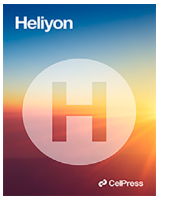
Heliyon 10 (2024) e27108

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon



Research article

Optimizing echo state networks for continuous gesture recognition in mobile devices: A comparative study

Alok Yadav [a], Kitsuchart Pasupa [a,*], Chu Kiong Loo [b], Xiaofeng Liu [c]

[a] School of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok, 10520, Thailand
[b] Faculty of Computer Science & Information Technology, University of Malaya, Kuala Lumpur, 50603, Malaysia
[c] College of IoT Engineering, Hohai University, Changzhou, 213022, China

ARTICLE INFO

Keywords:
Echo state networks
Continuous gesture recognition
Behaviour space analysis

ABSTRACT

Continuous gesture recognition can be used to enhance human-computer interaction. This can be accomplished by capturing human movement with the use of the Inertial Measurement Units in smartphones and using machine learning algorithms to predict the intended gestures. Echo State Networks (ESNs) consist of a fixed internal reservoir that is able to generate rich and diverse nonlinear dynamics in response to input signals that capture temporal dependencies within the signal. This makes ESNs well-suited for time series prediction tasks, such as continuous gesture recognition. However, their application has not been rigorously explored, with regard to gesture recognition. In this study, we sought to enhance the efficacy of ESN models in continuous gesture recognition by exploring diverse model structures, fine-tuning hyperparameters, and experimenting with various training approaches. We used three different training schemes that used the Leave-one-out Cross-validation (LOOCV) protocol to investigate the performance in real-world scenarios with different levels of data availability: Leaving out data from one user to use for testing (F_1-score: 0.89), leaving out a fraction of data from all users to use in testing (F_1-score: 0.96), and training and testing using LOOCV on a single user (F_1-score: 0.99). The obtained results outperformed the Long Short-Term Memory (LSTM) performance from past research (F_1-score: 0.87) while maintaining a low training time of approximately 13 seconds compared to 63 seconds for the LSTM model. Additionally, we further explored the performance of the ESN models through behaviour space analysis using memory capacity, Kernel Rank, and Generalization Rank. Our results demonstrate that ESNs can be optimized to achieve high performance on gesture recognition in mobile devices on multiple levels of data availability. These findings highlight the practical ability of ESNs to enhance human-computer interaction.

1. Introduction

Continuous gesture recognition is a task of identifying and classifying gestures from a continuous stream of data. Gesture recognition offers a new mode of input for mobile devices other than tactile and speech input. Most modern phones are equipped with Inertial Measurement Units (IMUs) that can measure changes in the orientation and position of the mobile device. This data could be used to predict a user's intended gesture, which can be viewed as a series of positions and orientations of a device. Accurately

* Corresponding author.
E-mail address: kitsuchart@it.kmitl.ac.th (K. Pasupa).

https://doi.org/10.1016/j.heliyon.2024.e27108
Received 5 October 2023; Received in revised form 21 February 2024; Accepted 23 February 2024

Available online 29 February 2024

predicting a gesture requires a model to remember information from past time steps. Echo state networks (ESNs), a Recurrent Neural Network (RNN) type, are a potential solution for this task due to their ability to model time series data.

ESNs [1] are a specific implementation of Reservoir Computing (RC) paradigm [2]. In Reservoir Computing, during training, only the readout weights are altered and the recurrent layer (reservoir) weights remain unchanged. The reservoir can generate rich and diverse nonlinear dynamics in response to input signals, capturing the temporal dependencies of time series data and making them effective at time series prediction [3,4]. The term "echo state" originates from the idea that the recurrently connected neurons in the reservoir will have dynamics that can echo the history of the input, due to the intrinsic memory properties of the reservoir. Only the readout layer of an ESN is trained by linear regression, making it much faster than other RNNs such as Long Short-Term Memory (LSTM) models [5].

Past research into continuous gesture recognition [6] was limited by a lack of hyperparameter tuning. Before [6] was published, Tietz et al. [7] used grid search; however, the reservoir size was fixed at 400. Additionally, past research did not experiment with different variants of ESN reservoirs and readouts, which could result in better performance. Lastly, all past researches followed the leave-one-out cross-validation (LOOCV) strategy where four out of five test subjects were used for training, and one for testing [6–8], with no examination of the impact on performance of inclusion of test-subject-specific data into the training dataset.

In [8], we experimented with different types of readouts and hyperparameter tuning for continuous gesture recognition. This work built further upon that research by experimenting with different types of reservoirs and experiment schemes and analyzing the model's behaviour space. The contributions from this research are listed in the following manner:

1. Instead of using conventional grid search for hyperparameter tuning, we employ Bayesian Optimization. This optimization method leads to a significant improvement in the performance of the ESN model on continuous gesture recognition.
2. To address the variability in gesture data across different users, we use various experiment schemes to evaluate how training a model with data from other users can affect its predictions for a particular user. We found that training and testing models using data from a single user provided a significant improvement in performance, resulting in nearly perfect accuracy.
3. We conduct behaviour space analysis in terms of the Memory Capacity (MC), Kernel Rank (KR), and Generalization Rank (GR) of the ESNs to understand model performance better. The behaviour space analysis provided insight as to how good models can be selected on the basis of their behaviour space. Models with a higher MC and lower KR yield better results.

This paper is structured as follows: the following section briefly overviews related work. This is followed by the materials and methods section, which provides an overview of ESNs, the different types of ESNs used in this research, the evaluation scheme used to obtain performance metrics, and the dataset. The experiments section explains the experiment schemes that were used, hyperparameter tuning, and how behaviour space analysis was conducted. This is followed by an evaluation of the results from this research as well as from past research. The paper concludes with a discussion of the results and potential future work.

2. Related work

Tietz et al. [7] developed a framework for continuous gesture recognition and an app to obtain user gesture recognition data to create the training dataset. That dataset was relevant since it was used later in [6,8] and this study. The dataset consisted of nine input features obtained using the IMU, and the target comprised a series of gestures and "no gesture" data points. Due to the difficulty that models face in identifying the beginning and end of a gesture, a modification scheme was proposed. In this scheme, chunks of time steps where any output neuron's activity was above a threshold of 0.5 were considered predictions corresponding to the dominant class in the model activities in that chunk. An ESN model was trained on the dataset and achieved an F_1-score of 0.734 and an accuracy of 0.846.

Jirak et al. [6] built upon the work of Tietz et al. [7]. An LSTM and ESN models were trained on the dataset from [7]. The LSTM model was found to be able to better deal with sub-gestures and gesture variability, achieving an accuracy of 0.93 and an F_1-score of 0.87. In contrast, the ESN model yielded an accuracy of 0.87 and an F_1-score of 0.78. Notably, the LSTM model's training time was considerably longer at 88.9 seconds compared to the ESN model, which took 2.6 seconds.

In our previous research [8], we aimed to build on the works of [6] and [7] by improving the performance of ESNs on the dataset provided in [7]. Our approach involved experimenting with the First-Order, Reduced, and Controlled Error (FORCE) readout method [9] and optimizing hyperparameters using Bayesian Optimization. As a result, our improved ESN model achieved an F_1-score of 0.87, which is equivalent to the LSTM model's score of [6]. Furthermore, the training time for our model was much lower, taking only 13 seconds.

Wang and Ma [10] attempted to perform gesture recognition using data obtained from a wearable sensor with an IMU. They used two types of gestures: single gestures that could belong to one of ten classes and continuous combinations of the ten single gestures. They employed Principal Component Analysis [11] and Linear Discriminant Analysis [12] to extract useful features. The resulting feature vectors were used to train a Support Vector Machine [13] to predict the user's gestures. To measure the similarity between temporal sequences of varying speed, Dynamic Time Warping [14] was employed. The model achieved perfect accuracy on single gesture predictions, 87% accuracy for the prediction of a sequence of predefined gestures, and 60% accuracy for predictions of random gesture combinations.

Given the potential for improvement in ESNs, it is worth considering other techniques and mechanisms that could enhance their performance. This research examines the impact of techniques such as IP, NVAR, and FORCE learning on ESN performance. In the remainder of this section, we elaborate further on the research about these techniques.

IP was introduced in [15] to regulate the excitability of reservoir neurons, similar to visual cortical neurons. Schrauwen et al. demonstrated the effectiveness of IP in [16], where they examined the impact of Gaussian and exponential IP in RC on three different benchmarks. The tasks used to evaluate IP were MC [15], NARMA task [17], and a task to recognize isolated digits from speech. IP-ESN achieved the best performance in all tasks, with exponential IP being the most effective in the speech task and Gaussian IP being the most effective in the first two tasks.

A limitation of RC is that it relies on randomly sampled matrices to define the underlying RNN and has many hyperparameters that need optimization. It has been demonstrated that RCs are identical to NVAR machines [18]. NVAR constructs a feature vector that includes k time-delayed observations of the dynamical system under study, along with the nonlinear functions derived from these observations [19]. In [19], it is demonstrated that NVAR excels at RC benchmarks while requiring less training data and training time.

In [9], the authors propose a solution called FORCE learning for training recurrently connected neural networks that exhibit chaotic dynamics, a common feature of spontaneously active neural circuits. FORCE learning is able to transform chaotic spontaneous activity into desired activity patterns by modifying the synaptic strengths of neural networks [9]. FORCE learning operates differently from traditional neural network training, which aims to reduce errors in network output slowly. In contrast, FORCE learning aims to minimize the modification amount required to keep errors small. After enough weight modifications, the network is able to generate the desired output autonomously. FORCE learning has been demonstrated to produce complex output patterns, such as human running and walking [9].

In conclusion, the existing literature demonstrates the effectiveness of ESNs and LSTM models in performing gesture recognition using IMU data. However, there are several areas where improvements can be made. Firstly, previous literature has not explored different variants of ESN reservoirs and readouts. Secondly, previous literature has often not fully explored the potential of hyperparameter tuning. Lastly, while LOOCV has been employed in prior research, the impact of including data from the test subject in the training data has not been thoroughly examined.

3. Materials and methods

Our study aimed to explore how different reservoir and readout types impact the performance of ESNs on continuous gesture recognition tasks. The model fit was assessed using the F_1-score. We chose to focus on ESNs due to their demonstrated proficiency in handling time-series data, which was crucial in gesture recognition tasks. In sections 3.1 to 3.3, we described ESNs, the different types of reservoirs and readouts as well as behaviour space analysis. ESNs consist of three main components: input, reservoir, and readout layers, with only the readout weights being adjustable during training. We provide the activation equations for these components and illustrate the ESN architecture in Fig. 1. The readouts and reservoirs investigated included the traditional reservoir, the IP reservoir, the Next Generation Reservoir Computing (NG-RC), the Ridge Regression readout, and the FORCE readout. These variants represented a spectrum of approaches within the ESN framework, each with distinct characteristics that could influence performance outcomes in gesture recognition tasks. We explain these ESN components and the equations governing their activations in detail in section 3.2 and 3.3. Additionally, we explain the three metrics (MC, KR, and GR) used during behaviour space analysis in section 3.4. These metrics were selected because they were able to measure the models' ability to recall the past, richness in reservoir dynamics, and ability to generalize to new data, respectively, offering a holistic view of the model's capabilities. Lastly, we elaborate on the dataset used in this study in section 3.5.

3.1. Echo state networks (ESNs)

An ESN consists of three parts: input, reservoir, and readout. During training, only the weights of the readout can be adjusted using the input data, while the input and reservoir weights remain fixed [1]. Consider an ESN with L output units, N reservoir units and K input units in Fig. 1. The activations (outputs) of the input, reservoir, and output units at time step t can be represented as (1),

$$\begin{aligned}\boldsymbol{u}(t) &= [u_t(t), \ldots, u_K(t)]^\top,\\ \boldsymbol{x}(t) &= [x_1(t), \ldots, x_N(t)]^\top,\\ \boldsymbol{y}(t) &= [y_1(t), \ldots, y_L(t)]^\top.\end{aligned} \tag{1}$$

We can define the matrices holding the weights between the ESN parts as follows: W^{in} represents the connections between input and internal units and is of shape $N \times K$; W represents the internal connections of the reservoir and is of shape $N \times N$; W^{out} represents the connections of the reservoir and input units to the output units and is of shape $L \times (K + N)$ matrix; and W^{back} represents the connections from the output units to the reservoir and is of shape $N \times L$ matrix. If f and g are activation functions, typically sigmoid, and $\boldsymbol{z}(t)$ is the concatenation of the reservoir and input states, the activation of reservoir and output units can be represented as (2),

$$\begin{aligned}\boldsymbol{x}(t+1) &= f(\mathrm{W}^{in}\boldsymbol{u}(t+1) + \mathrm{W}\boldsymbol{x}(t) + \mathrm{W}^{back}\boldsymbol{y}(t)),\\ \boldsymbol{y}(t+1) &= g(\mathrm{W}^{out}\boldsymbol{z}(t)).\end{aligned} \tag{2}$$

Using a fixed reservoir and a trainable readout makes ESNs more efficient to train than traditional RNNs. However, ESNs are sensitive to the weight initialization of the reservoir, as the weights cannot be modified afterwards. As such, parameters like reservoir connectivity and spectral radius need to be carefully chosen to arrive at a good reservoir for the training task.

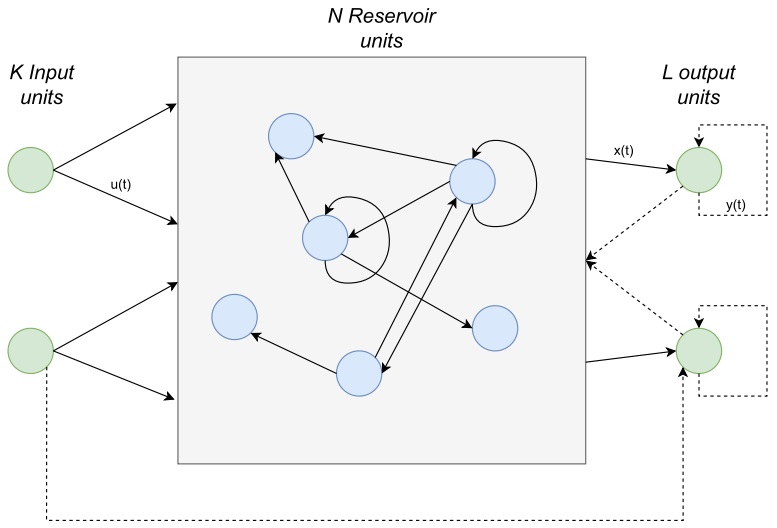


Fig. 1. ESN architecture: Connections that are possible but not required are represented with dashed arrows.

3.2. Reservoirs

3.2.1. Reservoir

The reservoir in an ESN is comprised of a collection of leaky-integrator neurons, which are interconnected randomly. These neurons serve as dynamic memory, capturing and processing temporal information from input data. The states of the neurons in the reservoir $\boldsymbol{x}$ evolve over time based on the internal dynamics, input signals, and feedback connections. The update rule for neuron states in the reservoir can be represented as (3),

$$\boldsymbol{x}(t+1) = (1-\gamma)\boldsymbol{x}(t) + \gamma\left(\mathrm{W}^{in}(\boldsymbol{u}(t+1) + c_{in}\xi) + \mathrm{W}\boldsymbol{x}(t) + \mathrm{W}^{back}(g(\boldsymbol{y}(t)) + c_{fb}\xi) + \boldsymbol{b}_{in}\right) + c\xi, \tag{3}$$

where γ refers to the leak rate; c_{in} is input noise; ξ refers to random noise; g is the identity activation function; c_{fb} is the feedback noise; $\boldsymbol{b}_{in}$ refers to input bias vector; and c is the reservoir noise.

Compared to the previous ESN description, here we have introduced a leak rate that allows the reservoir state to retain a fraction of its previous state, as well as noise terms and a bias term, making the reservoir dynamics richer and more complex.

3.2.2. Intrinsic plasticity (IP) reservoir

In biological neurons, IP is a mechanism that allows neurons to adjust their responsiveness to incoming signals. This helps the neuron adapt to different input patterns and optimize its performance in transmitting the information. Triesch developed a gradient-based method for IP in reservoirs that modifies the parameters of a sigmoid nonlinearity to push the distribution of neuron firing towards an exponential pattern [15]. IP-based reservoirs have been shown to perform better on RC tasks such as NARMA and MC [20]. In IP, the reservoir neuron states $\boldsymbol{x}$ are updated based on (4) and (5),

$$\boldsymbol{x}(t+1) = f(a\boldsymbol{r}(t+1) + b) + c\xi, \tag{4}$$

and

$$\boldsymbol{r}(t+1) = (1-\gamma)\boldsymbol{r}(t) + \gamma\left(\mathrm{W}^{in}(\boldsymbol{u}(t+1) + c_{in}\xi) + \mathrm{W}\boldsymbol{x}(t) + \mathrm{W}^{back}(g(\boldsymbol{y}(t)) + c_{fb}\xi) + \boldsymbol{b}_{in}\right). \tag{5}$$

Here, $\boldsymbol{r}$ refers to the internal activation vector of the reservoir. The activation vector $\boldsymbol{r}$ in the IP reservoir is calculated very similarly to the reservoir state $\boldsymbol{x}$ of the reservoir with leaky integrator neurons. The difference is that $\boldsymbol{r}$ is now passed through an activation function using gain a and bias b, where a and b are obtained using the Gaussian distribution learning rule explained in [15]. The activation functions used here are tanh and linear for f and g, respectively.

IP contributes to the self-regulation of neurons. Optimizing their firing rates in this manner improves information processing and enhances the performance of ESNs, particularly in dynamic environments. Nevertheless, the inclusion of IP increases the computational cost of training ESNs and introduces new hyperparameters that require tuning.

3.2.3. Next generation reservoir computing (NG-RC)

NG-RC relies on NVAR instead of a reservoir to produce feature vectors for the readout layer. NVAR is a generalization of the Vector Autoregression (VAR) model that allows for nonlinear terms in the regression equation. VAR is a statistical model used in econometrics to capture linear interdependencies among multiple time series [18]. These feature vectors consist of time-delayed observations of the input as well as nonlinear functions of the input. It has been demonstrated in [19] that NG-RC excelled at

RC benchmarks and required shorter training data and training times. Additionally, it has comparatively fewer hyperparameters that need to be tuned (delay, order, and stride). The feature vector $\mathbb{O}_{total}$ can be calculated as shown in (6), where $\oplus$ denotes concatenation,

$$\mathbb{O}_{total} = \mathbb{O}_{lin}(t) \oplus \mathbb{O}^{\delta}_{nonlin}(t). \tag{6}$$

The linear features $\mathbb{O}_{lin}$ can be obtained using the inputs based on (7),

$$\mathbb{O}_{lin}(t) = \boldsymbol{u}(t) \oplus \boldsymbol{u}(t-s) \oplus \boldsymbol{u}(t-2s) \oplus \ldots \oplus \boldsymbol{u}(t-(k-1)s), \tag{7}$$

where, s is the stride between delayed inputs, and k is the delay. Nonlinear representations $\mathbb{O}^{\delta}_{nonlin}$ are obtained by constructing all unique monomials of order δ from the inputs as shown in (8),

$$\mathbb{O}^{\delta}_{nonlin}(t) = \mathbb{O}_{lin}(t) \otimes \mathbb{O}_{lin}(t) \overbrace{\otimes \cdots \otimes}^{\delta-1 \text{ times}} \mathbb{O}_{lin}(t), \tag{8}$$

where $\otimes$ is the operator denoting an outer product followed by the selection of all unique monomials generated by this outer product.

3.3. Readouts

3.3.1. Ridge regression readout

In machine learning, ridge regression [21] is a type of linear regression that introduces regularization to the loss function. Doing so improves the model's generalization capability by preventing overfitting. In the context of ESNs, ridge regression can be used in the readout layer to map the reservoir activity to the desired output. The weights of the output layer are determined as shown in (9),

$$\mathrm{W}^{out} = \mathrm{Y}\mathrm{X}^{\top} \left(\mathrm{X}\mathrm{X}^{\top} + \lambda \mathrm{I}\right)^{-1}, \tag{9}$$

where X refers to the accumulation of the reservoir states during training; $\mathrm{X} = [\boldsymbol{x}, \ldots, \boldsymbol{x}_S]^{\top}, \mathrm{X} \in \mathbb{R}^{N \times S}$ training; Y represents the cumulative collection of all targets throughout the training process; $\mathrm{Y} = [\boldsymbol{y}, \ldots, \boldsymbol{y}_S]^{\top}, \mathrm{Y} \in \mathbb{R}^{L \times S}$; S is the number of samples; λ denotes the ridge regularization parameter; and I represents the identity matrix. Thus, the outputs $\boldsymbol{y}$ of the node are calculated as (10),

$$\boldsymbol{y} = \mathrm{W}^{out}\boldsymbol{x} + \boldsymbol{b}_{in}. \tag{10}$$

Ridge regression offers the advantage of computational simplicity, making it well-suited for handling the high-dimensional output of the reservoir. Furthermore, when confronted with noisy data, the utilization of ridge regression enhances the stability of the output. However, a drawback of ridge regression is its inherent lack of feature selection; it includes all features in the model.

3.3.2. FORCE readout

RNNs face certain challenges when training. These include feeding back the erroneous output into the network and determining which neurons are most responsible for errors and in need of tuning. The FORCE learning algorithm was developed to overcome these challenges [9]. The FORCE learning algorithm enables RNNs to generate complex and controllable patterns of activity in the absence or presence of an input signal. In our experiments, the input signal for the FORCE-based readout layer is the activation vector from the reservoir. Unlike the regression approach, where weight learning is one-shot, FORCE learning iteratively updates the weights of an RNN to predict the gesture based on the reservoir activity. At each time step, the network output is determined using reservoir activity $\boldsymbol{r}(t)$ by calculating $\mathrm{W}^{\top}(t-\Delta t)\boldsymbol{r}(t)$, where W is the weight matrix and Δt is the time delay. The error $\boldsymbol{e}\_(t)$ is then calculated by comparing the output with the desired output $\boldsymbol{y}(t)$, as shown in (11),

$$\boldsymbol{e}\_(t) = \mathrm{W}^{\top}(t-\Delta t)\boldsymbol{r}(t) - \boldsymbol{y}(t). \tag{11}$$

The weights are then updated based on (12),

$$\mathrm{W}(t) = \mathrm{W}^{\top}(t-\Delta t) - \boldsymbol{e}\_(t)\mathrm{P}(t)\boldsymbol{r}(t), \tag{12}$$

where $\mathrm{P}(t)$ is an $N \times N$ matrix, and N is the size of the network. $\mathrm{P}(t)$ is updated at the same time as the weight update matrix according to (13),

$$\mathrm{P}(t) = \mathrm{P}(t-\Delta t) - \frac{\mathrm{P}(t-\Delta t)\boldsymbol{r}(t)\boldsymbol{r}^{\top}(t)\mathrm{P}(t-\Delta t)}{1 + \boldsymbol{r}^{\top}(t))\mathrm{P}(t-\Delta t)\boldsymbol{r}(t)}. \tag{13}$$

Here $\mathrm{P}(0) = \mathrm{I}/\alpha$, I is an identity matrix, and α is a constant. The error after the weight update $\boldsymbol{e}_{+}(t)$ can be calculated using (14),

$$\boldsymbol{e}_{+}(t) = \mathrm{W}^{\top}(t)\boldsymbol{r}(t) - \boldsymbol{y}(t). \tag{14}$$

The weight update is aimed at reducing the errors by making $|\boldsymbol{e}_{+}(t)| < |\boldsymbol{e}\_(t)|$ and achieving a state where the weight vector no longer needs to be updated so that training can be stopped [9].

The drawback of this weight update method lies in its computational complexity, leading to high training times. In contrast to ridge regression, where the weight update is performed in a single shot, this method involves iterative updates, resulting in significantly longer processing times.

3.4. Behaviour space analysis

Behaviour space analysis can provide insight into factors that contribute to a model's performance. Moreover, metrics obtained from behaviour space analysis can be used to optimize models and achieve better performance, as demonstrated in [22]. This study examined the models using a behaviour space that measured their MC, KR, and GR.

3.4.1. Memory capacity (MC)

The MC of a reservoir represents its ability to represent historical inputs. Having a higher MC is important for achieving good performance on time series tasks that require a model to recollect information from the past. To calculate the MC, the reservoir is trained to repeat the input signal with a delay k, and the squared correlation between the actual output $\boldsymbol{y}(t-k)$ and network output $\hat{\boldsymbol{y}}_k(t))$ is measured [1]. The squared correlation for all values of k above 0 is summed to get the MC. MC can be calculated using (15),

$$\mathrm{MC} = \sum_{k=1}^{\infty} \frac{cov^2(\boldsymbol{y}(t-k), \hat{\boldsymbol{y}}_k(t))}{\sigma^2(\boldsymbol{y}(t-k))\sigma^2(\hat{\boldsymbol{y}}_k(t)))}, \tag{15}$$

where cov^2 and σ^2 refer to the covariance and variance operator, respectively.

3.4.2. Kernel rank (KR)

KR is a measure of the level that a model can distinguish between distinct input streams by producing distinct outputs [23]. A higher KR is better for performance since that indicates a better reservoir ability to separate inputs, making it easier for the readout to classify them. To measure the KR of a reservoir R on a dataset with S input streams $u_1, \ldots, u_S$, pass each input stream through the reservoir and get the reservoir state $x_1, \ldots, x_S$ at the end of each input stream. Using these states, construct a matrix M of shape $N \times S$ where N is the number of neurons in the reservoir. The columns of this matrix represent the reservoir states for each of the input streams. The KR of R is measured as the rank of the matrix M, as shown in (16),

$$\mathrm{KR} = rank(\mathrm{M}). \tag{16}$$

3.4.3. Generalization rank (GR)

GR measures the ability of a reservoir to generalize over a dataset. This metric is measured very similarly to KR, with the exception of the input streams being made noisy [24]. This is done by adding an interference signal (ϵ) to the input streams, as shown in (17),

$$\mathrm{GR} = rank(\mathrm{M} + \epsilon). \tag{17}$$

The interference signal is a uniformly distributed random signal within the range of 0 to 0.4. This interval was selected because the GR it produced made distinguishing good models from bad ones easier. A lower GR is desirable because it indicates that the network has a higher ability to generalize [24].

3.5. Dataset

The experimental procedures in this study were executed utilizing the gesture recognition dataset as provided in [6]. The data consisted of eleven classes identifying various gestures; the classes are listed as follows: snap right, snap left, snap backward, snap forward, bounce down, bounce up, rotate vertical, rotate horizontal, shake forward-backward, shake left-right, and no gesture. The dataset contained roughly as many data points representing a gesture as it does representing no gesture. Additionally, the ten types of gestures are represented approximately equally in the dataset. Nine variables were measured utilizing the IMU of a smartphone: the orientation axes (x, y, z), the acceleration along these axes, and the rotation velocity along these axes. The dataset consisted of gestures from five test subjects. Each test subject performed ten gestures ten times. This led to the generation of 500 sequences with variable lengths. Each of the input variables was normalized such that the highest value of each variable is 1.

4. Experiments

In this section, we describe our experimental setup, which was designed to investigate the performance of various ESN architectures and experiment schemes on a gesture recognition dataset. Our study included three different experiment schemes, which aimed to assess the usability of training data from other users in a model tested on a specific user. The goal was to identify the real-world applicability of these ESN architectures. We used the dataset and evaluation scheme from [6]. To tune the hyperparameters of the reservoirs and readouts used in the experiments, we apply Bayesian Optimization.

Table 1
Optimized parameters specific to each component type of the ESN.

	Component	Parameters
Reservoir	Reservoir	Spectral radius
		Leak rate
		Number of units
	IP-Reservoir	Spectral radius
		Leak rate
		Number of units
		Mean of target distribution (μ)
		Variance of target distribution (σ)
		Learning rate
	NG-RC	Delay
		Strides
Readout	Ridge regression	Ridge (regularization parameter)
	FORCE	Alpha (learning rate)

4.1. Experimental framework

In this research, we experiment with three schemes. The first (Global Model-1) follows the setup from prior research [6–8]. The latter two were introduced to match real-world scenarios where the model is trained on data from the end users. The different schemes are described as follows:

- Global Model-1: Global Model-1 follows the LOOCV protocol, similar to previous research [6]. In LOOCV, given n subjects, n folds are used. For each fold, $n-2$ sets are used for model training, one for validation during hyperparameter tuning and one for testing. This scheme is a standard approach to model evaluation and enables comparison with prior studies.
- Global Model-2: Global Model-2 also employs the LOOCV protocol but with a slight modification. At each fold, 20% of each test subject's data is reserved for testing, 20% for validation, and the remaining 60% for training. This approach more closely resembles real-world scenarios where partial data from end-users may be available for model training.
- User-Specific Model: The User-Specific Model scheme trains an ESN model using data from only one test subject, applying the LOOCV protocol. This approach aimed to assess the model's performance when trained and tested on data from the same user. This can help determine the model's ability to adapt to individual user characteristics and preferences.

To put this into practical use, new mobile devices can be equipped with a gesture recognition model that is pre-trained using the Global Model-1 scheme. As the user continues to use the device and more data is gathered, newer models can be trained and employed for the device based on the Global Model-2 or User-Specific Model schemes. This data-driven approach can enhance the model's performance by tailoring it to the specific characteristics and preferences of the user. It can be expected that the performance (F_1-score) in the different schemes will follow the following order, from worst to best: Global Model-1, Global Model-2, and User-Specific Model. The usefulness of data from other users can impact the disparity between models, with less helpful data leading to a larger disparity. Fig. 2 illustrates how the data was divided on each fold of cross-validation for each of the previously proposed schemes.

4.2. Evaluation scheme

An evaluation scheme for continuous gesture recognition is proposed in [7]. Therefore, this study uses the same evaluation scheme. The evaluation scheme calculates the activity $a(t)$ at every time step by applying ReLU non-linearity to the network output $y(t)$. If the sum of all elements of $a(t)$, $sum(a(t))$ is greater than 0.4, the time step marks the beginning of a gesture prediction. The gesture ends at time step $t+n$, where $sum(a(t+n))$ is less than 0.4. The class of this gesture can be obtained by summing all the activities from time step t to time step $t+n$ and selecting the class corresponding to the highest activity. Fig. 3 shows an example of classifying a prediction using this evaluation scheme.

4.3. Hyperparameter tuning

ReservoirPy [25] was used for implementing the ESNs in this study. Reservoirpy is a Python library for implementing RC models like ESNs. ReservoirPy provides support for IP reservoirs, as well as different kinds of readouts, such as ridge regression and FORCE readout. In this experiment, we aimed to optimize the various forms of ESN — with various components. Therefore, we needed to find optimal hyperparameters of each model, which were tuned using Bayesian Optimization in this study. The parameters optimized using Bayesian optimization are shown in Table 1 for each type of reservoir and readout.

Bayesian Optimization is a powerful technique that can be used to efficiently search for optimal solutions in complex search spaces and improve performance [26]. It constructs a statistical model using past evaluations. The model is able to link hyperparameters to the likelihood of achieving a particular score on the objective function. This model is simpler to optimize and is known as a surrogate

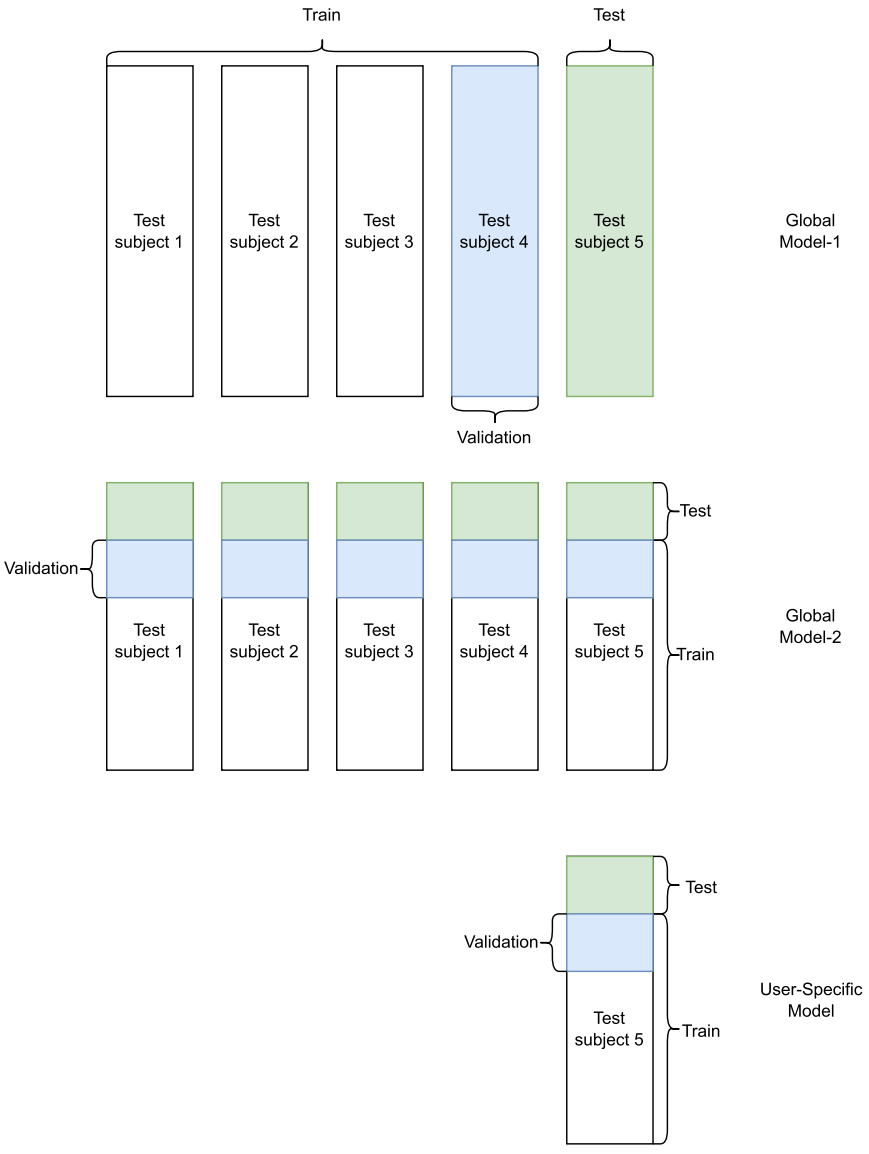


Fig. 2. Data split configuration during a cross-validation fold for each experiment scheme, including Global Model-1, Global Model-2, and User-Specific Model.

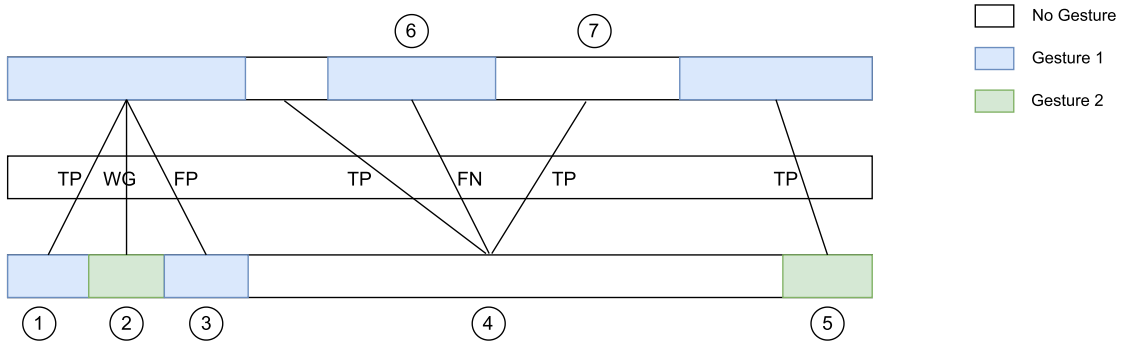


Fig. 3. Figure showing an example of the mapping scheme. The target gesture sequence is shown on top, and the predicted sequence is at the bottom. The predictions are labelled as TP (True Positive), FP (false positive), FN (false negative), and WG (wrong gesture).

Algorithm 1 Bayesian Optimization.

Require: Objective function f, maximum number of iterations $maxiter$, and number of random hyperparameters n
Ensure: Optimal hyperparameters p^*
Create n random hyperparameters $P_0 = p_1, p_2, \ldots, p_n$
Initialize dataset $D_0 = (p_i, f(p_i))\ \forall i \in 1, 2, \ldots, n$
Initialize surrogate model M_0 using D_0
Set $t = 0$
while $t < maxiter$ **do**
    Choose next hyperparameters to evaluate p_{t+1} by maximizing acquisition function over M_t
    Evaluate $f(p_{t+1})$ to get actual score s_{t+1}
    Augment dataset $D_{t+1} = D_t \cup (p_{t+1}, s_{t+1})$
    Update model M_{t+1} using D_{t+1}
    Set $t = t + 1$
end while
return $\text{argmax}_{(p,s)\in D_t}\ s$

Table 2
Parameter ranges used for Bayesian Optimization.

Parameter	Min Range	Max Range
Spectral radius	0.5	2
Leak rate	0	1
Number of units	100	1000
μ	−0.2	0.2
σ	0	2
Learning rate	1×10^{-5}	1×10^{-2}
Delay	1	10
Strides	1	10
Ridge	0	1×10^{-4}
Alpha	1×10^{-7}	1×10^{-5}

Table 3
Performances of different models using Global Model-1 scheme averaged over 10 runs, with standard deviation in brackets. Highlighting in bold represents the best performance. ESN-OPT is used to distinguish the optimized ESN model of this research from past research.

Experiment Scheme	Model	F_1-score	Accuracy	Train Time
Global-1	ESN [6]	0.78 (0.09)	0.87 (0.03)	**3.10 (0.03)**
	ESN [7]	0.73 (0.03)	0.85 (0.02)	-
	LSTM [6]	0.87 (0.03)	0.93 (0.04)	63.30 (27.50)
	ESN-OPT	0.86 (0.06)	0.93 (0.03)	13.20 (2.80)
	IP-ESN	**0.89 (0.05)**	**0.93 (0.03)**	22.50 (4.6)
	NG-RC	0.83 (0.04)	0.91 (0.02)	12.19 (2.33)
	ESN-FORCE	0.88 (0.06)	0.93 (0.03)	171.90 (83.90)

for the objective function. During our experiments, the F_1-score is used as the objective function. The Bayesian Optimization process is described in Algorithm 1. In this research, n and $maxiter$ from the algorithm are both 15. The ranges used for optimization in Bayesian Optimization are shown in Table 2.

The experiments in this research were conducted on a PC with the following specifications: a CPU with Intel Core i7-9750HF, a GPU with NVIDIA GeForce GTX 1650 (4 GB display memory), an operating system running Windows 11 Home 64-bit, and RAM with a capacity of 16 GB. The code for the experiments is available at https://github.com/aloksGithub/gesture_recognition.

5. Results and discussion

In this section, we discuss the performances of the models under the different experiment schemes as well as the model performances for each test subject. Additionally, we show the model performance for the IP-ESN model using a confusion matrix to highlight where the models struggle to make accurate predictions. Next, we discuss our behaviour space analysis, where we examine the relationships between the different behaviour space metrics and error rates. Lastly, we discuss the implications and limitations of this study.

5.1. Experimental results

Tables 3 and 4 illustrate the performances of all the models utilized in this study on the three proposed schemes, as well as the results from [6]. The models' performance was measured in terms of F_1-score, accuracy, and training time.

With regards to Global Model-1, the new ESN models were able to significantly outperform the previous models and matched the performance of the LSTM model. The F_1-score was improved by 13.9% and accuracy was improved by 6.8% as compared to ESN

Table 4
Performance of different models using Global Model-2 and User-Specific model scheme averaged over ten runs, with standard deviation in brackets. Highlighting in bold represents the best performance.

Experiment Scheme	Model	F_1-score	Accuracy	Train Time
Global-2	ESN-OPT	**0.96 (0.06)**	**0.98 (0.03)**	**16.97 (2.23)**
	IP-ESN	0.95 (0.06)	0.97 (0.03)	28.54 (4.60)
	NG-RC	0.92 (0.08)	0.96 (0.04)	19.50 (2.10)
	ESN-FORCE	0.96 (0.05)	0.98 (0.03)	289.45 (188.53)
User-Specific Model	ESN-OPT	0.98 (0.03)	0.99 (0.02)	**4.40 (0.58)**
	IP-ESN	0.97 (0.07)	0.98 (0.04)	7.31 (1.83)
	NG-RC	0.98 (0.03)	0.98 (0.02)	4.68 (1.38)
	ESN-FORCE	**0.99 (0.02)**	**0.99 (0.01)**	71.40 (45.27)

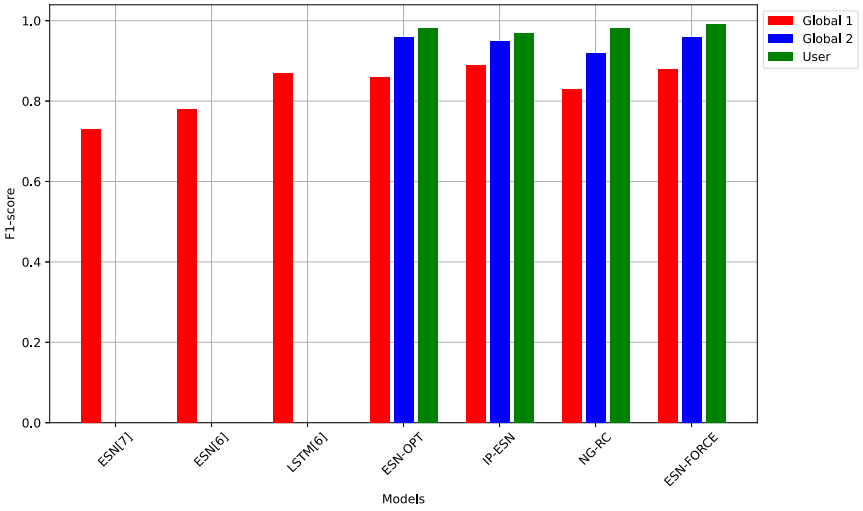


Fig. 4. Bar chart comparing the F_1-scores of models from past research and this research when using the different experiment schemes.

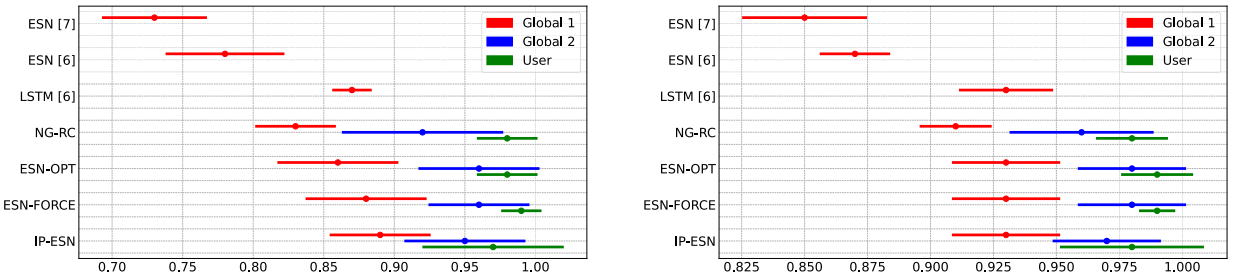
performance in [6]. Among the new ESN models, IP-ESN achieved the highest performance with an F_1-score of 0.89. However, all the ESN-based models had a fairly high standard deviation in performance. Thus, the p-value of comparing IP-ESN to the mean of the other ESN models using the t-test was 0.10. Consequently, it cannot be confidently asserted that IP-ESN was the best model. Jirak et al. [6] and Tietz et al. [7] used the same experiment setup as Global Model-1. The p-value when comparing the ESN from [6] and [7] to IP-ESN using the t-test was less than 0.001, suggesting that this comparison is statistically significant.

The performances of Global Model-2 and User-Specific Model showed a significant increase in the results shown in Fig. 4. Global Model-2, which was trained on all test subjects rather than just four out of five users, and a portion of each test subject was used for validation and testing in 5-fold cross-validation, resulting in higher performance. However, the models trained using the User-Specific Model scheme achieved the highest performance, with ESN-FORCE almost reaching perfect accuracy. This is because the models were trained and tested on the same test subject with no data from other test subjects as in Global Model-2. Additionally, the standard deviation of the results for the User-Specific Model scheme (excluding IP-ESN) is much lower than the other schemes, suggesting that models trained using this scheme are more robust. The difference in performance between these training schemes highlights the fact that there is significant variability in the way different users perform gestures. This variability reduces the usefulness of data from different users for predicting the performance of a single user.

The IP-ESN model performed best on Global Model-1, but its performance relative to others declined, resulting in the worst performance using the User-Specific Model training scheme. Additionally, the benefit that IP-ESN gained in switching from Global Model-2 to the User-Specific model was only about 0.012. This perhaps points to the IP-ESN model's ability to generalize better to previously unseen data since, in Global Model-1, none of the data from the subject used for testing was made available to the models during training in contrast to Global Model-2 and User-Specific Model where test set and training set included samples from the same test subjects. However, this cannot be said with high certainty due to the high standard deviation in results.

Training times for the User-Specific Model scheme were considerably shorter, as models were trained on a single test subject at a time, unlike in previous experiment schemes where four or five test subjects were trained at once.

Figs. 5a and 5b display the 95% confidence intervals for the F_1-score and accuracy of the models. Non-overlapping confidence intervals imply a statistically significant difference between the means, while overlapping intervals suggest that the means are not significantly different. For both F_1-score and accuracy within the same experiment scheme, the majority of models do not exhibit a statistically significant difference in their means, except for ESN [6] and ESN [7] in Global Model-1. Both of these models



(a) F_1-score mean with 95% confidence interval

(b) Accuracy mean with 95% confidence interval

Fig. 5. Visualization of 95% confidence intervals for average F_1-score and accuracy across Models. Overlapping intervals indicate statistically insignificant differences in model performance.

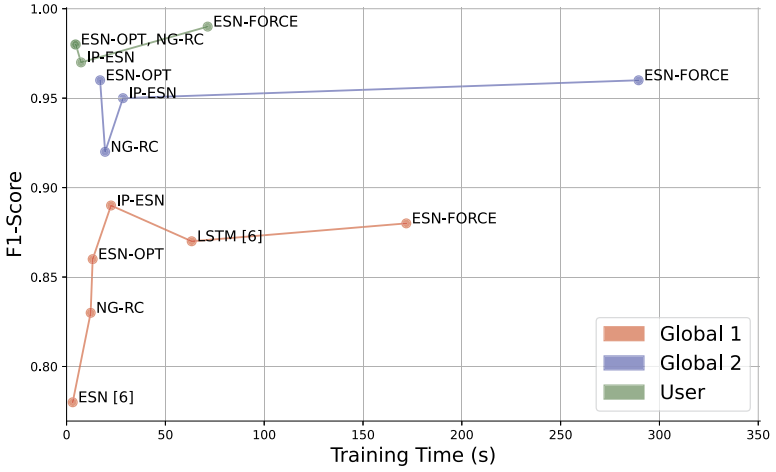


Fig. 6. Graph depicting the relationship between training time and F_1-score for the models.

Table 5
F_1-scores and standard deviations of models for each test set using the Global Model-1 scheme.

Model	J	Ni	S	Na	L
ESN [6]	0.64 (0.04)	0.81 (0.05)	0.80 (0.05)	0.80 (0.06)	0.88 (0.05)
LSTM [6]	0.95 (0.02)	0.83 (0.04)	0.90 (0.08)	0.88 (0.04)	0.80 (0.04)
ESN-OPT	0.89 (0.02)	0.90 (0.04)	0.86 (0.04)	0.90 (0.03)	0.77 (0.06)
IP-ESN	0.91 (0.02)	0.88 (0.05)	0.92 (0.04)	0.90 (0.03)	0.84 (0.05)
NG-RC	0.85 (0.01)	0.85 (0.02)	0.86 (0.02)	0.83 (0.02)	0.77 (0.02)
ESN-Force	0.89 (0.03)	0.92 (0.03)	0.88 (0.05)	0.88 (0.04)	0.81 (0.06)

exhibit a significantly lower performance compared to the other models. When comparing the means of models across experiment schemes, Global Model-2 and User-Specific Model consistently outperform Global Model-1, except in the case of IP-ESN. However, the difference between the same models when using Global Model-2 and User-Specific Model is not significant. In addition, the training times for the models are plotted against the F_1-score of each model, emphasizing the enhancement in both F_1-score and training time achieved through the implementation of the User-Specific scheme as shown in Fig. 6. Models trained with the User-Specific scheme outperform those trained with the other two schemes, demonstrating superior performance while requiring significantly less training time. This advantage is attributed to their training on a single user.

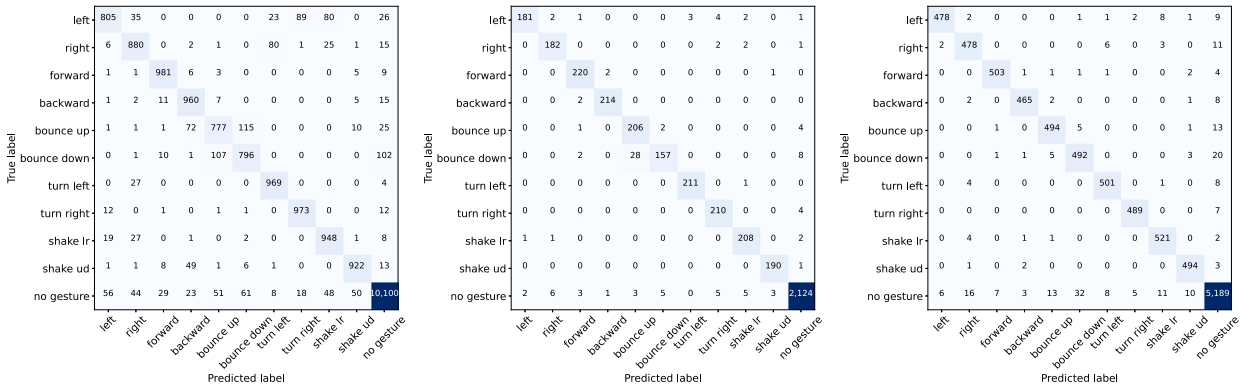
Tables 5, 6 and 7 show the F_1-scores for each model on each of the test subjects in Global Model-1, Global Model-2, and User-Specific Model, respectively. In the Global Model-1 scheme, most of the models seemed to struggle with test subject 'L' except for the ESN [6]. This was because test subject 'L' performed each gesture similarly, making it difficult to distinguish gestures [6]. The performance of ESN [6] on test subject 'L' was high because the mapping algorithm in [6] was tailored to the ESN output. The application of Global Model-2 and User-Specific Model training schemes was able to overcome this issue as seen in Tables 6 and 7, respectively. This is because the data distribution in the test set for subject 'L' was significantly different from the other test sets. The models in the Global Model-1 scheme did not have access to data from the specific user being used for testing during training, while the models in the Global Model-2 scheme and User-Specific Model did.

Table 6
F_1-scores and standard deviations of models for each test set using the Global Model-2 scheme.

Model	J	Ni	S	Na	L
ESN-OPT	0.98 (0.03)	0.99 (0.02)	0.98 (0.04)	0.90 (0.07)	0.96 (0.05)
IP-ESN	0.98 (0.04)	0.97 (0.04)	0.97 (0.05)	0.89 (0.08)	0.97 (0.04)
NG-RC	0.98 (0.03)	0.95 (0.05)	0.96 (0.05)	0.87 (0.07)	0.86 (0.09)
ESN-Force	0.98 (0.03)	0.98 (0.03)	0.97 (0.04)	0.91 (0.06)	0.95 (0.06)

Table 7
F_1-scores and standard deviations of models for each test set using the User-specific model scheme.

Model	J	Ni	S	Na	L
ESN-OPT	0.99 (0.02)	0.99 (0.02)	0.98 (0.03)	0.98 (0.02)	0.97 (0.05)
IP-ESN	0.94 (0.10)	0.97 (0.07)	0.99 (0.02)	0.94 (0.06)	0.99 (0.02)
NG-RC	0.98 (0.02)	0.97 (0.03)	0.97 (0.04)	0.97 (0.04)	0.99 (0.02)
ESN-Force	0.98 (0.02)	0.99 (0.02)	0.99 (0.02)	0.98 (0.03)	0.99 (0.02)



(a) Global Model-1 scheme (b) Global Model-2 scheme (c) User-Specific Model scheme

Fig. 7. Confusion matrices showing the IP-ESN model's predictions for each of the three schemes.

5.2. Confusion matrix

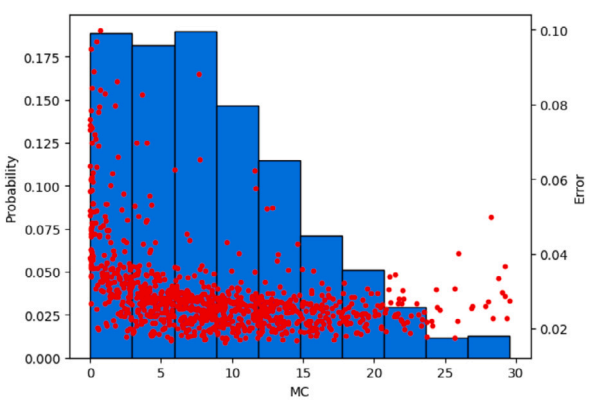
The IP-ESN model's predictions on the three schemes are represented as confusion matrices in Figs. 7a, 7b, and 7c. The confusion matrix offers insight as to where the models struggle with making predictions. In Global Model-1, a lot of misclassifications occurred when the model incorrectly predicted noise as a gesture. Misclassifications also commonly occur when a gesture contains sub-gestures or when gestures share sub-gestures. This can be seen in the bounce up and down classes, as well as the shake left-right class being confused with the left class. Global Model-2 sees a significant reduction in these errors; the only persistent issues are 'bounce down' being confused with 'bounce up' and the model predicting noise as a gesture. The User-Specific Model sees a further improvement in the prediction of the 'bounce' gesture; it fails to detect gestures more often and falsely detects a gesture when there is none.

5.3. Behaviour space analysis

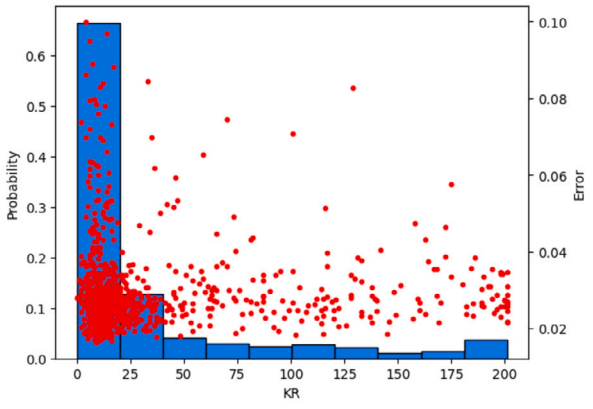
Behaviour space analysis was conducted on a thousand IP-ESN models that were generated using the hyperparameter optimization of the Global Model-1 experiments while trying to optimize the Mean Squared Error (MSE), as shown in Fig. 8. IP-ESN was chosen since it had the best performance on the Global Model-1 training scheme.

Fig. 8a shows that models with a low MC were more likely to have a higher error rate. This is expected since predicting a user's gesture requires information from previous time steps. Thus, if a model was unable to recall past information efficiently, it would perform poorly. This is reflected in the figure, as models with lower MC are more likely to exhibit poorer performance. A high MC is difficult to attain since the task of recalling information from the past gets more difficult with more time steps. This results in fewer models having a higher MC, as shown in the figure.

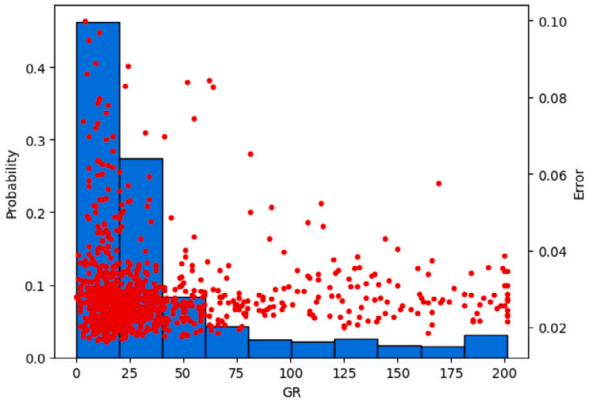
Figs. 8b and 8c compare the KR and GR with the error rate, respectively. Since GR is just KR measured with noisy data, the graphs show a similar trend, with the GR distribution being a bit more dispersed, as evident from the histogram. The correlation between KR and GR is 0.93, as can be seen from Fig. 8e. Since KR measures a reservoir's ability to separate between model inputs, it is expected that a model with a higher KR will have a higher performance. This trend is reflected in Fig. 8b where models with a low KR are more prone to a high error rate. The high probability for the models to have a low KR and low GR suggests that our models belong to an ordered dynamical regime rather than a chaotic one [27].



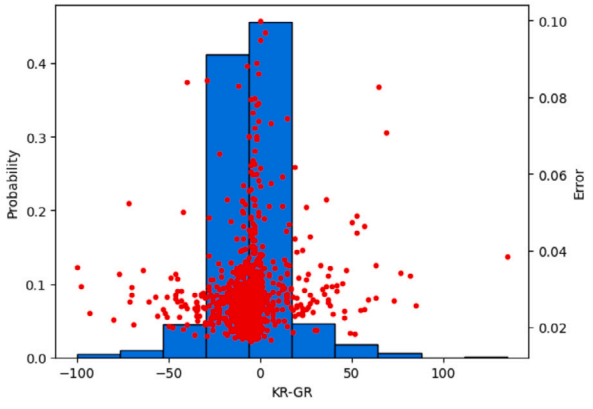
(a) Relationship between MC and MSE



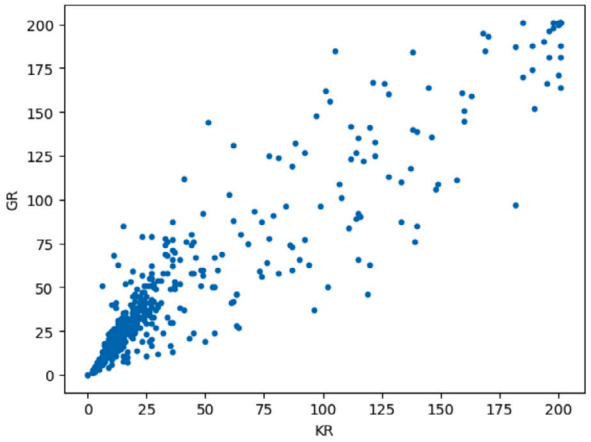
(b) Relationship between KR and MSE



(c) Relationship between GR and MSE



(d) Relationship between KR−GR and MSE



(e) Relationship between KR and GR

Fig. 8. This figure illustrates the effect of varying space construction parameters on the error of an ESN reservoir and distribution when applied to continuous gesture recognition. The results are based on 1000 randomly generated parameter sets.

Fig. 8d further examines the relationship between KR, GR, and error by plotting the difference between KR and GR against the error. The graph suggests that a large difference between KR and GR results in a lower error rate.

5.4. Implications

The results of our study have implications for both machine learning and mobile technology. Firstly, our findings reveal that, when properly optimized, ESNs can serve as a lightweight and viable alternative to LSTMs, achieving comparable performance. In

the context of mobile devices, the gesture recognition capabilities of ESNs have the potential to enhance user experience, facilitating more intuitive and fluid interactions with smartphones and other mobile devices.

5.5. Limitations

While this study provides valuable insights into the application of ESNs for continuous gesture recognition, it is important to acknowledge certain limitations. Firstly, the dataset consisted of data from only five test subjects, limiting the generalizability of our findings. A more extensive and diverse dataset may yield varying results. Secondly, Bayesian Optimization involves sequentially generating and evaluating models (from step 5 onward of Algorithm 1), which cannot be parallelized. This limits the speed at which good models can be obtained. Lastly, although we observed ESNs performing better than LSTMs, the difference is not statistically significant. Thus, their superiority should be interpreted with caution.

6. Conclusion

In this research, we experimented with different ESN variants and experiment schemes to enhance the performance of ESNs on continuous gesture recognition. Additionally, we performed a behaviour space analysis to better understand the ESN models. The models were tested with three schemes following the LOOCV protocol: Global Model-1, which used one subject for testing, one for validation, and the remaining subjects for training; Global Model-2, where at each fold, 20% of each test subject's data was used for testing, 20% for validation, and the remaining 60% for training; and User Specific Model, in which a model is trained and tested using data from only one test subject. The application of IP was able to perform slightly better with the Global Model-1 scheme; however, this performance did not persist in the other two experiment schemes. The differences in performances between the training schemes showed that users perform gestures quite differently, resulting in poor generalization performance when the training dataset has no data from the subject used for testing. In conclusion, the User-Specific model proved to be the most effective training scheme, achieving the highest performance by leveraging training and test datasets obtained from a single test subject, thereby making them more similar compared to the datasets used in other schemes. Additionally, the User-Specific model exhibited the lowest training times, attributed to its utilization of a smaller training dataset.

In addition, the behaviour space analysis showed that good models can be identified using the metrics MC, KR, and GR. In the future, the hyperparameter optimization process can be changed to use these metrics rather than F_1-score. KR and GR are faster to calculate since they only involve passing the input through the reservoir as opposed to F_1-score and MSE, which require model training and prediction on input. Additionally, MC is independent of the dataset; thus, models with pre-calculated MCs can be used during the optimization process. These alterations can greatly speed up the optimization process.

Additional information

No additional information is available for this paper.

CRediT authorship contribution statement

Alok Yadav: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Kitsuchart Pasupa:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Resources, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Chu Kiong Loo:** Writing – review & editing, Supervision, Conceptualization. **Xiaofeng Liu:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data included in article/supp. material/referenced in article.

Acknowledgements

This work was supported by the School of Information Technology, King Mongkut's Institute of Technology Ladkrabang.

References

[1] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks, GMD-Report 148, German National Research Institute for Computer Science, 1 2001.

[2] D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt, An experimental unification of reservoir computing methods, Neural Netw. 20 (3) (2007) 391–403, https://doi.org/10.1016/j.neunet.2007.04.003.

[3] H. Jaeger, M. Lukoševičius, D. Popovici, U. Siewert, Optimization and applications of echo state networks with leaky-integrator neurons, Neural Netw. 20 (3) (2007) 335–352, https://doi.org/10.1016/j.neunet.2007.04.016.
[4] M.H. Tong, A.D. Bickett, E.M. Christiansen, G.W. Cottrell, Learning grammatical structure with echo state networks, Neural Netw. 20 (3) (2007) 424–432, https://doi.org/10.1016/j.neunet.2007.04.013.
[5] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735.
[6] D. Jirak, S. Tietz, H. Ali, S. Wermter, Echo state networks and long short-term memory for continuous gesture recognition: a comparative study, Cogn. Comput. 10 (2020), https://doi.org/10.1007/s12559-020-09754-0.
[7] S. Tietz, D. Jirak, S. Wermter, A reservoir computing framework for continuous gesture recognition, in: Proceedings of the International Conference on Artificial Neural Networks (ICANN 2019) Workshop and Special Sessions, 2019, pp. 7–18.
[8] A. Yadav, K. Pasupa, C.K. Loo, Revisiting echo state networks for continuous gesture recognition, in: Proceedings of the 2022 IEEE Symposium Series on Computational Intelligence (SSCI), 2022, pp. 978–983.
[9] D. Sussillo, L. Abbott, Generating coherent patterns of activity from chaotic neural networks, Neuron 63 (2009) 544–557, https://doi.org/10.1016/j.neuron.2009.07.018.
[10] Y.-T. Wang, H.-P. Ma, Real-time continuous gesture recognition with wireless wearable IMU sensors, in: Proceedings of the 2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom 2018), 2018, pp. 1–6.
[11] S. Mishra, U. Sarkar, S. Taraphder, S. Datta, D. Swain, R. Saikhom, S. Panda, M. Laishram, Principal component analysis, Int. J. Livest. Res. 1 (2017), https://doi.org/10.5455/ijlr.20170415115235.
[12] A. Tharwat, T. Gaber, A. Ibrahim, A.E. Hassanien, Linear discriminant analysis: a detailed tutorial, AI Commun. 30 (2017) 169–190, https://doi.org/10.3233/AIC-170729.
[13] M.A. Hearst, Support vector machines, IEEE Intell. Syst. Appl. 13 (4) (1998) 18–28, https://doi.org/10.1109/5254.708428.
[14] C.S. Myers, L.R. Rabiner, A comparative study of several dynamic time-warping algorithms for connected-word recognition, Bell Syst. Tech. J. 60 (7) (1981) 1389–1409, https://doi.org/10.1002/j.1538-7305.1981.tb00272.x.
[15] J. Triesch, A gradient rule for the plasticity of a neuron's intrinsic excitability, in: W. Duch, J. Kacprzyk, E. Oja, S. Zadrozny (Eds.), Proceedings of the International Conference on Artificial Neural Networks (ICANN 2005), 2005, pp. 65–70.
[16] B. Schrauwen, M. Wardermann, D. Verstraeten, J.J. Steil, D. Stroobandt, Improving reservoirs using intrinsic plasticity, Neurocomputing 71 (7) (2008) 1159–1171, https://doi.org/10.1016/j.neucom.2007.12.020.
[17] A.F. Atiya, A.G. Parlos, New results on recurrent network training: unifying the algorithms and accelerating convergence, IEEE Trans. Neural Netw. 11 (3) (2000) 697–709, https://doi.org/10.1109/72.846741.
[18] E.M. Bollt, On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD, Chaos 31 (1) (2020) 013108.
[19] D.J. Gauthier, E. Bollt, A. Griffith, W.A.S. Barbosa, Next generation reservoir computing, Nat. Commun. 12 (1) (9 2021), https://doi.org/10.1038/s41467-021-25801-2.
[20] X. Wang, Y. Jin, K. Hao, Echo state networks regulated by local intrinsic plasticity rules for regression, Neurocomputing 351 (2019) 111–122, https://doi.org/10.1016/j.neucom.2019.03.032.
[21] A.E. Hoerl, R.W. Kennard, Ridge regression: biased estimation for nonorthogonal problems, Technometrics 12 (1) (1970) 55–67.
[22] Z. Zhang, Y. Zhu, X. Wang, W. Yu, Optimal echo state network parameters based on behavioural spaces, Neurocomputing 503 (2022) 299–313, https://doi.org/10.1016/j.neucom.2022.06.008.
[23] R. Legenstein, W. Maass, Edge of chaos and prediction of computational performance for neural circuit models, Neural Netw. 20 (3) (2007) 323–334, https://doi.org/10.1016/j.neunet.2007.04.017.
[24] L. Buesing, B. Schrauwen, R. Legenstein, Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons, Neural Comput. 22 (2009) 1272–1311, https://doi.org/10.1162/neco.2009.01-09-947.
[25] N. Trouvain, L. Pedrelli, T.T. Dinh, X. Hinaut, ReservoirPy: an efficient and user-friendly library to design echo state networks, in: I. Farkas, P. Masulli, S. Wermter (Eds.), Proceedings of the 29th International Conference on Artificial Neural Networks ICANN 2020, Bratislava, Slovakia, September 15-18, 2020, Proceedings, Part II, in: Lecture Notes in Computer Science, vol. 12397, Springer, 2020, pp. 494–505.
[26] F. Nogueira, Bayesian optimization: open source constrained global optimization tool for Python, https://github.com/fmfn/BayesianOptimization, 2014.
[27] M. Dale, J.F. Miller, S. Stepney, M.A. Trefzer, A substrate-independent framework to characterise reservoir computers, CoRR, arXiv:1810.07135 [abs], 2018, arXiv:1810.07135.