Multi-layered Echo State Machine: A novel Architecture and Algorithm

Zeeshan Khawar Malik, Member *IEEE*, Amir Hussain, Senior Member *IEEE*
and Qingming Jonathan Wu[1], Senior Member *IEEE*
University of Stirling, UK and University of Windsor, Ontario, Canada[1]
Email: zkm, ahu@cs.stir.ac.uk and jwu@uwindsor.ca[1]

***Abstract*—In this paper, we present a novel architecture and learning algorithm for a multi-layered Echo State Machine (ML-ESM). Traditional Echo state networks (ESN) refer to a particular type of Reservoir Computing (RC) architecture. They constitute an effective approach to recurrent neural network (RNN) training, with the (RNN-based) reservoir generated randomly, and only the readout trained using a simple computationally efficient algorithm. ESNs have greatly facilitated the real-time application of RNN, and have been shown to outperform classical approaches in a number of benchmark tasks. In this paper, we introduce novel criteria for integrating multiple layers of reservoirs within an echo state machine, with the resulting architecture termed the ML-ESM. The addition of multiple layers of reservoirs are shown to provide a more robust alternative to conventional reservoir computing networks. We demonstrate the comparative merits of this approach in a number of applications, considering both benchmark datasets and real world applications.**

I. Introduction

In machine learning and neural networks communities, several neural network models and kernel-based methods are applied to time series prediction tasks, such as MLPs (Multi-Layer Perceptrons) [1], RBF (Radial Basis Function) neural network [2], Extreme Learning Machine [3][4], deep learning autoencoders [5], FIR (Finite Impulse Response) neural network [6], SVR (Support Vector Regression) [7], SOM (Self-Organization Map) [8], GP (Gaussian Process) echo state machine [9], SVESM (Support Vector Echo State Machine) [10], RNNs (Recurrent Neural Networks) including NAR (Nonlinear AutoRegressive network) [11], Elman networks [12], Jordan networks, RPNN (Recurrent Predictor Neural Networks) [13] and ESN (Echo State Network) [14]. A number of alternative approaches are also being reported [15], [16], [17], [18] and [19].

Over the last decade, the echo state network has been recognised as the most efficient network structure for training RNNs. It was invented independently in the seminal works of Jaeger [20], who termed these RNNs: "echo state networks (ESNs). Maass *et al.* [21] developed a similar approach for spiking neural networks and termed the derived model: "liquid state machine" (LSM). These two pioneering methodologies have given rise to the novel paradigm of reservoir computing (RC) [22].

The standard echo state machines are state-space models with fixed state transition structures (the reservoir) and an adaptable readout form for the state space. Adding more connected layers inside the state space of fixed-state transition structures are expected to improve the overall performance of the model. Sequentially connecting each fixed-state transition structure externally with other transition structures creates a long-term memory cycle for each. This gives the ML-ESM proposed in this article, the capability to approximate with better accuracy, compared to state-of-the-art ESN based approaches.

Echo State Network [20] is a popular type of reservoir computing network mainly composed of three layers of 'neurons': an input layer, which is connected with random and fixed weights to the next layer, and forms the reservoir. The neurons of the reservoir are connected to each other through a fixed random, sparse matrix of weights. Normally only about 10 % of the weights in the reservoir are non-zero. The weights from the reservoir to the output neurons are trained using error descent. Only weights from the reservoir to the output node are trainable. In this paper we present a novel multiple layer reservoir network of ESM. To achieve this, we have introduced a new set of non-trainable weights which leads to a stronger synaptic connection between neurons of more than one reservoir, as part of the ESM.

We first define the idea of a reservoir as follows: $\mathbf{W}_{in}$ represents the weight from the $\mathbf{N_u}$ inputs $\mathbf{u}$ to the $\mathbf{N_x}$ reservoir units $\mathbf{x}$, $\mathbf{W}$ indicates the $\mathbf{N_x} \times \mathbf{N_x}$ reservoir weight matrix, and $\mathbf{W}_{out}$ indicates the $(\mathbf{N_x}+1) \times \mathbf{N_y}$ weight matrix connecting the reservoir units to the output units, denoted by $\mathbf{y}$. Typically $\mathbf{N_x} \ll \mathbf{N_u}$. $\mathbf{W}_{in}$ is fully connected and the weights are trainable.

The supervised training and testing of the echo state network is conducted by updating the reservoir state and network output as follows:

$$\begin{aligned} \mathbf{x}(t+1) &= (1-\gamma)h(\mathbf{W}\mathbf{x}(t) + \mathbf{W}_{\text{in}}\mathbf{u}(t+1)) \\ &\quad +\mathbf{W}_{\text{out}}\tilde{\mathbf{y}}(t)) + \gamma\mathbf{x}(t) \qquad (1) \\ \mathbf{y}(t+1) &= \mathbf{W}_{\text{readout}}[\mathbf{x}(t+1); \mathbf{u}(t+1)] \qquad (2) \end{aligned}$$

where $\mathbf{x}(t)$ is the state of the reservoir at time t, $\mathbf{W}$ is the weight matrix of neurons inside the reservoir i.e. the matrix of the weights of the synaptic connection between the reservoir neurons. $\mathbf{u}(t)$ is the input signal fed to the network at time t. $\tilde{\mathbf{y}}(t)$ is the target value of the readout (i.e., the desired network output) at time t. $\mathbf{y}(t)$ is the predicted value of the readout at time t. $\gamma \geq 0$ is the retainment rate of the reservoir network (with $\gamma > 0$ if leaky integrator neurons are considered). $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_{\text{out}}$ are the randomly initialized weights of $\mathbf{u}(t)$ and $\mathbf{y}(t)$, and $h(.)$ is the activation function of the reservoir. In this paper we will be considering hyperbolic tangent reservoir neurons, i.e. $h(.) = tanh(.)$.

To achieve an echo state property, the reservoir connection matrix $\mathbf{W}$ is typically scaled as $\mathbf{W} \leftarrow \alpha\mathbf{W}/\lambda_{\text{max}}$, where $|\lambda_{\text{max}}|$ is the spectral radius of $\mathbf{W}$ and $0 < \alpha < 1$ is a scaling parameter [23]. Echo State network training is essentially based on teacher-forced calculation of the corresponding reservoir states $\{\mathbf{x}(t)\}_{t=1}^{T}$ using (1), and application of a simple regression algorithm (e.g. linear regression or ridge regression) to train the readout weights $\mathbf{W}_{\text{readout}}$ on the resulting dataset $\{\mathbf{x}(t)\}_{t=1}^{T}$ [24]. All the weight matrices to the reservoir $(\mathbf{W}, \mathbf{W}_{\text{in}}, \mathbf{W}_{\text{out}})$ are initialized randomly. The state of the reservoir is initially set to zero i.e. $x(0) = \mathbf{0}$.

The rest of this paper is organized as follows:

The second section explains how a multiple layer echo machine can be trained in a supervised manner. The natural approach here is to adapt only the weights of the multiple layer network to output connections. Essentially, this trains readout functions which transform the multiple layer echo state into the desired output signal. This section also defines multiple layer echo states and provides equivalent characterizations.

The third section presents a comparative analysis of the proposed method with a number of state-of-the-art benchmark approaches, considering Mackey-Glass Series dataset, Henon map, NARMA sequence, an artificially generated Figure 8 dataset, 15 classification problems, Reuter-21578 dataset, and finally by predicting human motion using the Carnegie Melon University (CMU) MoCap dataset.

Section four concludes with a comparative discussion and some future work suggestions.

II. Multi-layered Echo State Machine (ML-ESM)

Historically, Minsky and Papert [25] are known to have left open the possibility that multi-layer networks may be capable of delivering better performance by overcoming the limitations of single-layered feedforward neural networks. Recently this idea has virtually exploded with impressive successes across a wide variety of applications [26] [27] and [28].

The idea of our ML-ESM can be explained through the following mathematical formulation: We consider discrete-time neural networks with K input units, R_i where $i = 1, 2, 3, ..., N$ internal reservoir units, J number of neurons inside each reservoir and L output units. Activations from the input units at time step t are $\mathbf{u}(n) = (u_1(t), , u_K(t))$. The internal units are represented as $\mathbf{x}(n) = (\mathbf{x}_1(t), ...\mathbf{x}_N(t))$ where $\mathbf{x}_1(n) = (x_{11}(t), ..., x_{1J}(t))$, and the output units are: $\mathbf{y}(n) = (y_1(t), ..., y_L(t))$. Real-valued connection weights are collected in a $J \times K$ weight matrix $\mathbf{W}^{\text{in}} = (w_{ij}^{\text{in}})$ for the input weights; in a $J \times J$ matrix $\mathbf{W}^{\text{internal}} = (w_{ij}^{\text{internal}})$ for the internal connections; in an $J \times J$ matrix $\mathbf{W}^{\text{external}} = (w_{ij}^{\text{external}})$ for the external connections; and, in a $L \times (K + \sum_{i=1}^{N} R_i + L)$ matrix $\mathbf{W}^{\text{out}} = (w_{ij}^{\text{out}})$ for the connections to the output units. Figure 1 shows the basic network architecture of the ML-ESM.

The internal activations for both training and testing of N reservoir units inside an echo state machine are updated according to:

$$
\begin{aligned}
\mathbf{x}_1(n+1) &= (1-\gamma)h(\mathbf{W}\mathbf{x}_1(t) + \mathbf{W}_{\text{internal}}\mathbf{u}(t+1)) \\
&\quad +\gamma\mathbf{x}_1(t) \\
\mathbf{x}_2(n+1) &= \gamma\mathbf{x}_1(n+1) + (1-\gamma)h(\mathbf{W}\mathbf{x}_2(t) + \\
&\quad \mathbf{W}_{\text{external}}\mathbf{x}_1(t+1)) + \gamma\mathbf{x}_2(t) \\
\mathbf{x}_3(n+1) &= \gamma\mathbf{x}_2(n+1) + (1-\gamma)h(\mathbf{W}\mathbf{x}_3(t) + \\
&\quad \mathbf{W}_{\text{external}}\mathbf{x}_2(t+1)) + \gamma\mathbf{x}_3(t) \\
\mathbf{x}_4(n+1) &= \gamma\mathbf{x}_3(n+1) + (1-\gamma)h(\mathbf{W}\mathbf{x}_4(t) + \\
&\quad \mathbf{W}_{\text{external}}\mathbf{x}_3(t+1)) + \gamma\mathbf{x}_4(t) \\
. &\quad .... \\
. &\quad .... \\
\mathbf{x}_N(n+1) &= \gamma\mathbf{x}_{N-1}(n+1) + (1-\gamma)h(\mathbf{W}\mathbf{x}_N(t) + \\
&\quad \mathbf{W}_{\text{external}}\mathbf{x}_{N-1}(t+1)) + \gamma\mathbf{x}_N(t)
\end{aligned} \tag{3}
$$

The output is computed as follows:

$$\mathbf{y}(t+1) = \mathbf{W}_{\text{readout}}[\mathbf{x}(t+1); \mathbf{u}(t+1)] \tag{4}$$

where $\gamma \geq 0$ is the retainment rate of the reservoir networks inside the echo state machine which can vary in each layer (with $\gamma > 0$ if leaky integrator neurons are considered). Initially all the weight matrices for each layer inside the echo state machine $(\mathbf{W}_{\text{internal}}, \mathbf{W}_{\text{external}})$ and $\mathbf{W}_{\text{out}}$ are initialized randomly.

To achieve an echo state property with N reservoir connections inside the ML-ESM, the internal reservoir connection matrix $\mathbf{W}_{\text{internal}}$ and the external reservoir connection matrix $\mathbf{W}_{\text{external}}$ are typically scaled as $\mathbf{W}_{\text{internal}} \leftarrow \alpha\mathbf{W}_{\text{internal}}/\lambda_{\text{in-max}}$ and $\mathbf{W}_{\text{external}} \leftarrow \alpha\mathbf{W}_{\text{external}}/\lambda_{\text{ex-max}}$, where $|\lambda_{\text{in-max}}|$ is the spectral radius of $\mathbf{W}_{\text{internal}}$ and $|\lambda_{\text{ex-max}}|$ is the spectral radius of $\mathbf{W}_{\text{external}}$.

In this paper we consider the input sequences in the first layer $(\mathbf{u}(n))_{n\epsilon J}\varepsilon U^{j}$, where U is required to be compact. We use shorthand $\bar{\mathbf{u}}^{-+\infty}$, $\bar{\mathbf{u}}^{+\infty}$, $\bar{\mathbf{u}}^{-\infty}$, $\bar{\mathbf{u}}^{-h}$ to denote input sequences for all the $\{1, 2, 3, ..., N\}$ layers, which are, respectively, left-right infinite ($J = Z$), right-infinite ($J = k, k+1, ...$ for some $k\ \epsilon$ Z), left-infinite, and finite of length h. The network state operator T is used to write $\mathbf{x}(n + h) = T(\mathbf{x}_1(n).....\mathbf{x}_N(n), \mathbf{y}(n), \bar{\mathbf{u}}^{h})$ to denote the network states, that results from an iterated application of equation (3), if the input sequence is fed into the network at time t, with resulting output $\mathbf{y}(n)$.

Thus our analysis for the ML-ESM will rely specifically on the following generic setup: (i) input to the first layer is drawn from a compact input space U; (ii) All network states of each layer lie in a compact set A. These conditions, as in [20], can be termed: standard compactness conditions.

Definition1 *Assume the network has no output feedback connections* and the network is maintaining the standard compactness condition. Then, the ML-ESM has $\{1, ..., N\}$ echo states, if the network states $\mathbf{x}_1(n)$ to $\mathbf{x}_N(n)$ are uniquely determined by any left-infinite input sequence $\bar{\mathbf{u}}^{-\infty}$.

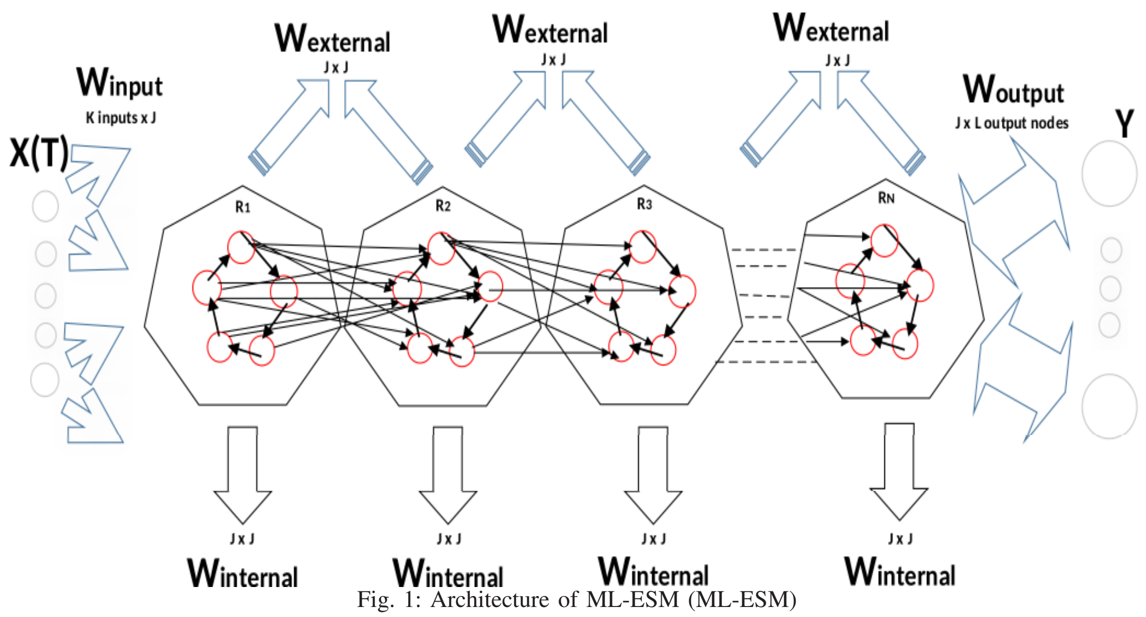


Fig. 1: Architecture of ML-ESM (ML-ESM)

Equivalently stating the echo state property for ML-ESM, is to say that there exist input echo functions $\mathbf{E} = (e_1, ..., e_n)$ for each layer of the machine, where $e_i : U^{-N} \to \mathbf{R}$, such that for all left-infinite input histories the current network state for each layer are:

$$\begin{aligned}
\mathbf{x_1}(n) &= \mathbf{E}(..., \mathbf{u}(n-1), \mathbf{u}(n)) \\
\mathbf{x_2}(n) &= \mathbf{x}_2 + \mathbf{E}(..., \mathbf{x_1}(n-1), \mathbf{x_1}(n)) \\
\mathbf{x_3}(n) &= \mathbf{x}_3 + \mathbf{E}(..., \mathbf{x_2}(n-1), \mathbf{x_2}(n)) \\
\mathbf{x_4}(n) &= \mathbf{x}_4 + \mathbf{E}(..., \mathbf{x_3}(n-1), \mathbf{x_3}(n)) \\
. &= .... \\
. &= .... \\
. &= .... \\
\mathbf{x}_N(n) &= \mathbf{x}_N + \mathbf{E}(..., \mathbf{x}_{N-1}(n-1), \mathbf{x}_{N-1}(n))
\end{aligned} \tag{5}$$

Definition2

(a) A state sequence at each layer of the ML-ESM is given by:

$$\bar{\mathbf{x}}_{\mathbf{1}}^{-\infty} = ..., \mathbf{x}_1(n-1), \mathbf{x}_1(n)$$

to

$$\bar{\mathbf{x}}_N^{-\infty} = ..., \mathbf{x}_N(n-1), \mathbf{x}_N(n) \ \epsilon \ \mathbf{A}^{-N} \tag{6}$$

(where $A \subset \mathbf{R}^L$ admissible states inside each layer) is called compatible with an input sequence from the first layer

$$\bar{\mathbf{u}}^{-\infty} = ..., \mathbf{u}(n-1), \mathbf{u}(n)$$

to the N^{th} layer

$$\bar{\mathbf{x}}_{N-1}^{-\infty} = ..., \mathbf{x}_{N-1}(n-1), \mathbf{x}(n), \tag{7}$$

if, at the first layer:

$$\forall i < n : T(\mathbf{x}_1(i), \mathbf{u}(i+1)) = \mathbf{x}_1(i+1)$$

to the N^{th} layer

$$T(\mathbf{x}_N(i), \mathbf{x}_{(N-1)}(i+1)) = \mathbf{x}_N(i+1). \tag{8}$$

A network state at each layer is end-compatible with the input sequence if there exists a state sequence at each layer

$$, ..., \mathbf{x}_1(n-1), \mathbf{x}_1(n).....\mathbf{x}_N(n-1), \mathbf{x}_N(n)$$

such that

$$\begin{aligned}
\{T(\mathbf{x}_1(i), \mathbf{u}(i+1)) &= \mathbf{x}_1(i+1) \\
......T(\mathbf{x}_N(i), \mathbf{x}_{N-1}(i+1)) &= \mathbf{x}_N(i+1), \}
\end{aligned}$$

and

$$\{\mathbf{x}_1 = \mathbf{x}_1(n)....\mathbf{x}_N = \mathbf{x}_N(n)\}. \tag{9}$$

(b) Similarly, a left-right-infinite state sequence $\bar{\mathbf{x}}^{\infty}$ is called compatible with an input sequence $\mathbf{u}^{\infty}$ starting from the first layer, if

$$\forall i : T(\mathbf{x}_1(i), \mathbf{u}(i+1)) = \mathbf{x}_1(i+1)$$

to the N^{th} layer if

$$\forall i : T(\mathbf{x}_N, \mathbf{x}_{N-1}(i+1)) = \mathbf{x}_N(i+1). \tag{10}$$

(c) A network state $\mathbf{x} \ \epsilon \ A$ is called end-compatible with an input sequence $\bar{\mathbf{u}}^{-\infty}$ if there exists a state sequence from the first layer

$$..., \mathbf{x}_{N-1}(n-1), \mathbf{x}_N(n)$$

such that

$$T(\mathbf{x}_1(i), \mathbf{u}(i+1)) = \mathbf{x}_1(i+1),$$

and

$$\mathbf{x}_1 = \mathbf{x}_1(n)$$

to the N^{th} layer

$$T(\mathbf{x}_N(i), \mathbf{u}(i+1)) = \mathbf{x}_N(i+1). \tag{11}$$

(d) A network state $\{\mathbf{x}_1....\mathbf{x}_N\}\epsilon A$ is called end-compatible with a finite input sequence $\bar{\mathbf{u}}^h$ at the first layer if there exist a state sequence:

$$\{\mathbf{x}_1(n-h), ...\mathbf{x}_1(n)\}$$

to

$$\{\mathbf{x}_N(n-h), ...\mathbf{x}_N(n)\}\epsilon A^{h+1} \tag{12}$$

such that

$$T(\mathbf{x}_1(i), \mathbf{u}(i+1)) = \mathbf{x}_1(i+1)$$

and

$$\mathbf{x}_1 = \mathbf{x}_1(n)$$

to the N^{th} layer

$$T(\mathbf{x}_N(i), \mathbf{u}(i+1)) = \mathbf{x}_N(i+1). \tag{13}$$

Metaphorically, the state $\mathbf{x}_i(n)$ of each layer can be understood as an echo of the input history. As demonstrated in [29], smaller reservoir sizes often yield better networks with higher probability. On the contrary, as noted in [14], the random connectivity and weight structure of the reservoir with a smaller size is unlikely to be optimal and does not give a clear insight into the reservoir dynamics [30]. The only way, as demonstrated in this paper, to determine the best approximation with a smaller reservoir size is by initializing more than one layer of reservoir inside an ESM.

Proposition 1 *Assume a sigmoid network at each layer with unit output function f_i =tanh. (a) Let the internal weight matrix* $\boldsymbol{W_{internal}}$ *and external weight matrix* $\boldsymbol{W_{external}}$ *satisfy $\gamma_{max} = \wedge < 1$, its largest singular value. Then $d(T(\mathbf{x}_1, \mathbf{u}), T(\mathbf{x}_1^{'}, \mathbf{u}) + ...d(T(\mathbf{x}_N, \mathbf{x}_{N-1}), T(\mathbf{x}_N^{'}, \mathbf{x}_{N-1}) < \wedge d(\mathbf{x}_1, \mathbf{x}_1^{'}) + ...d(\mathbf{x}_N, \mathbf{x}_{N-1}^{'})$ for all states $(\mathbf{x}_1, \mathbf{x}_1^{'})$ to $(\mathbf{x}_N, \mathbf{x}_N^{'})\epsilon\,[-1, 1]^N$. (b) Let the internal and external weight matrix at each layer have a spectral radius $|\lambda_{max}| > 1$, where λ_{max} is an eigenvalue of internal and external weight matrix with the largest absolute value. Then the network has an asymptotically unstable null state from 1 to N layers of an ESM.*

Proof "(a)":

$$\begin{aligned} d(T((\mathbf{x}_1, \mathbf{u}) + ...(\mathbf{x}_N, \mathbf{x}_{N-1}), T((\mathbf{x}_1^{'}, \mathbf{u}) + \\ ...(\mathbf{x}_N^{'}, \mathbf{x}_{N-1})) = d(\mathbf{f}(\mathbf{W}^{in}\mathbf{u} + \\ \mathbf{W}\mathbf{x}_1) + ...\mathbf{f}(\mathbf{W}^{ex}\mathbf{x}_{N-1} + \mathbf{W}\mathbf{x}_N), \\ \mathbf{f}(\mathbf{W}^{in}\mathbf{u} + \mathbf{W}\mathbf{x}^{'}) + ...\mathbf{f}(\mathbf{W}^{ex}\mathbf{x}_{N-1} + \mathbf{W}\mathbf{x}_N^{'})) \\ \leq d(\mathbf{W}^{in}\mathbf{u} + \\ \mathbf{W}\mathbf{x}_1) + ...\mathbf{W}^{ex}\mathbf{x}_{N-1} + \mathbf{W}\mathbf{x}_N, \\ \mathbf{W}^{in}\mathbf{u} + \mathbf{W}\mathbf{x}^{'} + ...\mathbf{W}^{ex}\mathbf{x}_{N-1} + \mathbf{W}\mathbf{x}_N^{'}) \\ = d(\mathbf{W}\mathbf{x}_1, \mathbf{W}\mathbf{x}_1^{'} + ...\mathbf{W}\mathbf{x}_N, \mathbf{W}\mathbf{x}_N^{'}) \\ = ||\mathbf{W}((\mathbf{x}_1 + ...\mathbf{x}_N)) - ((\mathbf{x}_1 + ...\mathbf{x}_N))|| \\ \leq \wedge d(\mathbf{x}_1 + ...\mathbf{x}_N, \mathbf{x}_1^{'} + ...\mathbf{x}_N^{'}) \end{aligned}$$

i.e., the distance between two states $(\mathbf{x}_1....\mathbf{x}_N, \mathbf{x}_1^{'}....\mathbf{x}_N^{'})$ shrinks by a factor $\wedge < 1$ at every step, regardless of the input.

"(b)": The null state input sequence along all the layers $\bar{\mathbf{0}}^{-\infty}\epsilon U^{-N}$ is compatible with the null state sequence $\bar{\mathbf{0}}^{-\infty}$. But if $|\lambda_{max} > 1|$ then the null state is not asymptotically stable. This implies the existence of another state sequence compatible with the null input sequence, which results in the violation of echo state property along each layer.

Proposition 2 *Assume standard compactness conditions and a multiple layer network without output feedback. Assume that T is continuous in state and input from first to last layer. Then the properties of being state contracting, state forgetting and input forgetting are all equivalent to the multiple layer network having echo states along each layer.*

Proof:

"state contracting $\Rightarrow$ echo state": Assume that the ML-ESM has no echo states along multiple layers, i.e.

$$\begin{aligned} \exists(\mathbf{x}_1, \mathbf{x}_1^{'}) + ...(\mathbf{x}_N, \mathbf{x}_N^{'}))\epsilon D^{+}, \\ d((\mathbf{x}_1, \mathbf{x}_1^{'}) + ...(\mathbf{x}_N, \mathbf{x}_N^{'})) > 2\epsilon > 0, \end{aligned} \tag{14}$$

where D^{+} is the set of all identical pairs $(\mathbf{x}_1, \mathbf{x}_1^{'})....(\mathbf{x}_N, \mathbf{x}_N^{'})$ compatible with some input sequences. This implies that there exists a strictly growing sequence $(h_i)_{i\geq 0}\epsilon\mathbb{N}^N$, finite input sequences along each layer

$$(\mathbf{u}_i^{h_i})_{i\geq 0}....(\mathbf{x}_{(N-1)i}^{h_i})_{i\geq 0},$$

state

$$((\mathbf{x}_{1i}, \mathbf{x}_{1i}^{'}).....(\mathbf{x}_{Ni}, \mathbf{x}_{Ni}^{'})), \tag{15}$$

such that

$$d((T(\mathbf{x}_{1i}, \bar{\mathbf{u}}_i^{h_i})....T(\mathbf{x}_{Ni}, \bar{\mathbf{x}}_{N-1}^{h_i})), (\mathbf{x}_1...\mathbf{x}_N)) < \epsilon$$

and

$$d((T(\mathbf{x}_{1i}^{'}, \bar{\mathbf{u}}_i^{h_i})....T(\mathbf{x}_{Ni}^{'}, \bar{\mathbf{x}}_{N-1}^{h_i})), (\mathbf{x}_1^{'}...\mathbf{x}_N^{'})) < \epsilon. \tag{16}$$

"state contracting $\Rightarrow$ state forgetting": Assume the multiple layer network is not state forgetting. This implies that there exists a left-infinite input sequence $\bar{\mathbf{u}}^{-\infty}$, a strictly growing index sequence $(h_i)_{i\geq 0}$, states $(\mathbf{x}_{1i}, \mathbf{x}_{1i}^{'}.....\mathbf{x}_{Ni}, \mathbf{x}_{Ni}^{'})$, and some $\epsilon > 0$, such that

$$\begin{aligned} \forall i : d(T(\mathbf{x}_{1i}, \bar{\mathbf{u}}^{+\infty}[-h_i], ..., \\ \mathbf{x}_{Ni}, \bar{\mathbf{x}}_{Ni-1}^{+\infty}[-h_i]), T(\mathbf{x}_{1i}^{'}, \bar{\mathbf{u}}^{+\infty}[-h_i] \\ , ..., \mathbf{x}_{Ni}^{'}, \bar{\mathbf{x}}_{Ni-1}^{+\infty}[-h_i])) > \epsilon, \end{aligned} \tag{17}$$

where ${}^{+\infty}[-h_i]$ denotes the suffix of length h_i.

"input forgetting $\Rightarrow$ echo state": Assume that the multiple layer network does not have the echo state property. Then there exist an input sequence from 1 to N-1 layer $\bar{\mathbf{u}}^{-\infty}, ..., \bar{\mathbf{x}}_{N-1}^{-\infty}$, states $(\mathbf{x}_1, \mathbf{x}_1^{'})....(\mathbf{x}_{N-1}, \mathbf{x}_{N-1}^{'})$ end-compatible with $(\bar{\mathbf{u}}^{-\infty}....\bar{\mathbf{x}}_{N-1}^{-\infty}) > 0$; which leads to a contradiction to input forgetting.

III. Experiments

In this section, we provide a thorough experimental evaluation of the ML-ESM Model, considering; 1) Mackey-Glass Series dataset, which provides the most classical benchmark task for time series modeling; 2) Henon map, which is a discrete-time dynamical system exhibiting characteristic chaotic behavior; (3) NARMA sequence which generates a sequence using a non-linear auto-regressive moving average (NARMA) model; (4) an artificially generated figure 8 dataset with the points of the figure moving around very quickly, and each cycle comprising only a few points; (5) robust evaluation of the proposed method compared with standard benchmark techniques on 15 classification problems (see Table VI); (6) Reuter-21578, a popular dataset mostly used for evaluating text mining algorithms; and (7) Finally predicting human motion using the Carnegie Melon University (CMU) MoCap dataset [31]. In all experimental evaluations, we consider reservoirs comprising analog neurons, with $tanh$ transfer functions. To demonstrate the advantages of the proposed ML-ESM, we also evaluate linear-regression based ESNs, ridge regression-based ESNs, and support vector echo state machine (SVESM) models with $\epsilon-$insensitive loss functions [10], using the same reservoirs as the evaluated ML-ESM model.

Our source codes were developed in MATLAB, and made partial use of the RC Toolbox [22]. The implementation of the SVESM method was based on the library of SVM of [32], written in C, hence providing a computationally more efficient implementation in comparison to the rest of the evaluated methods. Therefore, the execution times of the evaluated algorithm are not fully comparable. Our experiments were executed on a Intel (R) Core (TM) i7 CPU 3770 @ 3.40 GHz 3.40 GHz machine with 16.0 GB of RAM. Table I summarizes the configuration details of the employed reservoirs in the considered experiments.

In our experiments, the weights of the input $\mathbf{u}(t)$, stored in the matrix $\mathbf{W}_{in}$, the reservoir weight matrix $\mathbf{W}$ and the external reservoir weight matrix $\mathbf{W}^{ex}$ in terms of ML-ESM, are drawn randomly with a uniform distribution over (-0.1,0.1). The results provided in the remainder of this section are averaged over 5 different random reservoir initializations. Finally, training of all ESN-based methods was conducted using fivefold cross-validation.

A. *Mackey-Glass Series*

The Mackey-Glass delay differential equation has provided classical benchmark tasks for time series modeling. The Mackey-Glass delay differential equation in a discrete time setting is approximated as follows:

$$y(t+1) = y(t) + \delta \left(0.2 \frac{y(t-\tau/\delta)}{1+y(t-\tau/\delta}^{10} - 0.1y(t) \right) \quad (18)$$

where the stepsize δ typically set to $\delta = 1/10$ [20], [23]. The resulting time-series is later rescaled into the range $[-1, 1]$ by application of a tangent-hyperbolic transform $y_{ESN}(t) = tanh(y(t) - 1)$, so that it can be used to train ESNs with $tanh$ activation function in the reservoir. The system behave chaotic for values of the delay time $\tau < 16.8$.

The training sequence for our experiment was generated from equation (18) with a delay time $\tau = 17$, similar to [20] and [9]. The single and multiple layer ESN-based models were trained using a signal comprising 6000 time points, and the initial transient was washed out by employing a reservoir warm-up time of 100 steps. Subsequently, evaluation of single and multiple layer ESN-based models were conducted by simulating the trained models using a new time series of 250 samples, with a reservoir warm-up time of 100 time steps. We compare the performance of the considered models by calculating the obtained normalized absolute error (NAE) on a specific prediction horizon using the simulated network outputs. The NAE on a t-step prediction horizon reads as

$$\mathrm{NAE}_t = \sqrt{\frac{1}{s^2}(y((\mathrm{warmup}+t) - \tilde{y}(\mathrm{warmup}+t))^2} \quad (19)$$

where s^2 is the empirical variance of the actual target signal. Prediction on a t-step horizon in all the evaluated models was conducted by iteratively applying the predictor t times in a generative mode, where on each step it takes its own most recent prediction to compute the next prediction.

We consider a commonplace selection for the Mackey-Glass system prediction horizon, i.e., prediction 84 and 120 steps after the washout time elapses [20]. The obtained results are provided in Table II and Figure 2. These results are produced by calculating the mean and standard deviation of the obtained performance metrics over 50 test sequences (where execution of each test sequence includes first training of 6000 time points, with initial transient of 100 steps washed out, testing of 250 time points with a reservoir warm-up time of 100 time steps followed by a prediction horizon of 84, 120 upto 150 time steps). We observe that ML-ESM with linear and ridge-regression outperformed the considered alternatives, by producing much lower $\mathbf{NAE}_{84}$, $\mathbf{NAE}_{120}$ and $\langle \mathbf{NAE}_t \rangle$ (mean $\mathbf{NAE}_t$) values. We also find that the performance of the SVESM is worse than the performance of the rest of the considered methods. Another comment we would like to make is the overwhelming computational cost of the SVESM method. The SVESM method required more than 2.99993 x 10^3 seconds to execute only 50 test sequences for each prediction horizon (84, 120, 150) of the estimation algorithm.

Figure 3 further shows the comparison of single layer ESN with the proposed ML-ESM over 150 prediction time points of 50 test sequences. Figure 3 (a) demonstrates the results by using the linear regression technique whereas Figure 3 (b) shows results by using the ridge regression technique. It can be seen clearly in both Figure 3 (a) and (b) that adding layers in the beginning further lowers the NAE at each time step and then gradually increases with very little difference. This clearly shows that the behaviour of ML-ESM with the Mackey dataset maximally minimizes the error by adding only two layers to the network, over 150 prediction time points.

Finally, to provide a better insight into how the proposed ML-ESM method compares with its alternatives in regards to the imposed computational burden, we would like to highlight that as part of our optimized MATLAB implementation, the ridge regression-based ML-ESM required roughly 73 seconds, which is close to the ridge regression-based ESN which

TABLE I: Configuration of Reservoirs in our Experiments

Parameter	Mackey-Glass	Henon map	Figure 8	NARMA_SEQUENCE	15 Benchmark datasets	Reuter-21578	MoCap Video Dataset
Reservoir neurons	400	100	1000	100	1000	1000	100
Spectral radius	0.99	0.99	0.998	1.25	1.25	1.25	0.998
Reservoir connectivity	0.1	0.1	0.2	0.1	0.1	0.1	0.1
Warm-up time (model training)	100	100	0	0	0	0	0
Warm-up time (model evaluation)	100	100	0	0	0	0	0
γ	0.5	0.5	0.3	0.3	0.5	0.5	0.5
Number of Multiple Layers (ML-ESM)	2	2	2	2	2	2	2

required 65 seconds to execute only a 50 test sequence for each prediction horizon (84, 120, 150). This shows that the computational time required by the ML-ESM is far better than the SVESM (2.99993 x 10^3) and when compared with the single layer ESN, offers a very good trade-off between computational complexity and sequential data modeling performance.

B. Figure 8 Dataset

In this experiment, we evaluate the effectiveness of our ML-ESM model in learning complex sequential patterns. For this purpose, we consider an artificially generated figure 8 dataset with the points of the figure moving around very quickly, and each cycle comprising only a few points. To obtain this signal, we use the artificially created function figure8_dataset which generates figure 8, whose circles are centered at (384,302) and (384,722), with a radius of 210 and a channel width of 79 i.e. 1.5 cm.

The evaluated model was trained using a sequence of 600 data points from the figure 8 trajectory, and no reservoir warm up was employed. On the sequel the trained models were evaluated over 600 time steps. In Figure 4, we provide the trajectories produced by the evaluated methods. As can be observed, the ML-ESM using both linear and ridge regression technique works considerably better than the SVESM and the linear regression based ESN, and slightly better than the ridge-regression-based ESN. Specifically the ridge-regression-based ESN yields a normalized root mean square error (NRMSE) equal to 0.0085, whereas the ML-ESM using ridge-regression technique yields an NRMSE equal to 0.00002588.

C. Henon map

In this subsection, we consider another typical benchmark in the field of RNNs, the Henon map chaotic process [33]. It is a discrete-time dynamical system exhibiting a characteristic chaotic behavior. Henon map receives as input a 2-D point $\mathbf{y}(t) = [y_1, y_2]$, and maps it to a new point $\mathbf{y}(t+1) = [y_1(t+1), y_2(t+1)]$ on the 2-D plane, given by

$$y_1(t+1) = y_2(t) + 1 - \alpha y_1^2(t) \tag{20}$$
$$y_2(t+1) = \beta y_1(t) \tag{21}$$

where $\alpha = 1.4$ and $\beta = 0.3$. The starting point of the Henon map considered in our experiment is $\mathbf{y}(0) = 0$.

In our experiments, the analyzed ESN-based models, were trained using the first 1000 samples of the Henon map. The initial transient was washed out by employing a reservoir warm-up time of 100 steps. Afterwards, evaluation was conducted by using the trained models to generate the next 2000 samples of the Henon map, with the employed reservoirs being teacher-driven for the first 100 time points. The performance of the analyzed models in terms of the obtained $\mathbf{NAE}_{84}$, $\mathbf{NAE}_{120}$, $\mathbf{NAE}_{400}$ and $\langle \mathbf{NAE}_t \rangle$ metrics is depicted in Table III.

As can be observed, the ML-ESM model using both ridge and linear regression techniques, performs better than the considered alternatives. This is attributed to the externally connected structures temporally creating a long term memory cycle. This led to a reduction in the error compared to state-of-the art time series learning approaches. It is worth noting that the SVESM is the worst performing method in this experiment.

D. NARMA Sequence

In this subsection, we consider a sequence of a non-linear autoregressive moving average (NARMA) model. The sequence at the beginning includes a ramp-up transient. The output of the NARMA sequence model depends on past and present values of the input as well as the output.

$$y_i = f(y_{i-1}, ..., y_{i-M_y}, x_i, ..., x_{i-M_x}) + \eta_i \tag{22}$$

where y_i, x_i and η_i are the the output, input and noise respectively. M_x denote the output and input memory orders.

The NARMA sequence non-linear autoregressive moving average equation is approximated as

$$\mathbf{x}_i = a * \mathbf{x}_{i-M_x} + b + (1 - \mathbf{y}_{i-1}) * \mathbf{y}_{i-1} \tag{23}$$

where input sequence $\mathbf{x}$ is an array of size equivalent to twice the sequence length. Output sequence $\mathbf{y}_i$ is an array of size equivalent to the sequence length. The length of the sequence in our experiment was 1000. The constant variable a and b were initialized with 0.7 and 0.1 respectively, which were used to generate a linear sequence. The $\mathbf{NRMSE}$ and $\mathbf{MSE}$ of the ESN based models are shown in Table IV. Again the ML-ESM with linear and ridge-regression techniques out-performed the standard linear and ridge-regression based ESN as well as the SVESM model.

Figure 5 further shows the comparison of single layer ESN with the proposed ML-ESM using the same range of data samples. The left side of Figure 5 shows the results obtained by using the linear regression technique whereas the right side demonstrate the results obtained using ridge regression. It can be clearly seen in the both the sub figures (on left and right) in Figure 5 that the error first maximally decreases by using 2-layers with the proposed ML-ESM, in comparison to the single layer ESN; and then remains steady in the following layers, from three to five, with a slight and steady increase on addition of each layer. It can be clearly seen in Table IV and

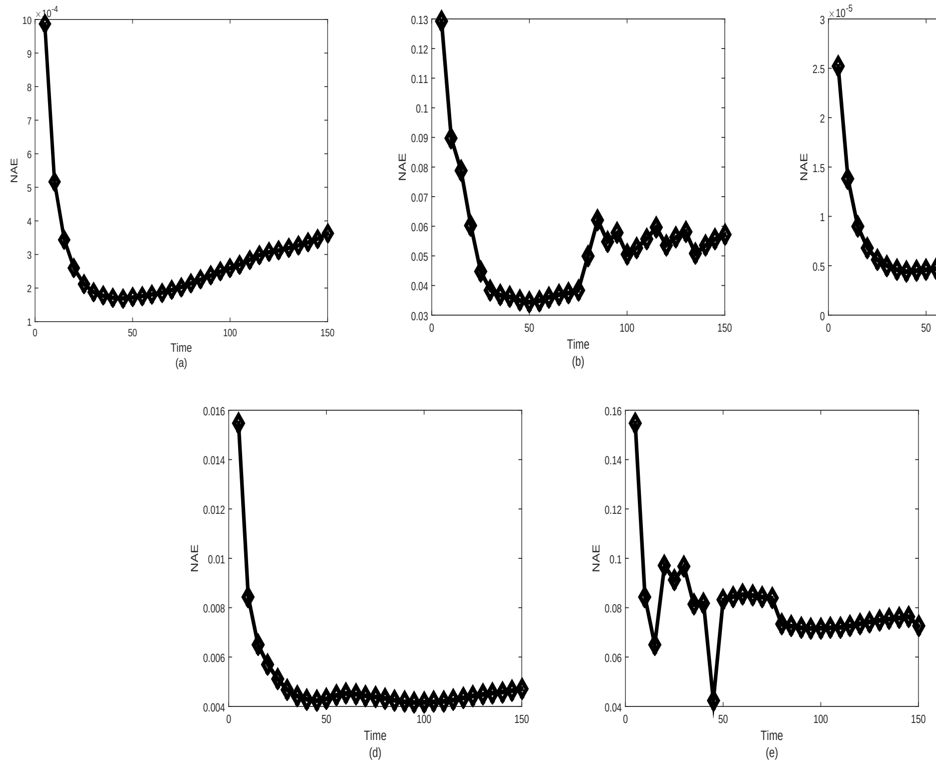


Fig. 2: Mackey-Glass series: NAE values over 150 prediction time points of the evaluated model. (a) Linear Regression ESN. (b) Ridge Regression ESN. (c) Multiple Layer Linear Regression ESN. (d) Multiple Layer Ridge Regression ESN. (e) SVESM (Support Vector Echo State Machine)

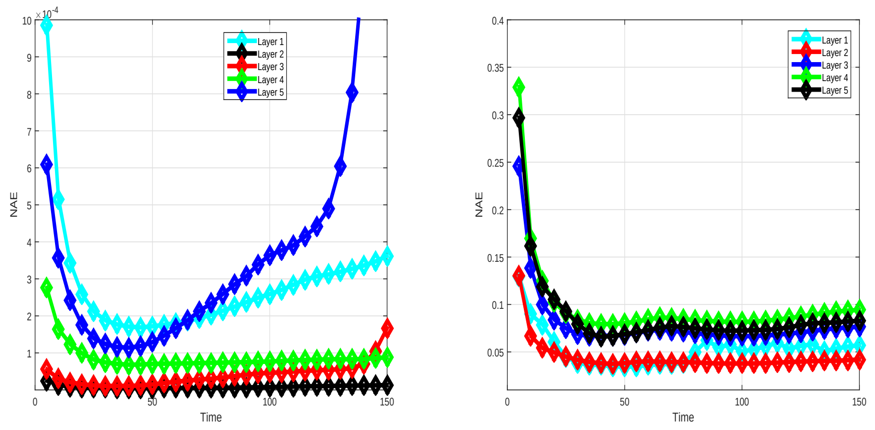


Fig. 3: Mackey-Glass series: NAE values over 150 prediction time points of the evaluated model. (a) Comparison of Single Layer ESN with Multiple Layer (2-5) ESN using Linear Regression technique (b) Comparison of Single Layer ESN with Multiple Layer (2-5) ESN using Ridge Regression technique.

TABLE II: Mackey-Glass Series: Performance of the Evaluated Models

Model	Linear-Regression-based ESN	Ridge-Regression-based ESN	Linear-Regression-based ML-ESM	Ridge-Regression-based ML-ESM	SVESM
NAE$_{84}$	1.1031 x 10^{-05} (7.7998 x 10^{-05})	0.0107 (0.0755)	**5.2633 x 10^{-06} (3.7217 x 10^{-05}**	**0.00105 (0.00752)**	0.0643 (0.0077)
NAE$_{120}$	3.0737 x 10^{-04} (4.7874 x 10^{-05}	0.0063 (0.0444)	**2.6713** x **10^{-06} (1.8889** x **10^{-05})**	**0.0021 (0.0323)**	0.0746 (0.0197)
NAE$_t$	1.592005 x 10^{-04}	0.0085	**3.9673 x 10^{-06}**	**0.001575**	0.06945

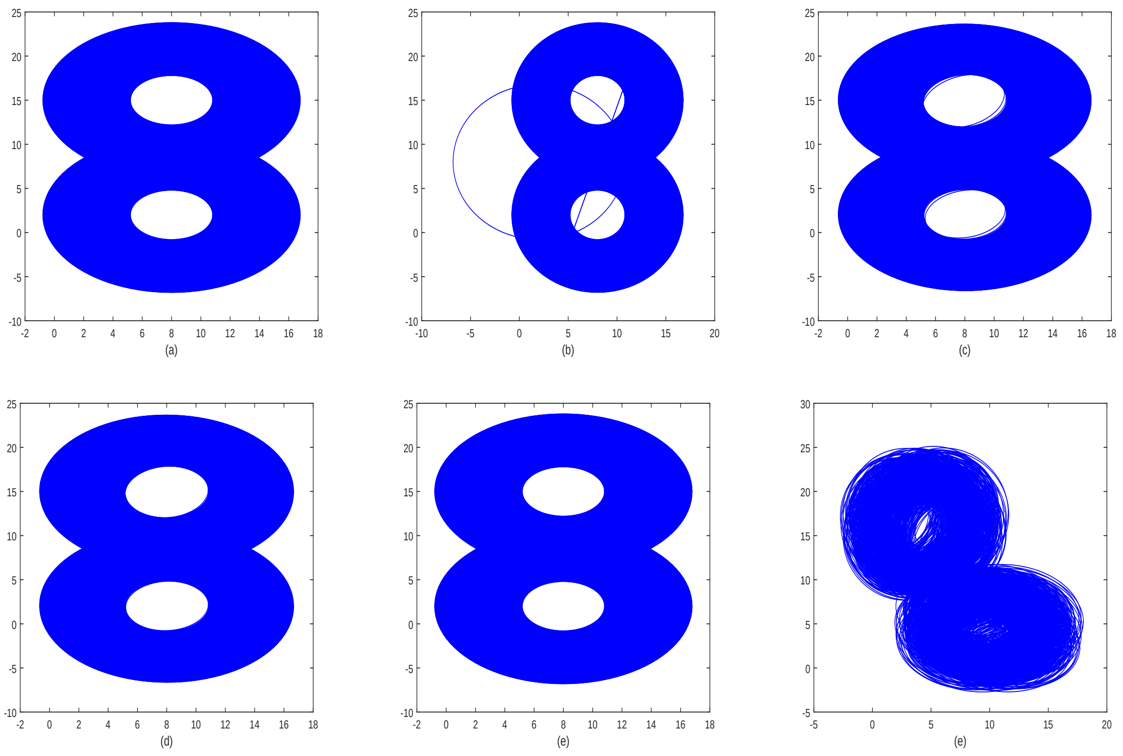


Fig. 4: Figure 8 dataset. (a) Original time series. (b) Reconstruction obtained by the ESN using linear regression. (c) Reconstruction obtained by the ESN using ridge regression. (d) Reconstruction obtained by the multiple layer ESN using linear regression. (e) Reconstruction obtained by the multiple layer ESN using ridge regression. (f) Reconstruction obtained by the SVESM (Support Vector Echo State Machine.)

TABLE III: Henon Map: Performance of The Evaluated Models

Model	Linear-Regression-based ESN	Ridge-Regression-based ESN	Linear-Regression-based ML-ESM	Ridge-Regression-based ML-ESM	SVESM
NAE$_{84}$	0.7536	0.7489	**0.7477**	**0.7661**	2.6469
NAE$_{120}$	0.7552	0.7432	**0.7481**	**0.7655**	1.3432
NAE$_{400}$	0.7813	0.7629	**0.7466**	**0.7465**	1.5134
$\langle$**NAE**$\rangle_t$	0.7633	0.7516	**0.7474**	**0.7593**	1.8345

Figure 5 that the ML-ESM produced best results with two and three layers where it significantly outperformed the standard ESN using both linear and ridge regression techniques. The SVESM model, like with the henon map dataset in section III-C, did not perform well with this NARMA sequence too. This is basically due to the poor prediction performance of the SVESM method at some time points, which adversely affects the average method performance.

E. *15 benchmark classification problems*

In this subsection, the 15 multivariate benchmark datasets are considered from UCI machine Learning repositories [34] and the [35]. The performance of all the methods including the ML-ESM and other state of the art recurrent neural network based time series learning approaches were encouraging. This is due to the time series learning nature of the proposed method and other state-of-the-art approaches which proved more useful in other time series chaotic predictive tasks, as demonstrated earlier in this paper. The proposed ML-ESM using both linear and ridge regression was still found to outperform other benchmark techniques. This occurred due to the externally connected transition structures sequentially creating a long term memory cycle, which further helped in reducing the error compared to standard state-of-the-art time series learning approaches.

F. *Reuters-21578 Textual Corpus*

Reuter-21578 [36] is a popular dataset mostly used for evaluating text mining algorithms. Due to the consistency of concepts and the connected component nature of this corpus [37] we have selected this corpus for evaluating the strength of the proposed methods in comparison to state- of-the-art.

TABLE IV: NARMA SEQUENCE: NRMSE and MSE of The Evaluated Models

Error	Linear-Regression-based ESN	Ridge-Regression-based ESN	Linear-Regression-based ML-ESM	Ridge-Regression-based ML-ESM	SVESM
NRMSE	6.6819 x e^{-10}	0.0278	**7.2090 x e^{-13}**	**0.0212**	2.6469
MSE	7.4175 x e^{-23}	1.2865 x e^{-07}	**8.6339 x e^{-29}**	**7.4626 x e^{-08}**	0.00069423

TABLE V: Human Motion Modeling: Testing Dataset NRMSE

Video	Linear-Regression-based ESN	Ridge-Regression-based ESN	Linear-Regression-based ML-ESM	Ridge-Regression-based ML-ESM	SVESM
35_02	0.0743	0.0894	**0.0739**	**0.0892**	0.0873
02_03	0.0322	0.0322	**0.0312**	**0.0321**	0.03432
16_21	0.1538	0.1424	**0.1416**	**0.1536**	0.1534
02_06	0.0370	0.0372	**0.0370**	**0.0372**	0.0387
02_10	0.0360	0.0362	**0.0360**	**0.0362**	0.08345
05_02	0.0521	0.0596	**0.0512**	**0.0595**	0.09321
03_01	0.0910	0.0912	**0.0400**	**0.0401**	0.05345
06_01	0.1159	0.1289	**0.1145**	**0.1287**	0.1843

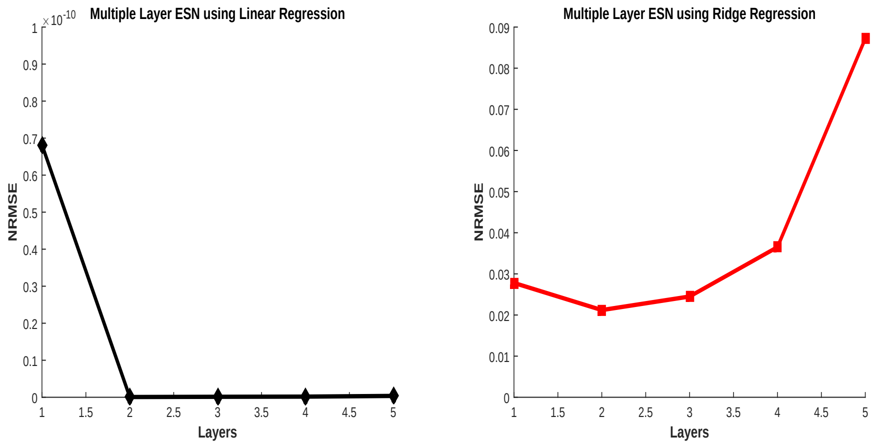


Fig. 5: NARMA Sequence: NRMSE of the evaluated model. LEFT: Comparison of Single Layer ESN with Multiple Layer (2-5) ESN using Linear Regression technique. RIGHT: Comparison of Single Layer ESN with Multiple Layer (2-5) ESN using Ridge Regression technique.

TABLE VI: Specification of 15 Benchmark Datasets

Number	Datasets	# features	# Train	# Test
1	Connect-4	42	50000	17557
2	banknote authentication	5	1000	372
3	EEG	4	10000	200
4	Breast Cancer	32	300	260
5	Hill-Valley	101	606	606
6	segmentation	19	2000	310
7	Page Blocks	10	3000	2473
8	Ecoli	8	200	136
9	Pima Indian Diabetes	8	500	268
10	MNIST Digit	784	60000	10000
11	Yeast	8	1400	484
	Liver Disorders	7	200	145
12	Poker Hand	11	25010	10000
13	Abalone	8	2088	2089
14	SPECT Heart	22	80	187
15	STATLOG (Heart)	13	200	187

The Reuters corpus contains 21578 documents grouped into 135 clusters. It is very unbalanced, with some large clusters more than 300 times larger than some of the smaller ones. We have considered the ModeApte version of this corpus which discards documents with multiple category labels, and only selects the categories with more than 10 documents. This left 8123 documents in a total of 65 categories.

We followed the ModeApte split of training and testing documents which provides 5946 training documents and 2347 testing documents. Overall after preprocessing, this corpus contained 18933 distinct terms. The normalized absolute error obtained on the testing dataset, using the SVESM, the state-of-the-art ESN based techniques and the proposed ML-ESM (using both linear and ridge regression) are shown in Table VIII. Further, since the weight matrices of the ML-ESM are normally initialized randomly, to test their effect on the output we initialized the weight matrices including the input weight matrix, internal weight matrices within each reservoir, and external weight matrices between the reservoirs of each layer with different spectral radii. The spectral radius of the weight matrices codetermines (i) the effective time constant of the ML-ESM (larger spectral radius implies slower decay of impulse response) and (ii) the amount of nonlinear interaction of input components through time (larger spectral radius implies longer range interactions). The gradual effect on NAE due to the change in spectral radius of the weight matrices and the number of reservoir layers is demonstrated in Figure 6.

Figure 6 (a) shows the output of ML-ESM using linear regression and Figure 6 (d) shows the output using the ridge regression technique. Firstly it is observed that while using the linear regression technique, the use of a small number of layers with a gradual increase in spectral radius slightly increases the error. On the other hand, using the ridge regression technique decreases the error with a small number of layers and a gradual increase in the spectral radius. This implies that, particularly

TABLE VII: Performance Comparison of Recurrent Neural Network Based Learning Techniques: Mean: Normalized Absolute Error, STD: Standard Deviation

Datasets	Features	Echo State Network (Linear Regression)		Echo State Network (Ridge Regression)		SVESM		Multiple Layer ESN (Linear Regression)		Multiple Layer ESN (Ridge Regression)	
		MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
Connect-4	43	0.7790	0.0019	0.7552	0.0138	0.8412	0.0013	**0.5916**	**0.0063**	**0.5797**	**0.0222**
banknote authentication	5	0.7293	8.9443 x e^{-05}	0.2664	0.0023	0.7145	0.0054	**0.6845**	**0.0022**	**0.1706**	**3.0822 x e^{-04}**
EEG	4	0.7788	0.0063	0.7816	4.4721 x e^{-05}	0.7315	0.0143	**0.7383**	**2.1679** x e^{-04}	**0.7674**	**1.3416** x e^{-04}
Breast Cancer	32	**0.1732**	**0.0163**	**0.1804**	**1.3416** x e^{-04}	0.9528	0.00034	0.1805	8.9443 x e^{-05}	0.1805	2.8636 x e^{-04}
Hill Valley	101	**0.4141**	**0.0089**	0.1085	0.0045	0.5456	0.0017	0.5554	0.0089	**0.0849**	**0.0044**
segmentation	19	0.5689	0.0295	0.2663	0.0363	**0.0266**	**0.0013**	0.2197	0.0055	0.1353	0.0045
Page Blocks	10	0.1610	0.0084	0.2887	0.0046	0.2643	0.0021	**0.1597**	**0.0035**	**0.2064**	**0.0085**
Ecoli	8	0.3284	0.0084	0.1666	0.00075	**0.0057**	**0.0045**	0.3124	0.00034	0.0942	1.24 x e^{-05}
Pima Indian Diabetes	8	0.7053	0.0024	0.5802	0.00012	0.4652	0.0012	**0.5162**	**0.0023**	**0.4850**	**0.0087**
MNIST Digit	784	0.4557	0.0013	0.4554	0.00062	0.4333	0.0026	**0.4296**	**0.0152**	**0.4132**	**0.0143**
Yeast	8	0.1476	0.0023	0.4942	0.01022	0.4496	0.0014	**0.1366**	**0.0352**	**0.2232**	**0.0324**
Liver Disorders	7	0.6873	0.00015	0.5774	0.01457	0.5658	0.0116	**0.5343**	**0.0642**	**0.5254**	**0.00165**
Abalone	8	0.5787	2.34 x e^{-05}	0.9103	0.001254	0.9874	0.0147	**0.4659**	**0.0173**	**0.9099**	**0.0123**
SPECT Heart	22	0.8015	0.0415	0.7911	0.034221	0.8346	0.0075	**0.7808**	**0.0144**	**0.7826**	**0.0453**
Statlog (Heart)	13	0.5224	0.09415	0.3072	0.00024221	0.3355	0.0758	**0.4836**	**0.0924**	**0.2151**	**0.06433**

on this dataset, longer range interactions between the input components, specifically while using linear regression, do not prove useful in improving the performance of the ML-ESM. Further, using both linear and ridge regression techniques, with an increase in the number of layers and a gradual decrease in the spectral radius of the weight matrices, significantly decreases the error in the beginning, whereas the difference became smaller, eventually reaching a constant.

Secondly, Figures 6 (b) and (e) show the output error of the ML-ESM using linear and ridge regression techniques. This time, the effect on error is observed by varying the number of neurons inside each reservoir and the number of layers of the proposed method. The spectral radius is held fixed (equal to 1.25) throughout the experiment for both the techniques. Firstly, it is observed that for both linear and ridge regression techniques, fewer number of neurons inside each reservoir produce a smaller error using multiple layers of the reservoir inside an ESM. Secondly, the highest variation in error from low to high, is observed using two layers inside the ML-ESM. The variation of error is observed to be much greater in linear regression compared to the ridge regression technique. Lastly, for both the techniques, the higher the number of neurons inside each reservoir, the larger the error produced, with a smaller number of layers leading to a higher error than a larger number of layers.

Thirdly, Figure 6 (c) and (f) shows the output for both ML-ESM, using linear and ridge regression techniques. In these two sub figures, the effect on the error is analyzed by sequentially changing the number of neurons inside each reservoir and the spectral radius of the weight matrices. It can again be clearly seen in these two sub figures, that a higher number of neurons inside each reservoir increases the error compared to when fewer number of neurons are used. The proposed method with ridge regression came out, overall, slightly better compared to the other. Further this implies that multiple layers of the proposed method produce the best approximation with fewer number of neurons inside each reservoir, compared to the standard state-of-the-art which is normally observed to work better with a larger number of neurons.

Finally after comparing both the linear and ridge regression outputs with the proposed method, in all the sub figures of Figure 6; in this particular dataset, the additional smoothing capability and employment of regularization in ridge regression came out as a comparatively better performer in comparison to the linear regression technique.

In accordance with the configurations of Table I, for this dataset, the comparison of the proposed ML-ESM with the state-of-the-art ESN using both linear and ridge regression techniques, and the SVESM method is also demonstrated. Here too, the ML-ESM outperformed the standard ESN by improving the predicted accuracy and reducing the error. SVESM's performance in this particular experiment, as shown in Table VIII, was not bad but it was computationally very expensive in comparison to the other ESN based approaches.

G. MoCap Human Motion Modeling

In this experiment, we trained the evaluated method using four walking sequences, a running sequence, a bending sequence, a washing and dancing sequence respectively, from the Carnegie Melon University (CMU) MoCap dataset [31]. The considered training sequence, corresponding to five different

TABLE VIII: Reuter-21578: Mean: NAE and Standard Deviation (Std): NAE of The Evaluated Models

Error	Linear-Regression-based ESN	Ridge-Regression-based ESN	Linear-Regression-based ML-ESM	Ridge-Regression-based ML-ESM	SVESM
Mean	0.5474	0.5384	**0.5144**	**0.5066**	0.8653
Standard Deviation	0.7254	0.7159	0.7049	0.6928	0.6234

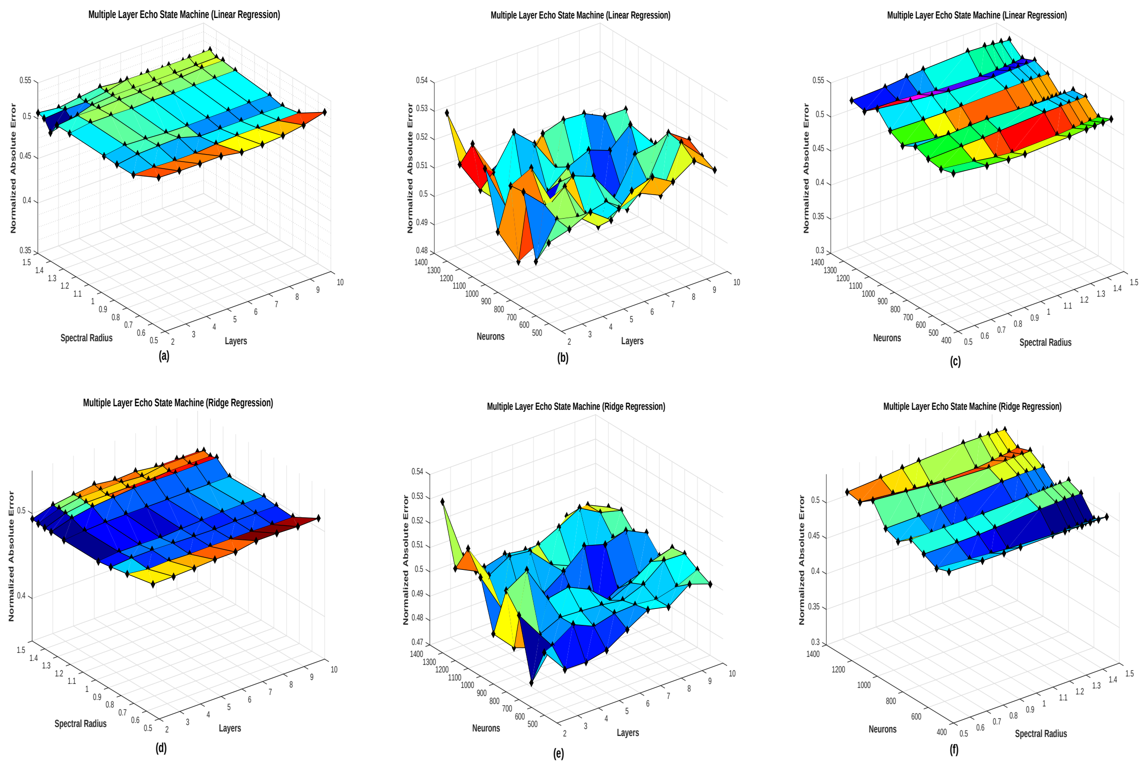


Fig. 6: Top:(Left, Middle, Right): Multiple Layer Echo State Network using Linear Regression, Bottom:(Left, Middle, Right): Multiple Layer Echo State Network using ridge regression

subjects, were obtained from the CMU database files: 35_02, 02_03, 16_21, 02_06, 02_10, 05_02, 03_01 and 06_01. In the sequel, we used fifty percent of each considered video as a training set of the evaluated algorithms, and the remaining fifty percent as the testing set. In Table V, we provide the **NRMSEs** obtained for each considered method. Similarly in Figures 11 and 15 we illustrate the selected frame from the testing dataset of each considered video to show the difference between the actual testing frame and the predicted frame using linear and ridge regression with the proposed ML-ESM. It can be clearly seen from Figures 11 and 15 that the two right columns, which show the predicted output of ML-ESM using both linear and ridge regression technique, predict human motion almost correctly compared with the actual frame with a slight vibration/noise seen on the moving part, and the remaining part very clearly visible. The NRMSE shown in Table V clearly reflect the predicted output of frames visualized in Figure 11 and 15. This enables us to conclude that adding another layer in ML-ESM leads to comparatively better predictions compared to those of the SVESM and the single layer ESN, for the case of both linear and ridge regression techniques. Further we tested the performance of our proposed ML-ESM method using 2-5 layers, and compared results with the single layer ESN, using both linear and ridge regression techniques, as shown in Figure 7. The left subfigure in Figure 7 shows the output using linear regression whereas the right-sub-figure shows the output using the ridge regression technique. As can be seen, there is always a slight decrease in error by adding more than one layer, compared with the single layer ESN, in every video file. The decrease mostly occurs with the addition of 2 or 3 layers, whereas with additional layers, the error is seen to be relatively unchanged. It is also useful to note that that running these experiments took around 45 seconds for the ML-ESM, and over an hour for the case of the SVESM, which further demonstrated the superiority of the ML-ESM for this challenging real-time human motion-based modeling application.

IV. Conclusions

In this paper we proposed a novel ML-ESM model for sequential data. A detailed theoretical analysis has been carried out, and a number of widely used time series benchmarks of different characteristics and origins, have been used to demonstrate:

1) The addition of multiple layers of reservoir provide a more robust alternative to conventional reservoir computing networks.
2) A multiple layers connected cyclic topology is often sufficient for obtaining better performance compara-

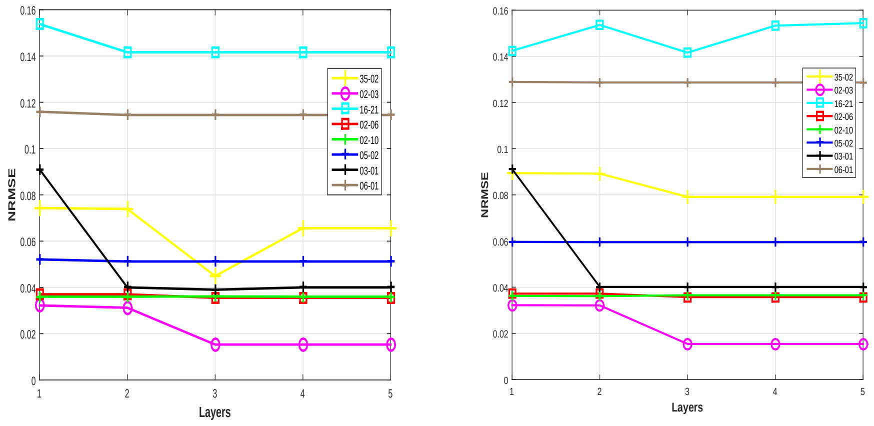


Fig. 7: MoCap Video Dataset: NRMSE of the evaluated model. LEFT: Comparison of Single Layer ESN vs Multiple-Layered (2-5) ESN using Linear Regression technique. RIGHT: Comparison of Single Layer ESN with Multiple-Layered (2-5) ESN using Ridge Regression technique.

ble to those of simple standard cyclic topology.

3) A competitive multiple layer cyclic reservoir network can be constructed in a fully deterministic manner.
4) The state forgetting, state contracting and input forgetting properties are all equivalent to the multiple layer network, having an echo state property in each layer.
5) The null state input sequence along all the layers of the ML-ESM is compatible with the null state sequence.

Additionally, with respect to state-of-the-art single layer ESN methodologies for sequential data modeling, we found that our method is overwhelmingly more accurate in terms of reducing the error and computational competitiveness. Similarly with respect to conventional SVM based ESN methodologies, such as SVESM [10], our proposed method was again found to be computationally more efficient, especially in cases where large data corpora were required to be processed.

A a number of open issues relating to the ML-ESM that we aim to address in the near future include; (1) carrying out a more detailed theoretical and comparative experimental evaluation of the proposed model against other state-of-the-art approaches, using different kinds of reservoir kernels and benchmark datasets; (2) derivation of multi-objective optimization criteria that can help select the appropriate number of layers, number of neurons in each reservoir, learning and adaptation parameters for a particular task; (3) adaptation of the multiple layer reservoir network in an unsupervised fashion for appropriate tasks and evaluating the role of topological organization of reservoirs within the ML-ESM.

V. Acknowledgement

This work of A. Hussain was supported by the U.K. Engineering and Physical Sciences Research Council under Grant EP/I009310/1 and Grant EP/M026981/1. This paper was recommended by Associate Editor G.-B. Huang

References

[1] A. S. Lapedes and R. M. Farber, "How neural nets work," in *Neural information processing systems*, 1988, pp. 442–456.

[2] M. Casdagli, "Nonlinear prediction of chaotic time series," *Physica D: Nonlinear Phenomena*, vol. 35, no. 3, pp. 335–356, 1989.

[3] G.-B. Huang, "An insight into extreme learning machines: random neurons, random features and kernels," *Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.

[4] M. Van Heeswijk, Y. Miche, T. Lindh-Knuutila, P. A. Hilbers, T. Honkela, E. Oja, and A. Lendasse, "Adaptive ensemble models of extreme learning machines for time series prediction," in *Artificial Neural Networks–ICANN 2009*. Springer, 2009, pp. 305–314.

[5] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.

[6] E. A. Wan, "Time series prediction by using a connectionist network with internal delay lines," in *SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS VOLUME-*, vol. 15. ADDISON-WESLEY PUBLISHING CO, 1993, pp. 195–195.

[7] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[8] J. Vesanto, "Using the som and local models in time-series prediction," in *Proc. Workshop on Self-Organizing Maps 1997*. Citeseer, 1997, pp. 209–214.

[9] S. P. Chatzis and Y. Demiris, "Echo state gaussian process," *Neural Networks, IEEE Transactions on*, vol. 22, no. 9, pp. 1435–1445, 2011.

[10] Z. Shi and M. Han, "Support vector echo-state machine for chaotic time-series prediction," *Neural Networks, IEEE Transactions on*, vol. 18, no. 2, pp. 359–372, 2007.

[11] J. Príncipe and J.-M. Kuo, "Dynamic modelling of chaotic time series with neural networks," *Advances in neural information processing systems*, pp. 311–318, 1995.

[12] J. Zhang, K. Tang, and K. Man, "Recurrent nn model for chaotic time series prediction," in *Industrial Electronics, Control and Instrumentation, 1997. IECON 97. 23rd International Conference on*, vol. 3. IEEE, 1997, pp. 1108–1112.

[13] M. Han, J. Xi, S. Xu, and F.-L. Yin, "Prediction of chaotic time series based on the recurrent predictor neural network," *Signal Processing, IEEE Transactions on*, vol. 52, no. 12, pp. 3409–3416, 2004.

[14] A. Rodan and P. Tiňo, "Minimum complexity echo state network," *Neural Networks, IEEE Transactions on*, vol. 22, no. 1, pp. 131–144, 2011.

[15] Z. K. Malik, A. Hussain, and J. Wu, "Novel biologically inspired approaches to extracting online information from temporal data," *Cognitive Computation*, vol. 6, no. 3, pp. 595–607, 2014.

[16] Y. Yang, Q. Wu, Y. Wang, Z. K. Malik, X. LIN, and X. Yuan, "Data partition learning with multiple extreme learning machines," *Cybernetics, IEEE Transactions on*, vol. 45, no. 8, 2014.

[17] Z. K. Malik, A. Hussain, and J. Wu, "An online generalized eigenvalue

version of laplacian eigenmaps for visual big data," *Neurocomputing, doi:10.1016/j.neucom.2014.12.119*, 2015.

[18] Y. Yang, Y. Wang, Q. Wu, X. LIN, and M. Liu, "Progressive learning machine: A new approach for general hybrid system approximation," *Neural Networks and Learning System,IEEE Transactions on*, vol. 26, no. 9, pp. 1855–1874, 2014.

[19] Y. Yang and Q.M.J. Wu, "Multilayer Extreme Learning Machine With Subnetwork Nodes for Representation Learning," *Cybernetics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–14, doi:10.1109/TCYB.2015.2481713, 2015.

[20] H. Jaeger, "The echo state approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, p. 34, 2001.

[21] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[22] D. Verstraeten, B. Schrauwen, M. dHaene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, vol. 20, no. 3, pp. 391–403, 2007.

[23] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[24] M. LukošEvičIus and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[25] M. Minsky and S. Papert, "Perceprrons,"*Oxford, England, M.I.T Press* 1969.

[26] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron,"*IEEE transactions on neural networks and learning systems*,vol. 27, no. 4, pp. 809-821, 2016.

[27] S. S. Malalur, M. T. Manry, and P. Jesudhas, "Multiple optimal learning factors for the multi-layer perceptron," *Neurocomputing*, vol. 149, pp. 1490–1501, 2015.

[28] Z. Zhao, S. Xu, B. H. Kang, M. M. J. Kabir, Y. Liu, and R. Wasinger, "Investigation and improvement of multi-layer perception neural networks for credit scoring," *Expert Systems with Applications*, vol. 42, no. 7, pp. 3508–3516, 2015.

[29] D. Koryakin and M. V. Butz, "Reservoir sizes and feedback weights interact non-linearly in echo state networks," in *Artificial Neural Networks and Machine Learning–ICANN 2012*. Springer, 2012, pp. 499–506.

[30] M. C. Ozturk, D. Xu, and J. C. Príncipe, "Analysis and design of echo state networks," *Neural Computation*, vol. 19, no. 1, pp. 111–138, 2007.

[31] M. Muller, T. Roder, M. Clausen, B. Eberhardt, B. Kruger and A. Weber,"*Documentation mocap database hdm05* [online], available: http://www.ics.uci.edu/ mlearn/mlrepository.html.", *citeseer*, 2007

[32] C. Chang and C. Lin, "Libsvm: A library for support vector machines {Online}," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, pp. 27, 2001.

[33] M. Hnon, "A two-dimensional mapping with a strange attractor," *Comm. Math. Phys.*, vol. 50, no. 1, pp. 69–77, 1976. [Online]. Available: http://projecteuclid.org/euclid.cmp/1103900150

[34] M. Lichman and K. Bache, "Uci machine learning repository, 2013," *URL http://archive. ics. uci. edu/ml, University of California, School of Information and Computer Science*, p. 92, 2013.

[35] Y. LeCun and C. Cortes, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2010.

[36] D. Lewis and others, "Reuters-21578." URL:davidlewis.com/resources /testcollection/reuters21578, *Test Collections*, vol. 1, 1987

[37] D. Cai, X. He, and J. Han, "Locally consistent concept factorization for document clustering," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, no. 6, pp. 902–913, 2011.

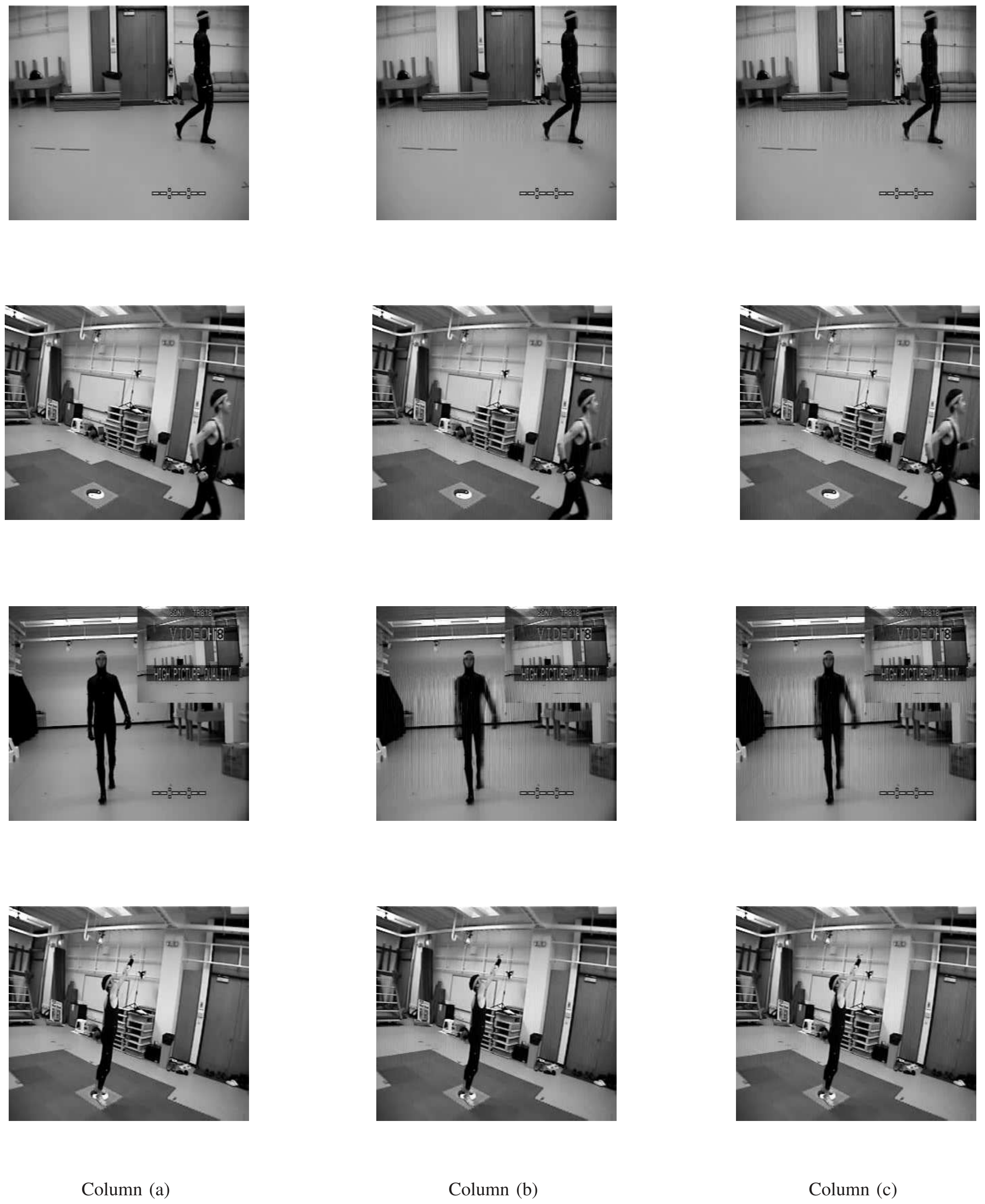
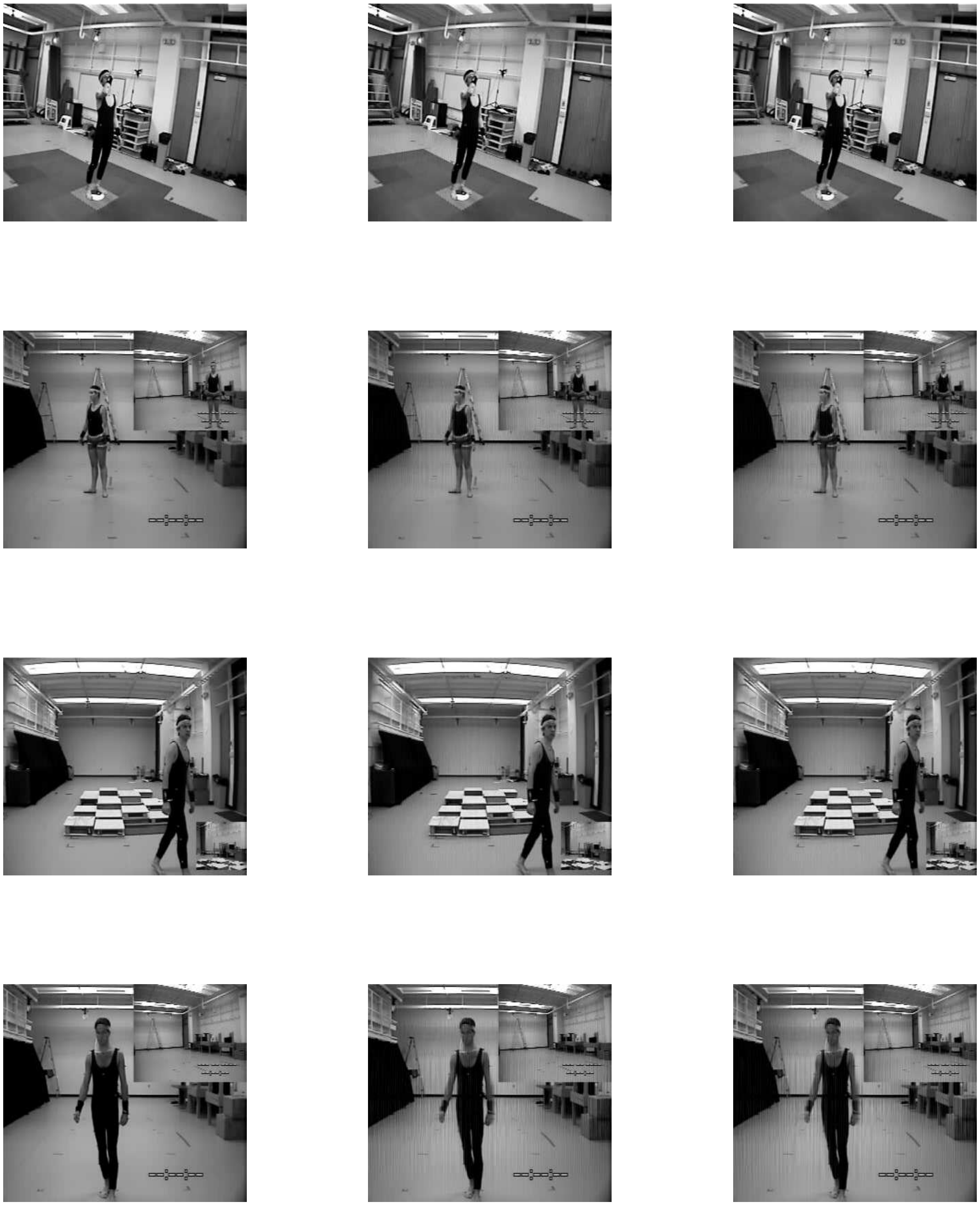


Fig. 11: Column (a) Ground Truth (35_02, 02_03, 16_21, 02_06) Column (b) Predicted Frame (ML-ESM-Linear-Regression). Column (c) Predicted Frame (ML-ESM-Ridge-Regression)



Column (a) Column (b) Column (c)

Fig. 15: Column (a) Ground Truth (02_10, 05_02, 03_01, 06_01) Column (b) Predicted Frame (ML-ESM-Linear-Regression). Column (c) Predicted Frame (ML-ESM-Ridge-Regression)