamic systems controlled by humans can be approximated by SI or DI dynamics, or dynamics that behave as an SI at low frequencies and as a DI at higher frequencies. Also, HC behavior is markedly different for these SI and DI CE dynamics [1], which should enable reliable optimization of classifier performance for various dataset configurations, to investigate effects of the observed window size (WS), scaling techniques, input signals, and amounts of training data on classifier performance.

This all serves the main purpose to maximize classification performance across multiple tracking experiments with the *smallest* amount of data, but also allows an investigation of what tradeoffs need to be made. Having a successful classifier for just small "snippets" of data would save computational effort, reduce the need for large experiments, and potentially also allow for an *online* use of the classifier.

The classifier's robustness across experiments is investigated using data from a second experiment, with different subjects, tracking task settings, and CE characteristics (different SI and DI dynamics gains) [33]. Can the classifier trained and validated on data from the first experiment be used in an experiment with different subjects and conditions? Finally, the best performing LSTM network obtained is applied on test data from a third experiment [16]. This experiment featured slightly different CE dynamics (close to SI and DI dynamics, but not equal), was performed in a different simulator, and featured a time-varying change in these CE dynamics somewhere halfway the tracking runs. This to investigate the network's capability on detecting time-varying human control behavior.

The rest of this article is organized as follows. Section II provides elementary background on tracking tasks and describes the three datasets, the data processing options, and the LSTM model structure. Section III discusses the approach to answer the main research questions. Section IV presents the results of our study. Discussion and recommendations are presented in Section V. Finally, Section VI concludes this article. Specifics on the LSTM implementation are included in the Appendix.

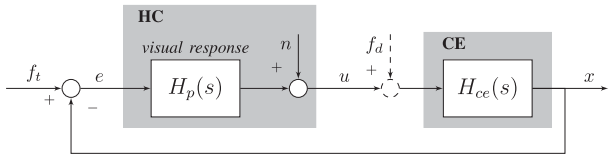


Fig. 1. Compensatory tracking task.

II. Method

A. Background: Compensatory Tracking Tasks

The compensatory target-following task (see Fig. 1) is investigated to classify (the adaptation of) human manual control behavior with SI and DI CE dynamics. The reader is referred to many studies describing this task [1], [3], [6], [7], [10], [11]. The HC closes the loop by responding to the only visual cue presented—the error e between the target signal (forcing function) f_t and the CE response x—with a control output u, to minimize the error. The linear part of the HC control response is described by $H_p(s)$; all nonlinearities and time variations are lumped into the remnant n [1], [7], [34], [35].

Many dynamic systems controlled by humans can be approximated through "pure" SI or DI controlled dynamics or as dynamics that behave as an SI at low frequencies and as a DI at frequencies above the break frequency ω_{ce}. An example is the aircraft pitch angle response to elevator inputs [32]:

$$H_{\text{ce}}(s) = K_{\text{ce}}/(s(s+\omega_{\text{ce}})) \tag{1}$$

Different values for the gain K_{ce} and break frequency ω_{ce} change the CE response x to HC control inputs u.

Forcing functions, i.e., the target f_t and disturbance f_d signals in Fig. 1, are used to "excite" the HC response, to assess control performance, and to allow for the identification of HC dynamics [7]. The disturbance signal f_d is generally added when both visual and motion cues are available, in order to dissect the visual and motion contributions to HC behavior [7]. Both f_t and f_d are often sum-of-sinusoids signals.

The HC dynamics response $H_p(s)$ when controlling SI and DI CE dynamics is well known to be modeled as [1], [7]

$$H_p(s) = K_p(T_L s + 1)e^{-s\tau_e} \tag{2}$$

with K_p, T_L, and τ_e the HC gain, lead compensation time constant, and lumped time delay, respectively. These parameters are task dependent and can be selected using the "verbal adjustments rules," based on experiments [1]. With SI dynamics, the lead time is zero; with DI dynamics, the lead is substantial (≈ 2 s). Here, we aim to capture not only these distinctive linear HC adaptations to CE dynamics, but also more subtle and less well-known nonlinear HC adaptations [34], [35], using an LSTM classifier.

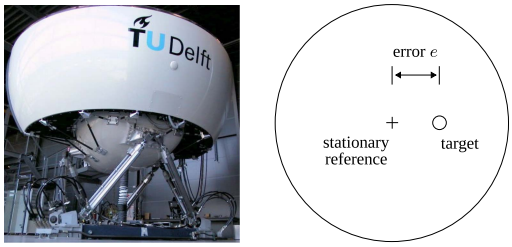


Fig. 2. TU Delft's SIMONA Research Simulator (left) and compensatory visual display (right), used in Experiments 1 and 2.

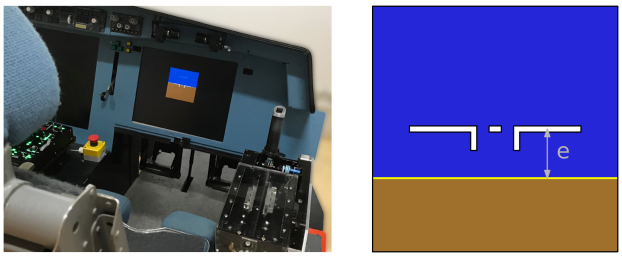


Fig. 3. TU Delft's Human–Machine Interaction lab (left) and compensatory pitch visual display (right), used in Experiment 3.

TABLE I
CONTROLLED ELEMENT DYNAMICS SPECIFICS

	CE	K_{ce} [−]	ω_{ce} [rad/s]	$f_{t,d}$	Role
Experiment 1	$H_{ce,SI} = K_{ce}/s$	1.5	–	f_t	training,
	$H_{ce,DI} = K_{ce}/s^2$	5	–		validation
Experiment 2	$H_{ce,SI} = K_{ce}/s$	5	–	f_t, f_d	validation
	$H_{ce,DI} = K_{ce}/s^2$	15	–		
Experiment 3	$H_{ce,1} = \frac{K_{ce}}{s(s+\omega_{ce})}$	90	6	f_t	test
	$H_{ce,2} = \frac{K_{ce}}{s(s+\omega_{ce})}$	30	0.2		

B. Three Datasets

Three compensatory tracking task datasets are used for the research in this article. In Experiment 1 by Zollner et al. [32] and Experiment 2 by Lu et al. [33], HC behavior was measured for different CE dynamics, using the (left/right) compensatory display of Fig. 2. In Experiment 3 by Plaetinck et al. [16], HC responses were measured, for time-varying CE dynamics, using the (up/down) compensatory display of Fig. 3.

The three experiments all used different f_t, f_d, K_{ce}, and ω_{ce} settings; see Table I and the corresponding publications for the full details [16], [32], [33]. The most distinct conditions are the pure SI $H_{ce,SI}$ and pure DI $H_{ce,DI}$ CE dynamics. These conditions are used in this article to train, validate, and test the initial LSTM network.

Data from Experiment 1 are used to *train and validate* the pattern recognition network, Experiment 2 data are used to *validate* the network performance, and Experiment 3 data are used for *testing* of the trained classifier to detecting HC adaptation with time-varying CE dynamics. All data were recorded with a 100-Hz sampling frequency (SF). The experiments are summarized as follows.

1) Experiment 1: Zollner et al. [32] studied combinations of CE dynamics and forcing function bandwidths ω_i. Ten subjects participated; each condition was tested five times (runs); each run lasted 110 s. Compared to Experiment 2, this experiment has

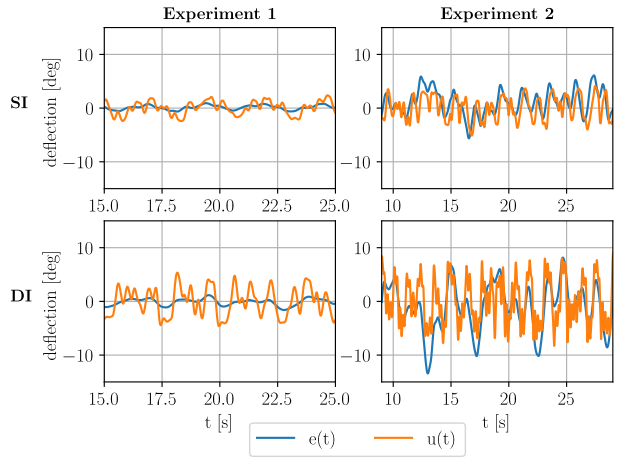


Fig. 4. Comparison of tracking data amplitude between Experiment 1 (left) and Experiment 2 (right), for SI (top) and DI (bottom) CE dynamics.

lower gain K_{ce} values and, thus, lower amplitudes of signals e and u (see Fig. 4). While Zollner et al. [32] also varied the forcing function bandwidth ω_i—as defined by McRuer and Jex [1]—over different settings, in this article, only the SI and DI data for the medium bandwidth $\omega_i = 2.5$ rad/s are used.

2) Experiment 2: Lu et al. [33] investigated combinations of CE dynamics with/without simulator motion. A disturbance signal f_d was added to dissect the visual and motion contributions to HC behavior. Six subjects participated, conditions were tested five times, and a run lasted 81.92 s. Here, only the data of the SI and DI runs *without* motion are used.

3) Experiment 3: Plaetinck et al. [16] investigated HC behavior with fixed or time-varying CE dynamics described by (1). In the time-varying runs, the CE dynamics changed halfway the run, from $H_{ce,1}$ to $H_{ce,2}$, requiring the HC to adapt. These dynamics approximated SI and DI dynamics, respectively: both transitioned from SI to DI dynamics at either a high frequency (6 rad/s) or a very low frequency (0.2 rad/s) (see Table I). Eight subjects participated, conditions were tested three times, and a run lasted 81.92 s.

C. Data Processing

Fig. 4 illustrates differences in the amplitudes for both the SI and DI conditions, as well as for Experiments 1 and 2. Clearly, to ensure classifier robustness across different experiments, which possibly use different settings for the CE gain or forcing function power, the time-series data need to be scaled to prevent the classifier from simply "learning" signal magnitude characteristics. Hence, HC time-series data must be carefully prepared before it is provided to the LSTM network.

1) Partitioning: A single run of Experiment 1 has 11 000 samples. Using such a long sequence as input to a neural network can degrade the performance of some neural network models significantly [36]. It is foreseen that it is required to segment data in a fixed observed WS and/or to reduce the SF to obtain adequate classification performance. Moreover, using a smaller WS (e.g., 1 s rather than 30 s) would allow for *real-time* online classification applications. Overlap is a way to improve time sequence classification performance [37]. It increases the amount of data available for training: 50% or 90% overlap would

increase the dataset with 200% or 1000%, respectively. The tradeoffs between WS, SF, overlap, and classification accuracy will need to be investigated.

2) Input Feature Usage and Signal Time Derivatives: Basically, any signal in the tracking task (f_t, e, u, x, etc.) can be used as input to the classifier. Signals can be used one by one, altogether, or in any combination. To capture solely the adaptation of HC behavior to different CE dynamics, and not the change in CE dynamics itself, only the HC *input* (the error e) and HC *output* (the control signal u) are used as inputs for the classifier. Early attempts using other control-loop signals, like f_t or x, led to the classifier "picking up" on trivial patterns—e.g., the known CE dynamics that relate u to x—and thus no longer providing a useful classification of HC behavior.

To increase potential information for the LSTM, first-order time derivatives are computed for e and u using a second-order central difference approximation in the interior points and first-order one-sided forward or backward difference approximations at the signal boundaries. This yields two additional signals, $\dot{e}$ and $\dot{u}$. Second-order derivatives $\ddot{e}$ and $\ddot{u}$ are not used; the first-order derivatives yielded satisfactory performance.

3) Scaling: Applying scaling on the data is essential for correct classification across different experiments and can also increase convergence [38]. Three common scaling methods are considered and here applied on the HC output u for illustration: 1) *normalizing*, where a maximum absolute scaler is used to ensure positive and negative sign convention by scaling the entire sequence between -1 and 1: $\hat{u}(t) = u(t)/\max(|\min(\overrightarrow{u})|, |\max(\overrightarrow{u})|)$; 2) *standardizing* ($Z$-score), which removes the mean and scales to unit variance: $\hat{u}(t) = (u(t) - \mu)/\sigma$, with μ the mean and σ the standard deviation of the entire time sequence u; and 3) *robust scaling*, which removes the median and scales the time sequence to its interquantile ranges (first quantile and third quantile) [39].

Scalers can be applied to the entire tracking run, for offline analysis, or to windows (splitted intervals), for real-time classification. When the observed WS is very small (e.g., 1 s), there may be insufficient data to perform standardizing and robust scaling. We test four options: 1) window-wise normalized (suitable for real-time use); 2) entire run normalized; 3) entire run standardized; and 4) entire run robustly scaled. All scalers were implemented using *Scikit-learn 0.21.1* [39].

The signals of Experiments 1 and 2 are originally expressed in degrees and radians, respectively, resulting in large amplitude differences. Fitting scaling parameters solely on the first experiment and using these parameters to transform the data of the second experiment results in large discrepancies between the datasets. For the network to be able to generalize on data from other experiments, each individual tracking run for each individual participant has been scaled separately. Subsequently, the tracking runs are split into separate windows and assigned to the train, validation, and test datasets. For window-wise normalized, the individual tracking runs are first split into separate windows, then scaled for each separate window, and finally assigned to train, validation, and test datasets.

D. Network Architecture

An LSTM neural network architecture is used to classify HC behavior from time-domain data. LSTMs are a particular form of recurrent neural networks [40]. Long-range dependencies between data sequences (such as delayed effects) cannot be detected in standard recurrent neural networks due to the vanishing gradient problem [41], which is solved in LSTM networks. The LSTM network and its hyperparameters are summarized in the Appendix.

LSTMs are designed for sequence prediction problems and can use time signals as inputs. Here, it is attempted to classify HC behavior by training an LSTM neural network in a supervised fashion, where all training data are labeled. The input data exist in observed windows of (combinations of) measured signals (e, $\dot{e}$, u, and $\dot{u}$) from each individual subject and tracking run. Labels are assigned to the observed windows according to its corresponding class; here, HC behavior is representative for control of either SI or DI CE dynamics.

III. Approach

Data provided to machine learning models are referred to as *features*, and providing appropriate features is crucial in achieving high performance [42]. Section II-C discussed a variety of settings in the dataset configuration, such as the observed WS, the SF, the scaling technique, the signals used, the usage of overlap, and the required size of the training dataset. The four-step approach adopted here to establish a well-performing dataset configuration is illustrated in Fig. 5.

A. Step 1: Window Sizes and Sampling Frequencies

Changing the WS affects the temporal information provided to the network, while the SF determines the number of samples in the window. An optimized WS and SF can be crucial for the classifier and also affect its potential for online application. In this step, effects on performance are investigated for a range of WS (0.2–50 s) and SF (100, 50, and 25 Hz) settings. In Step 1, only the signals e and u are used as input features.

B. Step 2: Scalers and Input Feature Combinations

The four scaling methods of Section II-C3 are tested in this step. Time derivatives $\dot{e}$ and $\dot{u}$ are computed to test their ability to add information. To demonstrate the importance of the input features used, all the possible (univariate and multivariate) combinations are tested. To ensure robustness across experiments, 100% of the dataset from Experiment 2 is used as the validation dataset. The dataset of Experiment 2 is equally transformed as Experiment 1, depending on the considered dataset configuration.

C. Step 3: Overlap and Required Training Dataset Size

To test the amount of HC data required for accurate classifier training, reducing data and using overlap are tested. Overlap is expected to increase network performance and stability, especially when only a small dataset is available for training [43]. The minimum amount of data required is tested by reducing the number of subjects, runs, and tracking run length included in the training data. Subjects are randomly eliminated during each repetition. Then, the number of runs is reduced by removing the last run per subject. Finally, the length per tracking run is reduced either "sequentially," "reversed sequentially," or "randomly" (see Fig. 6).

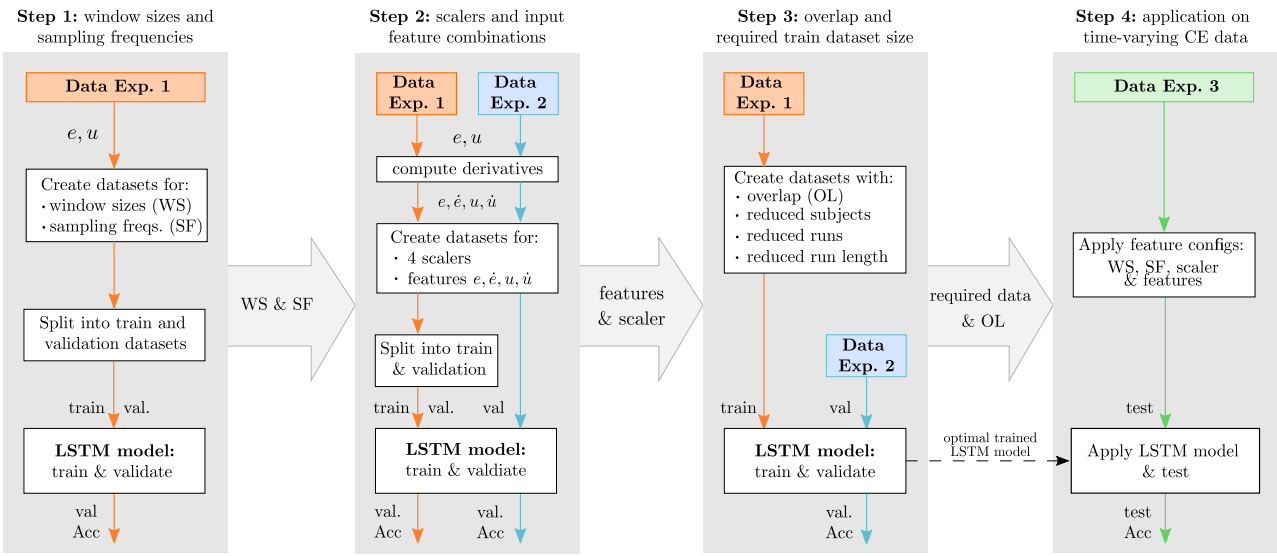


Fig. 5. Flowchart of the four-step approach to determine an adequately performing dataset configuration.

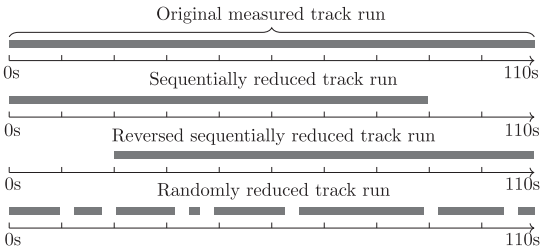


Fig. 6. Three methods used to reduce the tracking run length.

Experiment 2 is used as the validation dataset, without overlap, and is according to the outcome of Step 2. The validation dataset is identical during this step for all tested variations, which allows for a direct comparison of performance between the networks.

D. Step 4: Application on Time-Varying CE Data

The optimized LSTM network obtained in Steps 1–3 is tested on the data of Experiment 3 to apply it for classification on data *not used* for training and validation, and where the CE dynamics *may change* over time. Also, the Experiment 3 CE dynamics do not exactly match the pure SI and pure DI used for model training (see Table I). Although the network is expected to have an overall poorer classification performance, it is still expected to detect the HC behavior *adaptations*.

E. Accuracy and Model Selection

Since the number of samples of SI and DI CE dynamics is distributed 50/50, the performance results of Steps 1–3 are measured and compared using the accuracy metric

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \times 100\% \qquad (3)$$

where the "Number of correct predictions" is calculated as the sum of all "true positive" SI and DI classifications. It should be noted that for the binary classification problem considered here, this is identical to the classical definition of accuracy, which sums all "true positive" and "true negative" results for the numerator term.

In Steps 1–3, the LSTM network is trained for a total of 30 repetitions, due to the stochastic nature of the training process (see the Appendix). Each repetition has 20 training epochs. In Step 3, the complete Experiment 1 dataset is used for training; in Steps 1 and 2, the Experiment 1 dataset is uniquely shuffled and split in a 80/20 ratio training/validation dataset for *each* of the 30 repetitions. LSTM networks are prone to overfitting, resulting in varying classification accuracy after each training epoch. The epoch with the highest mean accuracy on the validation set is selected for *each* repetition as the final model.

IV. Results

A. Step 1: Window Sizes and Sampling Frequencies

Following the approach of Fig. 5, we first optimize the WS and SF settings using the signal combination $e + u$ and nonscaled data from Experiment 1 to train the LSTM network.

Fig. 7 shows the accuracy on the *validation* dataset from Experiment 1 for the full range of considered WS values, i.e., 0.2–50 s. All WSs are tested with three SF settings: 100, 50, and 25 Hz. A very small WS (<0.8 s) results in an up to 10% decrease in accuracy. A larger WS (>2 s) results in reduced consistency between the different training repetitions (increased spread). The increasing accuracy for (very) large WS might be caused by the neural network learning particular characteristics of Experiment 1, such as the (shape of the) f_t signal.

Lower accuracy and less consistent results are found for SF = 100 Hz, suggesting that higher SFs increase the amount of irrelevant information (i.e., HC remnant). Zooming-in on the right inset of Fig. 7 (WS 1.2–2 s), it can be seen that for four different combinations of SF and WS, the median classification accuracy on the *validation* dataset is equal to 100%: SF = 50 Hz and WS = 1.6 s, as well as SF = 25 Hz with a WS of 1.6, 1.8, or 2 s. For the further analysis in Step 2, the combination

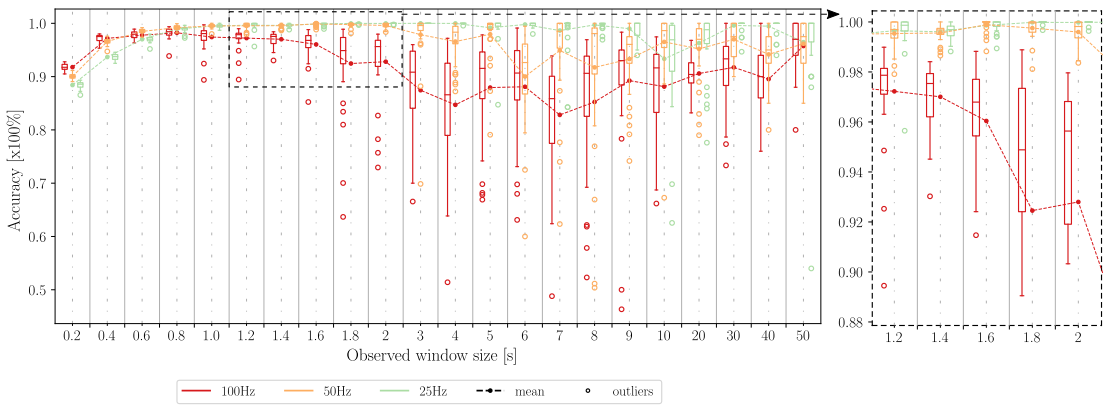


Fig. 7. Validation accuracy on Experiment 1 for varying observed WS (columns) and three SFs (colors). *Settings:* training data = 80% of Experiment 1 and validation data = 20% of Experiment 1.

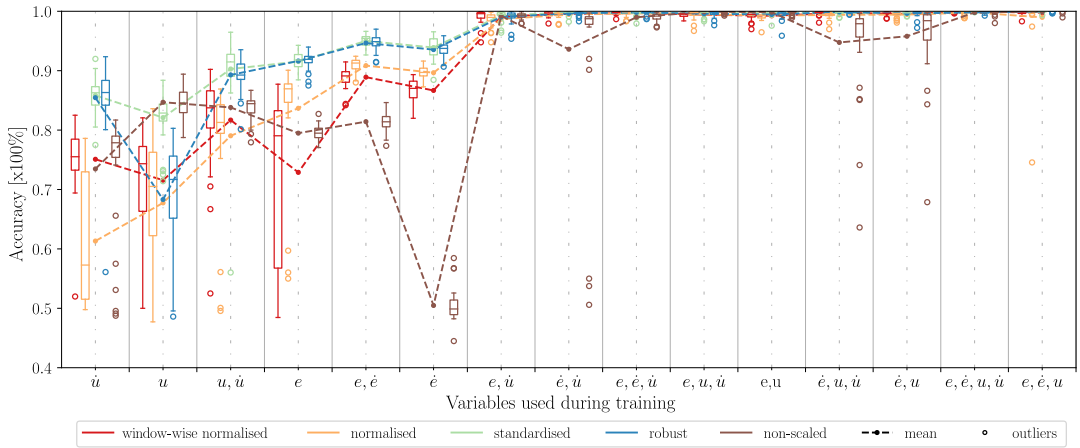


Fig. 8. Validation accuracy on Experiment 1 for varying combinations of input variables e, $\dot{e}$, u, and $\dot{u}$ (columns) and four scalers (colors). *Settings:* training data = 80% of Experiment 1, validation data = 20% of Experiment 1, SF = 50 Hz, and WS = 1.6 s.

of the smallest WS and highest SF (highest resolution in each window)—i.e., SF = 50 Hz, and WS = 1.6 s—was selected.

It should be emphasized that this high performance applies to the *validation* dataset and only describes the generalization capability of the LSTM network to recognize HC behavior on unseen examples that match the training dataset (Experiment 1).

B. Step 2: Scalers and Input Feature Combinations

Scaling techniques (see Section II-C3) and the (univariate and multivariate) HC input–output signal combinations used as LSTM input features (i.e., e, $\dot{e}$, u, and $\dot{u}$) are tested in Step 2. Data of both Experiments 1 and 2 are used to validate the LSTM performance.

Fig. 8 shows the classification accuracy on the Experiment 1 *validation* dataset for varying input feature combinations and the four scaling methods as well as the nonscaled dataset. Poor generalization and low consistency is obtained for univariate use of e, $\dot{e}$, u, or $\dot{u}$ and multivariate use of "same signal" combinations, i.e., $e + \dot{e}$ and $u + \dot{u}$. The inability to generalize on univariate use of $\dot{e}$ is likely caused by its low amplitude $(\pm 10^{-2})$ and the minor difference in $\dot{e}$ between SI and DI. Scaling removes this issue and increases accuracy from 50% to 90%. Generalization approximates 100% on the Experiment 1 *validation* dataset when multivariate combinations of $e + \dot{e}$ and $u + \dot{u}$ are used, revealing their importance for HC behavior classification.

Fig. 9 shows the classification accuracy on the *validation* dataset from Experiment 2 for the same combinations of input features and scaling variations. Training the network on nonscaled data results in poor generalization on this validation dataset, as expected. In fact, when no scaling is used, all data are consistently classified as SI, indicating undesired abstraction on lower level features, such as signal amplitudes. Again, poor generalization ranging from 55% to 80% accuracy and increased inconsistency are observed for univariate use of e, $\dot{e}$, u, or $\dot{u}$ and multivariate use of $e + \dot{e}$ and $u + \dot{u}$. Accuracy increases up to 96% using multivariate combinations of e and $\dot{e}$ together with u and $\dot{u}$. Note that only a small drop to 95% accuracy is obtained when window-wise normalized scaling is used, the preferred method for online applications.

Solely using the error signal (e and $\dot{e}$) yields better generalization than using only the HC output (u and $\dot{u}$). Apparently, e contains crucial information for classifying HC behavior when

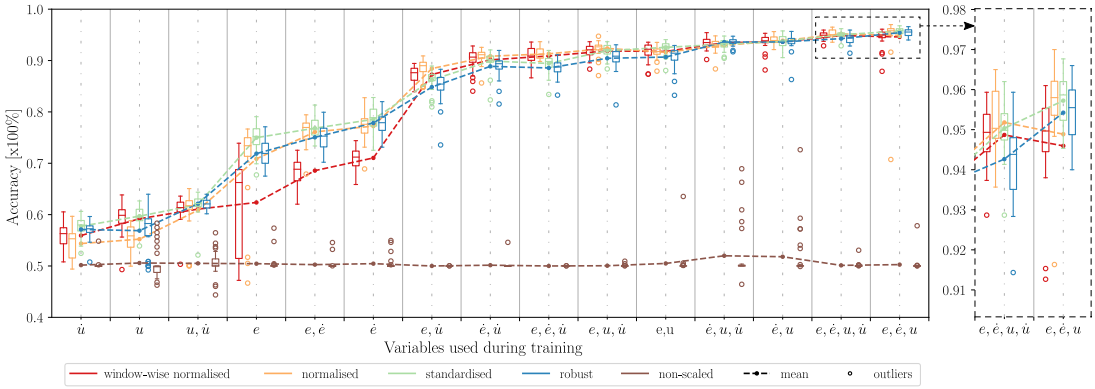


Fig. 9. Validation accuracy on Experiment 2 for varying combinations of variables e, $\dot{e}$, u, and $\dot{u}$ and four scalers. *Settings:* training data = 80% of Experiment 1, validation data = 100% of Experiment 2, SF = 50 Hz, and WS = 1.6 s.

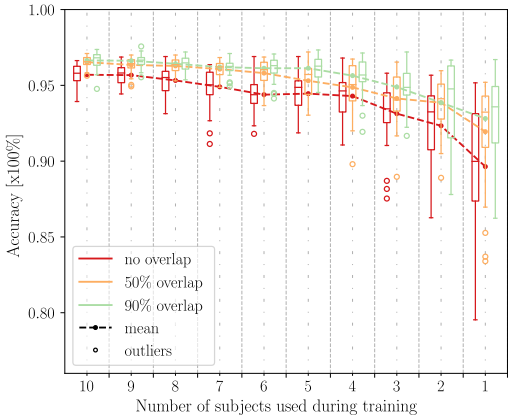


Fig. 10. Validation accuracy on Experiment 2 for reducing number of subjects with and without overlap on the training dataset *Settings:* training data = variable percentage of Experiment 1, validation data = 100% of Experiment 2, SF = 50 Hz, WS = 1.6 s, scaler = standardized, and input variables = $e + \dot{e} + u$.

controlling SI or DI CE dynamics. Using these signals together with their time derivatives increases accuracy with around 3%, a modest yet consistent benefit. The best average performance of 96% classification accuracy is obtained with standardization of the entire time series and the combination $e + \dot{e} + u$ (see the inset in Fig. 9). This configuration is used in Step 3.

C. Step 3: Overlap and Required Training Dataset Size

Three training dataset reducing methods are tested to determine the minimum amount of tracking data required. First, subjects in the Experiment 1 dataset are randomly dropped from the training data until only one subject remains. This is done for both nonoverlapping and overlapping observed windows. The randomly dropped subjects during each repetition are stored and reused in reducing the tracking runs and tracking length.

1) Reducing Subjects and Testing Overlap: Fig. 10 shows the effect of reducing the number of subjects and using different overlap settings. Reducing the training data to five subjects results in a minor decrease of 0.5% in accuracy. Further reduction leads to reduced consistency between repetitions and decreases accuracy by 3%.

Overlap increases accuracy when using one subject by 4%, when using five subjects by 2%, and when using ten subjects by 1%. It also increases consistency by providing more examples. The slight increase of 1% in accuracy between 50% overlap and 90% overlap is a poor benefit, however, compared to the fivefold increase in training time because of using five times more training data.

Using a single subject results in a relatively high accuracy of 90%. This high accuracy for using only a single subject can be explained by the facts that the subjects are highly trained (the training data are not included in the datasets) and that the SI and DI tracking tasks are markedly different. Overall, subjects must respond in a similar fashion to either of these tracking tasks, which may cause the strong prediction capabilities on other subjects.

Based on Fig. 10, data from five subjects with 90% overlap are selected as optimized settings.

2) Reducing Tracking Runs: Reducing the number of tracking runs used for each subject from five to three has no effect on classification accuracy (not shown); reducing it further decreases accuracy only by around 0.5%. A more pronounced decrease in accuracy only occurs when further reducing the number of subjects. As a result, three tracking runs per subject are used in the training dataset.

3) Reducing Tracking Length: The run length is reduced using the three techniques of Fig. 6. Effects on the validation accuracy are shown in Fig. 11, where all techniques are seen to lead to a decrease in accuracy. Sequentially reducing the run length shows the largest accuracy decrease of around 5%; it also leads to higher inconsistency. Reversed sequentially reducing run length leads to less degradation in performance. This indicates that data from the last 40% of the tracking run contain more valuable information for generalization to Experiment 2.

This difference may be caused by excluding a, apparently valuable, part of the forcing function f_t. Indeed, when inspecting f_t a spike was observed at $t = 107$ s (not shown). This spike is only included when going from 80 to 90 tracking percentage used in sequentially reduced, and going from 10 to 20 tracking percentage data used in reversed sequentially reduced. This may have caused the increase of classification accuracy at these percentages (see Fig. 11).

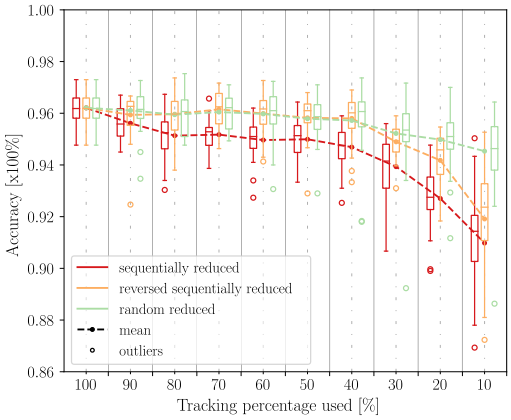


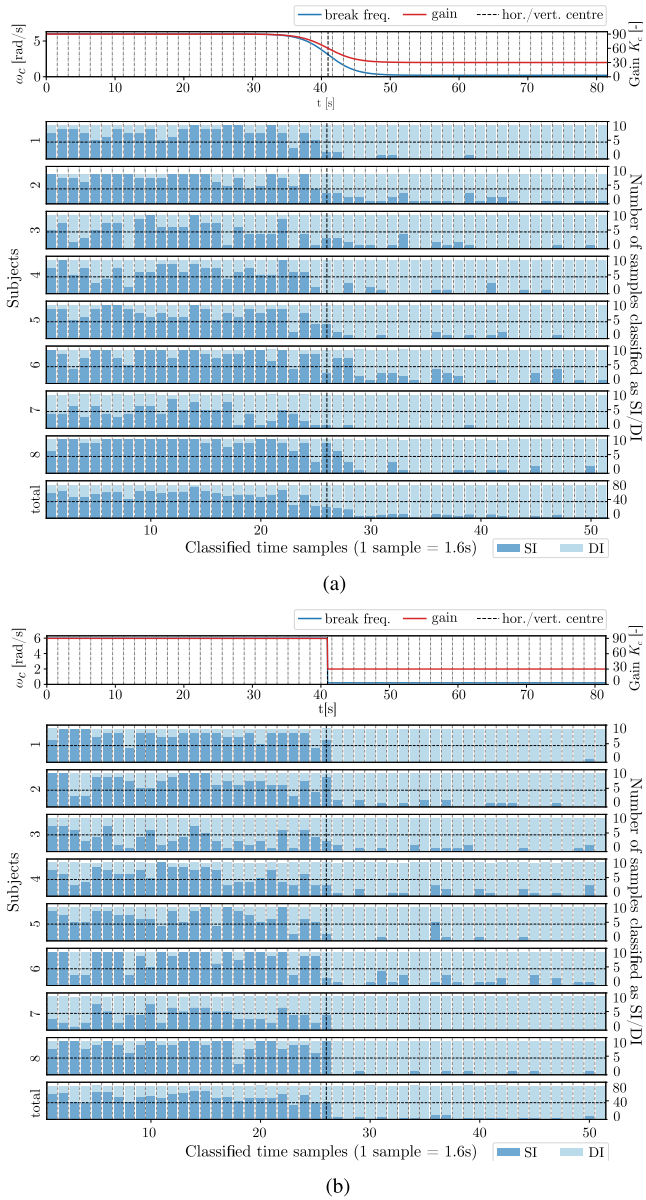
Fig. 11. Validation accuracy on Experiment 2 for reducing the length of the tracking runs *Settings:* training data = variable percentage of Experiment 1 (overlap = 90%, subjects = 5, runs = 3), validation data = 100% of Experiment 2, SF = 50 Hz, WS = 1.6 s, scaler = standardized, and input variables = $e + \dot{e} + u$.

Randomly reducing the run length decreases classification accuracy less. This can be explained by the fact that with random data reduction, the network receives a more diverse set of samples for training, as compared to the sequentially reduced methods. The model benefits the most from the entire tracking length, though, which is selected for the final step.

D. Step 4: Testing on Time-Varying CE Data

With a well-performing configuration defined in Steps 1–3, in Step 4, the performance of the trained and optimized LSTM network in classifying SI or DI HC behavior is tested on data from Experiment 3. Recall that the simulated CE dynamics $H_{ce,1}$ (SI-like) and $H_{ce,2}$ (DI-like) were *not* pure SI and DI (see Table I). In the time-varying runs, the CE dynamics transition from $H_{c,1}$ to $H_{c,2}$ halfway the run, either gradually (within 30 s) or instantly [16], as illustrated in the top rows of Fig. 12(a) and (b), respectively. In these two figures, the rectangles in the following eight rows show the summation of the number of samples classified as SI (dark blue) or DI (light blue), as a function of the time sample (horizontal axis) for all eight subjects. Recall that each subject performed nine tracking runs, and a perfect classification at the beginning of the first half of the run would mean nine dark blue SI classifications. A perfect classification at the end of the second half of the run would mean nine light blue DI classifications. In the last row called "total," the SI and DI classifications are added for all subjects, again as a function of time sample, where a perfect classifier at the first half of the run would score 72 SI classifications. In the following, classifier performance is expressed as the percentage of "correct" SI and DI classifications. Note that because the CE dynamics were neither SI nor DI in Experiment 3, this definition of correct/incorrect is not completely fair.

For the time-invariant CE data of [16], the percentages of correct classification were 76.3% ($H_{ce,1}$, SI-like) and 95.7% ($H_{ce,2}$, DI-like). These results can be explained by the fact that *all* CE dynamics in Experiment 3 had a nonzero lag component (see (1) and Table I), which could lead subjects to *always* exert a small lead equalization. This can explain the bias toward



(a)

(b)

Fig. 12. Classifications for gradual (top) and instant (bottom) changing CE dynamics, as a function of time (all subjects, Experiment 3). (a) Gradual change. (b) Instant change.

classifying HC behavior as "DI," as no *pure* SI dynamics were simulated for which the classifier was trained and validated. Nonetheless, its performance when dealing with new experiment data which it has *never* encountered before, for classifying HC behavior for CE dynamics which are *not* the same as those for which it was trained, is promising.

Fig. 12 shows classification performance as a function of time, using the WS of 1.6 s, for the gradual [see Fig. 12(a)] and instant [see Fig. 12(b)] change in CE dynamics. In each subfigure, the top row shows the varying CE dynamics parameters of Table I; the bottom row shows the total number of SI (dark blue) and DI (light blue) classifications; intermediate rows show the SI/DI classifications for all eight subjects. Classifications vary during the entire tracking run, due the different CE dynamics, the time-varying CE, and time variations in HC behavior. Differences

between subjects exist: the classifier is successful for Subjects 1 and 8, but varies for Subjects 2, 3, and 6. As explained, the classifier leans toward DI classification: DI behavior is classified mostly correctly in the second part of the run, but often incorrectly classifies HC behavior as "DI" behavior in the first part.

The reader should note the extremely rapid adaptation of the HC in the instantly changing CE dynamics runs [see Fig. 12(b)] and the classifier's ability to *immediately* capture that change. This shows the great potential of these classifiers, detecting rapid changes in HC behavior using short WSs of only 1.6 s. This is much faster than the approximately 10-s detection delay obtained on the same data with a "classical" time-varying HC modeling method [16].

V. Discussion

The goal of this study was to test the capability of an LSTM recurrent neural network to classify HC behavior when controlling either SI or DI CE dynamics. Different dataset configurations were used to train an LSTM network on data of Experiment 1, e.g., varying the WS, the SF, the scaling technique, the signals selected as input features, overlap, and the size of the training dataset (see Fig. 5). The performance of the classifier was validated on data of Experiment 2. The resulting optimized LSTM network was successfully tested for classifying the adapting HC behavior when controlling time-varying CE dynamics (Experiment 3 data).

The stepwise dataset configuration technique (see Fig. 5) arguably results in a suboptimal outcome. Quoting Snoek et al. [44]: "*...tuning is often a "black art" requiring expert experience, rules of thumb, or sometimes bruteforce search*," indicating the difficulty of obtaining an overall optimal configuration when using a black box modeling technique. A formal optimization technique was not considered for finding the fully optimized dataset configuration and LSTM hyperparameter settings, to reduce the risk of testing irrelevant dataset configurations and wasting computational resources. Instead, a heuristic approach was used, tailored by a priori knowledge in cybernetics, which yielded excellent performance of 100% classification accuracy on the Experiment 1 validation dataset and 96% classification accuracy on the Experiment 2 validation dataset.

To further improve on this result, a (computationally expensive) complete single-step analysis of all hyperparameters' main effects, and their potential interactions, on classifier performance is important future work. Apart from model structure improvements, the LSTM network hyperparameters could, for instance, be optimized using a Bayesian optimization method to increase performance [44]. The role of the forcing functions, their power and shape, needs to be further investigated as local parts of these signals may impact the training of the classifier.

The lower accuracy for large WSs agrees with the work of Levison et al. [36], who report that long-term dependencies become inaccessible to the LSTM network for sequences of more than 1000 points. This may be overcome by using a different network structure, such as proposed in [45], although here shorter sequences are preferred to allow for online applications. The increased performance by *reducing* the SF is in line with the findings in [46]. Reducing the SF from 100 to 50 Hz apparently increases the capability to abstract details of higher importance. This finding may be tracking-task dependent; however, as when the task contains more fast or slow control responses, for instance, as the result of using target signals f_t with different bandwidths ω_i, it might be essential to increase or reduce the SF, respectively.

Scaling was found to have a major impact on classifier performance across multiple experiments, increasing accuracy from 50% up to 96%. Without scaling, the LSTM network consistently classifies HC behavior as controlling an SI CE, indicating abstraction on low-level features such as signal amplitude. The amplitudes of tracking task signals can be affected by many factors, such as the used units, the manipulator hardware/software, and the CE's properties. Fig. 9 shows that when not accounted for in data preprocessing, generalization toward different datasets and signal amplitudes is poor. Extreme care should be taken how experimental data are used in this application of time-series classification, requiring insight into how classifiers actually work, but also knowledge on experimental equipment and human behavior. Although the use of scalers benefited this study, it may also reduce performance, for instance, when the main difference between classes is a change in HC gain [K_p in (2)].

Given that the best accuracy was achieved with standardized data, one could argue that the LSTM network requires scaling performed on the entire tracking run, forcing the implementation of the model to offline usage only. Several techniques exist, however, to still enable online application. An initializing phase can be used to compute the standardization parameters, such as the standard deviation σ. The sample-wise normalization method decreased classification accuracy only by 1% (see Fig. 9) and can be implemented online.

The time-varying CE dynamics application of our classifier (Step 4) showed a larger bias of $\approx 23\%$ toward DI classifications during SI-like CE control, compared to only $\approx 5\%$ SI classifications for the DI-like CE. These biases are not unexpected, as even when the SI dynamics have a high break frequency ω_{ce}, the HC may still compensate for this lag by applying lead [1], [32]. The LSTM network is trained for classifying HC behavior with pure SI and DI dynamics, and apparently small amounts of lead equalization are detected by the LSTM. Between-subject differences in classification accuracy in Fig. 12 may originate from the fact that some subjects performed better than others by, for example, applying larger lead time constants [T_L in (2)]. For the "one-size-fits-all" classifier trained in this article, the expected accuracy depends on how well a certain HC's data match the (majority of) participants in the training data. Between-subject differences in classification accuracy may be amended by ensuring uniformly distributed training data or through personalizing the classifier to recognize individual participants.

In the application to the time-varying CE data in Step 4, the trained LSTM network shows increased variability in classifying SI and DI CE dynamics. One approach to amend this would be to increase the number of CE dynamics that the LSTM network can distinguish. The Experiment 1 dataset [32] contains three more tested CE dynamics (break frequencies ω_{ce} of 3, 2, and 1 rad/s). Including samples of these conditions, with subtler differences between dynamics compared to the clear-cut SI versus DI dynamics as used here, may extend the applicability of the classifier and is a valuable next step. When reliable separation of

different classes of HC behavior proves difficult, convolutional layers can be stacked on top of the LSTM layers, as used in [31] and [47]. These layers extract the most significant features of a time series by applying multiple local filters and generating invariant local features.

Although the detection performance in Step 4 was lower, the accuracy is still impressive, given that the classifier dealt with data it never encountered before, obtained in an experiment performed with different subjects, in a different simulator, with different control axis (pitch versus roll), control manipulators, CE dynamics and forcing function settings. The proposed method shows promising capabilities for online (and time-varying) HC behavior detection. It outperforms state-of-the-art techniques, such as recursive ARX-models [16], enabling HC adaptation detection well within 10 s achieved in [16]. With our current settings, the LSTM method can classify control behavior every 1.6 s, yielding a six times faster detection. With overlapping windows, the detection delay can be further reduced, i.e., by shifting the data window with steps of 0.1 s, yielding *near-continuous* detection capabilities.

Other promising time-series classification methods can be considered for HC behavior classification. These methods rely more on statistical feature engineering (see, e.g., [24] and [25]). Future research will need to assess their potential and compare it to the LSTM approach adopted here. Furthermore, more insights could be obtained by visualizing the internal decision structure of the LSTM network, as performed in [48].

This study focused on compensatory tracking, in a first attempt to test the capability of a black box pattern recognizer to classify human control behavior. Results are promising, but the feasibility of the approach should be investigated in other more relevant tasks, such as control with pursuit and preview displays [5], [11], in biodynamic studies [49], and in multichannel tasks [9] such as vehicular control. These all include more signals presented to the HC, visual preview, and physical motion, which require renewed considerations on the use of scalers and input features. However, above all, it means the method will be tested in situations where we do *not* know what the human input(s) are exactly, what loops are closed, scratching the surface of the great unknown: how is the observable human control output synthesized? Answering this question will facilitate the design of the human control behavior monitoring component required for safe, truly cooperative, shared human–automation teaming systems.

VI. Conclusion

An LSTM neural network is tuned and applied to classify (adaptation of) human control behavior in compensatory tracking tasks. To optimize the network for HC time-series classification, key input feature data configurations and settings are optimized. Effects of varying WS, input feature combinations, scaling, and data reduction techniques are presented. A classification accuracy of 96%, determining HC behavior every 1.6 s, is achieved for validation on HC data from a *different* experiment than used for classifier training. For an accurate and generalizable result across our three experimental validation and test datasets, input feature scaling and use of both the error e and HC output u signals as features were found to be essential to achieve reliable classification accuracy. Reducing the number of subjects, tracking runs, and tracking run length included in the training dataset affected accuracy only slightly. The LSTM time-series classification approach shows promise for *online* detection of HC adaptations, in HC monitoring and support applications.

Appendix

LSTM structure: The network has two stacked LSTM layers [50], each containing 100 LSTM cells, to create a deep recurrent neural network [29]. These settings are a good rule of thumb [51]. This structure results in 122 202 trainable parameters for the LSTM network. A dropout layer is applied after each LSTM layer to decrease overfitting and increase performance [52]. The dropout value is set at 0.2.

Implementation: Networks are set up in Python (version 3.6.8) using *Keras* (version 2.2.4) [53], and *Tensorflow* (version 1.12.0) [54]. Standard values of the tunable hyperparameters are used; these are fixed during the entire study.

Training: Neural networks learn the most from unexpected samples [38]. Training samples are shuffled after each epoch, i.e., each time the model has seen the entire dataset. Datasets are shuffled prior to splitting into a training and validation dataset, to prevent the network of learning: a) control patterns of individual subjects or b) time-varying HC behavior (even with fixed CE dynamics). This study focused on comparing effects of dataset configuration on classification performance rather than optimizing a single LSTM network, so a mini-batch size of 100 is used. This decreases computational time with a slight cost of training stability and generalization performance [55]. Keras' standard learning rate, $\eta = 0.001$ [53], is used.

Optimizer: The Adam optimizer is used, with default settings proposed in [56] and [57]: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ (used to avoid division by zero). Adam is a stochastic optimizer, with shuffling and dropout, producing varying results each training session. To objectively compare different dataset configurations, 30 repetitions of the LSTM network are created for each data configuration. This number is assumed to be sufficient to account for the stochastic outcomes.

References

[1] D. T. McRuer and H. R. Jex, "A review of quasi-linear pilot models," *IEEE Trans. Hum. Factors Electron.*, vol. HFE- 8, no. 3, pp. 231–249, Sep. 1967.
[2] L. R. Young, "On adaptive manual control," *IEEE Trans. Man Mach. Syst.*, vol. MMS-10, pp. 292–331, Dec. 1969.
[3] E. R. Boer and R. V. Kenyon, "Estimation of time-varying delay time in nonstationary linear systems: An approach to monitor human operator adaptation in manual tracking tasks," *IEEE Trans. Syst., Man, Cybern., A, Syst. Humans*, vol. 28, no. 1, pp. 89–99, Jan. 1998.
[4] R. A. Hess, "Modeling pilot control behavior with sudden changes in vehicle dynamics," *J. Aircr.*, vol. 46, no. 5, pp. 1584–1592, 2009.
[5] K. van der El, D. M. Pool, H. J. Damveld, M. M. van Paassen, and M. Mulder, "An empirical human controller model for preview tracking tasks," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2609–2621, Nov. 2016.
[6] X. Zhang, S. Wang, J. B. Hoagg, and T. M. Seigler, "The roles of feedback and feedforward as humans learn to control unknown dynamic systems," *IEEE Trans. Cybern.*, vol. 48, no. 2, pp. 543–555, Feb. 2018.
[7] M. Mulder et al., "Manual control cybernetics: State-of-the-art and current trends," *IEEE Trans. Human-Mach. Syst.*, vol. 48, no. 5, pp. 468–485, Oct. 2018.

[8] S. Xu, W. Tan, and X. Qu, "Modeling human pilot behavior for aircraft with a smart inceptor," *IEEE Trans. Human-Mach. Syst.*, vol. 49, no. 6, pp. 661–671, Dec. 2019.
[9] S. Barendswaard, D. M. Pool, M. M. van Paassen, and M. Mulder, "Dual-axis manual control: Performance degradation, axis asymmetry, crossfeed and intermittency," *IEEE Trans. Human-Mach. Syst.*, vol. 49, no. 2, pp. 113–125, Apr. 2019.
[10] S. A. S. Mousavi, X. Zhang, T. M. Seigler, and J. B. Hoagg, "Characteristics that make linear time-invariant dynamic systems difficult for humans to control," *IEEE Trans. Human-Mach. Syst.*, vol. 51, no. 2, pp. 141–151, Apr. 2021.
[11] S. A. S. Mousavi, F. Matveeva, X. Zhang, T. M. Seigler, and J. B. Hoagg, "The impact of command-following task on human-in-the-loop control behavior," *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 6447–6461, Jul. 2022.
[12] M. M. Lone and A. K. Cooke, "Review of pilot models used in aircraft flight dynamics," *Aerosp. Sci. Technol.*, vol. 34, pp. 55–74, 2014.
[13] S. Xu, W. Tan, A. Efremov, L. Sun, and X. Qu, "Review of control models for human pilot behavior," *Ann. Rev. Control*, vol. 44, pp. 274–291, 2017.
[14] P. M. T. Zaal, "Manual control adaptation to changing vehicle dynamics in roll–pitch control tasks," *J. Guid., Control Dyn.*, vol. 39, no. 5, pp. 1046–1058, 2016.
[15] P. M. Thompson, D. H. Klyde, and M. J. Brenner, "Wavelet-based time-varying human operator models," in *Proc. AIAA Atmos. Flight Mech. Conf. Exhibit*, 2001, pp. 1–12.
[16] W. Plaetinck, D. M. Pool, M. M. van Paassen, and M. Mulder, "Online identification of pilot adaptation to sudden degradations in vehicle stability," in *Proc. 2nd IFAC Conf. Cyber-Phys. Human-Syst.*, 2018, pp. 347–352.
[17] A. Popovici, P. M. T. Zaal, and D. M. Pool, "Dual extended Kalman filter for the identification of time-varying human manual control behavior," in *Proc. AIAA Aviation Forum*, 2001, pp. 1–17.
[18] D. A. Abbink et al., "A topology of shared control systems—Finding common ground in diversity," *IEEE Trans. Human-Mach. Syst.*, vol. 48, no. 5, pp. 509–525, Oct. 2018.
[19] C. M. Belcastro et al., "Aircraft loss of control problem analysis and research towards a holistic solution," *J. Guid., Control Dyn.*, vol. 40, no. 4, pp. 733–774, 2017.
[20] J. W. Senders, A. B. Kristofferson, W. H. Levison, C. W. Dietrich, and J. L. Ward, "The attentional demand of automobile driving," *Highway Res. Rec.*, vol. 195, pp. 15–33, 1967.
[21] M. Mulder, M. M. Van Paassen, and E. R. Boer, "Exploring the roles of information in the manual control of vehicular locomotion: From kinematics and dynamics to cybernetics," *PRESENCE: Tele-Oper. Virtual Environ.*, vol. 13, no. 5, pp. 535–548, 2004.
[22] A. Ignatov, "Real-time human activity recognition from accelerometer data using convolutional neural networks," *Appl. Soft. Comput.*, vol. 62, pp. 915–922, 2018.
[23] T. Ersal, H. J. A. Fuller, O. Tsimhoni, J. L. Stein, and H. K. Fathy, "Model-based analysis and classification of driver distraction under secondary tasks," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 692–701, Sep. 2010.
[24] M. Wollmer et al., "Online driver distraction detection using long short-term memory," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 574–582, Jun. 2011.
[25] F. Tango and M. Botta, "Real-time detection system of driver distraction using machine learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 894–905, Jun. 2013.
[26] R. Wang, P. V. Amadori, and Y. Demiris, "Real-time workload classification during driving using hypernetworks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3060–3065.
[27] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series FeatuRe extraction on basis of scalable hypothesis tests (tsfresh—A Python package)," *Neurocomputing*, vol. 307, pp. 72–77, Sep. 2018.
[28] A. Jain, A. Singh, H. S. Koppula, S. Soh, and A. Saxena, "Recurrent neural networks for driver activity anticipation via sensory-fusion architecture," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 3118–3125.
[29] K. Saleh, M. Hossny, and S. Nahavandi, "Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks," in *Proc. Conf. Int. Transp. Syst.*, 2017, pp. 1–6.
[30] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel, "Learning to diagnose with LSTM recurrent neural networks," in *Proc. Int. Conf. Learn. Representations*, 2016.
[31] B. Ballinger et al., "DeepHeart: Semi-supervised sequence learning for cardiovascular risk prediction," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2079–2086, doi: 10.1609/aaai.v32i1.11891.
[32] H. G. H. Zollner, D. M. Pool, H. J. Damveld, M. M. van Paassen, and M. Mulder, "The effects of controlled element break frequency on pilot dynamics during compensatory target-following," in *Proc. AIAA Guid, Navig., Control Conf.*, 2010, pp. 1–12.
[33] T. Lu, D. M. Pool, M. M. van Paassen, and M. Mulder, "Use of simulator motion feedback for different classes of vehicle dynamics in manual control tasks," in *Proc. 5th CEAS Air Space Conf.*, 2015, pp. 1–13.
[34] W. H. Levison, S. Baron, and D. L. Kleinman, "A model for human controller remnant," *IEEE Trans. Man-Mach. Syst.*, vol. MMS-10, no. 4, pp. 101–108, Dec. 1969.
[35] K. Van der El, D. M. Pool, and M. Mulder, "Analysis of human remnant in pursuit and preview tracking tasks," *IFAC-PapersOnLine*, vol. 52, no. 19, pp. 145–150, 2019.
[36] I. Sutskever, "Training recurrent neural networks," Ph.D. dissertation, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2013.
[37] O. Baños, J. M. Gálvez, M. Damas, H. Pomares, and I. Rojas, "Window size impact in human activity recognition," *Sensors*, vol. 14, no. 4, pp. 6474–6499, 2014.
[38] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade*, 2nd ed. G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Germany: Springer, 2012, pp. 9–48.
[39] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[41] S. Hochreiter et al., "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," in *Field Guide to Dynamical Recurrent Networks*. Hoboken, NJ, USA: Wiley/IEEE Press, 2001, ch. 14, pp. 237–243.
[42] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, Oct. 2012.
[43] S. J. Preece, J. Y. Goulermas, L. P. J. Kenney, and D. Howard, "A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data," *IEEE Trans. Biomed. Eng.*, vol. 56, no. 3, pp. 871–879, Mar. 2009.
[44] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 2951–2959.
[45] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently recurrent neural network (IndRNN): Building a longer and deeper RNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, 2018, pp. 5457–5466, doi: 10.1109/CVPR.2018.00572.
[46] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased LSTM: Accelerating recurrent network training for long or event-based sequences," in *Proc. Adv. Neural Inf. Process. Syst.*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds. 2016, vol. 29. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/5bce843dd76db8c939d5323dd3e54ec9-Paper.pdf.
[47] R. Zhao, R. Yan, J. Wang, and K. Mao, "Learning to monitor machine health with convolutional bi-directional LSTM networks," *Sensors*, vol. 17, no. 2, Jan. 2017, Art. no. 273.
[48] J. van der Westhuizen and J. Lasenby, "Techniques for visualizing LSTMs applied to electrocardiograms," in *Proc. Int. Conf. Mach. Learn. Workshop Hum. Interpretability Mach. Learn. (WHI 2018)*, Stockholm, Sweden, 2018, pp. 99–106.
[49] J. Venrooij et al., "A framework for biodynamic feedthrough analysis, Part I: Theoretical foundations," *IEEE Trans. Cybern.*, vol. 44, no. 9, pp. 1686–1698, Sep. 2014.
[50] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. 27th Conf. Neural Inf. Proc. Syst.*, 2014, pp. 3104–3112.
[51] N. Reimers and I. Gurevych, "Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks," 2017, *arXiv:1707.06799*.
[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.
[53] F. Chollet et al., "Keras: Deep learning for humans," 2015. [Online]. Available: https://github.com/fchollet/keras
[54] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: tensorflow.org
[55] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," 2018, *arXiv:1804.07612*.
[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, *arXiv:1412.6980*.
[57] S. Ruder, "An overview of gradient descent optimization algorithms," 2017, *arXiv:1609.04747*.