Multilayer Fuzzy Extreme Learning Machine Applied to Active classification and Transport of objects using an Unmanned Aerial Vehicle

Rolando A. Hernandez-Hernandez[1], Uriel Martinez-Hernandez[2] and Adrian Rubio-Solis[1]

***Abstract*— Based on hierarchical Multilayer Extreme Learning Machine (ML-ELM) and Fuzzy Logic theory (FL), in this paper a Multilayer Fuzzy Extreme Learning Machine (ML-FELM) has been developed with an application to active classification and transport of objects using an indoors Unmanned Aerial Vehicle (UAV). The learning approach that follows the proposed ML-FELM is a forward two-step hierarchical methodology. First, by stacking a number of Fuzzy Autoencoders (FAEs), input data is projected into a feature representation space. Each FAE is functionally equivalent to a Mamdani Fuzzy Logic System of type-1 (T1 FLS). Finally, in the second stage, features achieved by stacking a number of FAEs are classified by using a Fuzzy ELM (FELM) based on T1 FLS theory and ELM. To evaluate the effectiveness of the proposed ML-FELM, a number of other existing machine learning approaches were employed for the active classification and transport of four different geometrical objects. To further ensure the efficiency of the ML-ELM, a number of popular benchmark data sets for classification problems are also suggested. Based on our experimental results and compared to other deep learning strategies, the ML-FELM not only represents a fast machine learning approach, but also produces a high model accuracy for image classification.**

***Index Terms*— Multilayer Learning, Extreme Learning Machine, Fuzzy Logic, Image processing, Neural Networks.**

I. INTRODUCTION

Multilayer Extreme Learning Machine (ML-ELM) is an emerging field in computational intelligence that is gaining more and more in popularity. This is mainly due to its ability to provide a high trade-off between accuracy and model simplicity in comparison to deep learning methods [1–12]. This popularity is also accredited to the ability of ML-ELM to achieve a high feature representation while maintaining a low computational cost for its parameter identification and good generalisation properties. Strictly speaking, compared to deep learning based on Gradient Descent methods, ML-ELM does not require a greedy layer-wise training. The training of ML-ELM usually involves a hierarchical methodology that consist of two different levels where each hidden layer can be considered as an independent module. At first level, an unsupervised multilayer feature encoding is carried out by stacking a number of ELM-based autoencoders, followed by an ELM for supervised feature classification. In other words, ML-ELM decomposes input data representation into multiple layers where outputs of previous layers are used as the input of the current one [1, 13]. That is, the parameters of each hidden layer are randomly generated and need not to be tuned while a feature mapping is generated [13]. As detailed in [14–21], the basic learning block of ELM aims at determining the optimal output weights of Single-Layer-Feedforward-Networks (SLFNs) in which the number of hidden nodes is randomly selected. Thus, the universal approximation capability of ML-ELM has been extended to other Machine Learning (ML) approaches and SLFNs such as kernel learning [5], RBF networks [2], convolutional neural networks [6] and ridge regression [6, 7] with a large number of applications to the solution of real-world problems in the field of regression and classification.

In this paper, a Multilayer Fuzzy Extreme Learning Machine (ML-FELM) that is based on hierarchical ML-ELM and fuzzy logic of type-1 is suggested for active classification and transport of objects using an Unmanned Aerial Vehicle (UAV). The proposed ML-FELM can be viewed as a ML Fuzzy Logic System (ML-FLS) of either Mamdani or Takagi-Sugeno-Kan (TSK). At first level, representation learning is achieved by transforming the input data into a new feature space. This is done, by stacking a number of ELM-based Fuzzy Autoencoders (FAEs), followed by a FELM-based classifier for feature classification. To evaluate the performance of the proposed ML-FELM, two different types of experiments are carried out. First a number of data sets about classification problems is used to measure the efficiency of the ML-FELM with respect to other ML-ELMs. Finally, a second experiment where the ML-FELM is implemented to guide a UAV to recognise and transport a number of four different objects is performed.

This paper is organised as follows: Section II briefly reviews the theory of ELM and ML-ELM as well as Fuzzy ELM. In section II, the proposed ML-FELM is described, while section IV, the methods and robotic platform used for active classification and transport of objects is explained. Section V presents the corresponding results and section VI draws conclusions.

II. BACKGROUND THEORY

A. Extreme Learning Machine (ELM)

According to ELM theory, SLFNs with M hidden nodes whose parameters are randomly selected (including biases)

[1] Rolando A. Hernandez-Hernandez and Adrian Rubio-Solis are with the department of Energy, Center for Engineering and Industrial Development, Qto. Mexico, 76125, Mexico. a.rubiosolis@cidesi.edu.mx, r.hernandez@posgrado.cidesi.edu.mx

[2] Uriel Martinez-Hernandez is with the Department of Electronic and Electrical Engineering, University of Bath, Bath, UK, BA2 7AY, United Kingdom. u.martinez@bath.ac.uk

can approximate (learn) $'P'$ distinct samples $(\mathbf{x}_p, \mathbf{t}_p)$ with zero means $\sum_{p=1}^{M} \parallel \mathbf{y}_p - \mathbf{t}_p \parallel = 0$, in which $\mathbf{x}_p = [x_{p1}, \ldots, x_{pN}]^T \in \mathbf{R}^N$ and $\mathbf{t}_p = [t_{p1}, \ldots, t_{p\tilde{N}}]^T \in \mathbf{R}^{\tilde{N}}$. Thus, a generalised model for SLFNs with M hidden nodes and activation $g_{(x)}$ function can be defined: [14, 15]:

$$\sum_{k=1}^{M} \beta_k g_k(x_p) = \sum_{k=1}^{M} \beta_k g(w_k \cdot x_p + b_k) = \mathbf{y}_p \tag{1}$$

where $\mathbf{w}_k = [w_{k1}, \ldots, w_{kN}]^T$ is the weight vector connecting the kth hidden node and the input nodes, and $\beta_p = [\beta_{p1}, \ldots, \beta_{p\tilde{N}}]^T$ is the weight vector connecting the kth hidden node to the nth output. A compact representation is:

$$\mathbf{H}(w_1, \ldots, w_M, b_1 \ldots, b_M, \mathbf{x}_1, \ldots, \mathbf{x}_P) = \begin{pmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_M \cdot x_1 + b_M) \\ \vdots & \vdots & \vdots \\ g(w_1 \cdot x_P + b_1) & \cdots & g(w_M \cdot x_P + b_M) \end{pmatrix}_{P \times M}$$

$$\beta = (\beta_1, \ldots, \beta_M)_{M \times \tilde{N}}; \quad \mathbf{T} = (\mathbf{t}_1, \ldots, \mathbf{t}_P)_{P \times \tilde{N}} \tag{2}$$

Where $\mathbf{H}$ is the hidden layer output matrix of an SLFN with respect to the inputs $\mathbf{x}_p$. The minimum norm least-squares solution of the linear system $\mathbf{H}\beta = \mathbf{T}$ is unique and can be achieved by calculating the pseudo-inverse $H^\dagger$ as $\hat{\beta} = \mathbf{H}^\dagger\mathbf{T}$. In many real-world applications, $M \ll P$ [15]. Hence, $\mathbf{H}$ is a non-square matrix, such that one specific value for $\hat{\mathbf{w}}_k$, $\hat{\mathbf{b}}_k$ and $\hat{\beta}_k$ need to be found as follows:

$$||\mathbf{H}(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_M, \hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_M)\hat{\beta} - \mathbf{T}|| = \min_{\mathbf{w}_k, \mathbf{b}_k, \beta} ||\mathbf{H}(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_M, \hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_M)\hat{\beta} - \mathbf{T}|| \tag{3}$$

B. *Fuzzy Extreme Learning Machine (FELM)*

As pointed out in [16, 22], a Fuzzy Inference System (FIS) can be interpreted as an SLFN if for a given number of distinct training samples $(\mathbf{x}_p, \mathbf{t}_p)$ ELM can be directly applied. In other words, the parameters of the Membership Functions (MFs, $\mathbf{c}_k$, $\mathbf{a}_k$) are randomly generated, and based on this, ELM is applied to determine the consequent parameters β_k. A model of an FIS with M fuzzy rules is given by:

$$\mathbf{y}_p(\mathbf{x}_p) = \sum_{k=1}^{M} \beta_k G(\mathbf{x}_p, \mathbf{c}_k, a_k) = \mathbf{t}_p, \; p = 1, \ldots, P \tag{4}$$

in which, $\mathbf{x}_p = [x_{p1}, \ldots, x_{pN}] \in \mathbf{R}^N$ and $\mathbf{t}_p = [t_{p1}, \ldots, t_{p\tilde{N}}] \in \mathbf{R}^{\tilde{N}}$. In general, an FIS can be defined by a number of fuzzy rules R^k of the form [23, 24]

$$R^k : \text{IF } x_{p1} \text{ is } A_{1k} \text{ AND } x_{p2} \text{ is } A_{2k} \text{ AND } \ldots \\ \text{IF } x_N \text{ is } A_{Nk} \text{ THEN } (y_1 \text{ is } \beta_{k1}) \ldots \; (y_{\tilde{N}} \text{ is } \beta_{k\tilde{N}}) \tag{5}$$

where, $A_{sk}(s = 1, \ldots, N, k = 1, \ldots, M)$ are the fuzzy sets that correspond to the sth input variable x_{ps} in the kth rule, where $\tilde{N}$ is the dimension of the pth output vector $\mathbf{y}_p = [y_1, \ldots, y_{\tilde{N}}]$. When an FIS employs a TSK inference engine, β_{kl} $(k = 1, \ldots, M, l = 1, \ldots, \tilde{N})$ is a linear combination of input variables, i.e. $\beta_{kl} = q_{kj,0} + q_{kj,0}x_1 + \ldots q_{kj,N}x_N$, otherwise if the FIS is of Mamdani type, β_{kl} is a crisp value. In Fuzzy Logic System theory (FLS), the degree to which any given input x_{ps} satisfies the quantifier A_{sk} is specified by its Membership Function (MF) $\mu_{A_{ks}}(c_{ks}, a_k)$, where usually a non-constant piece-wise continuous MF is used [25]. By using the symbol $\otimes$ for the representation of fuzzy logic AND operations, the firing strength of each kth fuzzy rule is

$$R^k(\mathbf{x}_p; \mathbf{c}_k, a_k) = \mu_{A_{k1}}(x_{p1}, c_{k1}, a_k) \\ \otimes \mu_{A_{k2}}(x_{p2}; c_{k2}, a_k) \otimes \ldots \otimes \mu_{A_{kN}}(x_{pN}; c_{kN}, a_k) \tag{6}$$

Each fuzzy rule R^K can be normalised as

$$G(\mathbf{x}_p; c_{ks}, c_k) = R^k(\mathbf{x}_p; c_k, a_k) \Bigg/ \sum_{k=1}^{M} R^k(\mathbf{x}_p; c_k, a_k) \tag{7}$$

Similar to [16], G is called fuzzy basis function. Thus, for each pth input-output, Eq. (4) can be defined as

$$\mathbf{y}_p = \beta_k R^i(\mathbf{x}_p; \mathbf{c}_k, a_k) \Bigg/ \sum_{k=1}^{M} \beta_k R^i(\mathbf{x}_p; \mathbf{c}_k, a_k) \tag{8}$$

Consequent parameters are the linear combination of inputs $\beta_k = \mathbf{x}_{p,e}^T \mathbf{q}_k$, while for a Mamdani fuzzy model, $\mathbf{x}_{p,e} = 1$, and $\mathbf{q}_k = \beta_k = [\beta_{k1}, \ldots, \beta_{k\tilde{N}}]^T$, where $\mathbf{q}_k$ is a weight vector of crips values. For a TSK fuzzy model $\mathbf{x}_{p,e} = [1 \; \mathbf{x}_p^T]^T$ is the extended version of $\mathbf{x}_p$.

$$\mathbf{q}_k = \begin{pmatrix} q_{k1,0} & \cdots & q_{k\tilde{N},0} \\ \vdots & & \vdots \\ q_{k1,N} & \cdots & q_{k\tilde{N},N} \end{pmatrix}_{(N+1) \times \tilde{N}} \tag{9}$$

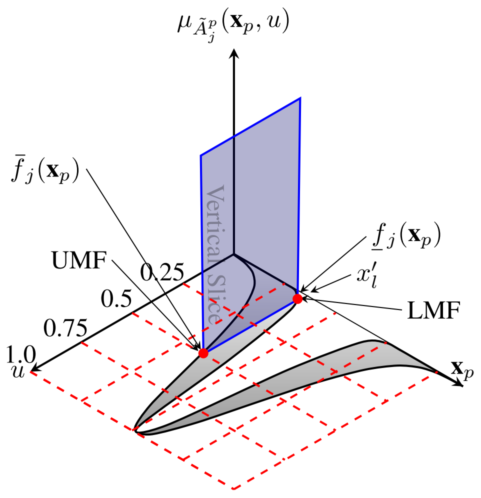
Therefore, Eq. (4) becomes

$$\mathbf{y}_p(\mathbf{x}_p) = \sum_{k=1}^{M} \mathbf{x}_{p,e}^T \mathbf{q}_k G(\mathbf{x}_p, \mathbf{c}_k, a_k) \tag{10}$$

Where $\mathbf{HQ} = \mathbf{T}$ is a compact representation of Eq. (10), in which, $\mathbf{Q} = (\mathbf{q}_1, \ldots, \mathbf{q}_M)^T$ is the matrix of coefficients $q_{kj,s}$.

C. *Interval Type-2 Fuzzy Extreme Learning Machine for Classification (IT2-FELM)*

Most of the proposed neural fuzzy network systems of the interval type-2 and based on ELM theory have been particularly applied to solve regression problems [26]. Such systems usually employed Multiple-Input-Single-Output (MISO) neural structures with a Karnik-Mendel type-reduction layer [24, 27]. In [26], A Multi-Input-Multi-Output (MIMO) fuzzy network based on the functional equivalence between the Radial Basis Function Neural Network (RBFNN) and IT2 FL was suggested to the solution of classification problems [20].



$\mu_{\tilde{A}^p_j}(\mathbf{x}_p, u)$
$\bar{f}_j(\mathbf{x}_p)$
UMF
0.25
0.5
0.75
1.0
u
Vertical Slice
$\underline{f}_j(\mathbf{x}_p)$
x'_l
LMF
$\mathbf{x}_p$

Fig. 1: Singleton fuzzification and interval secondary MF activated when $\mathbf{x}_p = x'_l$ for the jth receptive unit of the IT2-RBFNN (Taken from [24]).

According to [26], given an input vector $\mathbf{x}_p = [x_1, \ldots, x_N]^T$, each corresponding fuzzy rule R^j in an MIMO IT2 is described by a multi-variable Gaussian MF $\mu_{R^i}(\mathbf{x}_p, y_p^s) = \mu_{R^j}[x_1, \ldots, x_n, y_p^s]$, $s = 1, \ldots, \tilde{N}$, where the input vector $\mathbf{x}_p \in X_1 \times \ldots X_N$ and the implication engine is:

$$\mu_{R^j}(\mathbf{x}_p, y_p^s) = \mu_{A^j \to G^j} = \left[T_{k_1}^N \mu_{F_k^j}(x_k) \star \mu_{G^j(y)} \right] \quad (11)$$

$$= [\underline{f}_j(\vec{x}_p), \bar{f}_j(\vec{x}_p)] \quad (12)$$

where $\star$ is the minimum $t - norm$ that represents the shortest Euclidean distance to the input vector $\mathbf{x}_p$. $F^j := [\underline{f}_j(\vec{x}_p), \bar{f}_j(\vec{x}_p)]$ is the lower and upper membership function (LMF, UMF) respectively. Each MF in the MIMO IT2-RBFNN may be interval Gaussian MF with an uncertain width $\sigma_j = [\sigma_j^1, \sigma_j^2]$ and fixed center (mean) μ_{kj} as shown in Fig. 1:

$$F^j := \begin{cases} \underline{f}_j(\mathbf{x}_p) = \exp\left[-\sum_{k=1}^{N} \left(\frac{x_k - \mu_{kj}}{2\sigma_j^2} \right)^2 \right] \\ \bar{f}_j(\mathbf{x}_p) = \exp\left[-\sum_{k=1}^{N} \left(\frac{x_k - \mu_{kj}}{2\sigma_j^1} \right)^2 \right] \end{cases} \quad (13)$$

In this paper, an IT2-RBFNN of Mamdani having a product inference rule and a singleton output space is implemented. To reduce the associated computational load that usually implies the implementation of karnik-Mendel algorithms, a close-form Nie-Tan method is implemented as a direct defuzzification process that employs the vertical representation of the Footprint Of Uncertainty (FOU). As illustrated in Fig. 2, an IT2-RBFNN using a Nie-Tan direct-defuzzification as the output layer can be viewed as an Interval Type-2 Fuzzy Extreme Learning Machine (IT2-FELM) that does not require a sorting process. The application of a Nie-Tan layer represents a zero Taylor series approximation of Karnik-Mendel+defuzzification method [22]. Moreover, a Nie-Tan operator is equivalent to an exhaustive and accurate type-reduction for both discrete and continuous IT2 Fuzzy Sets (FSs) [22].

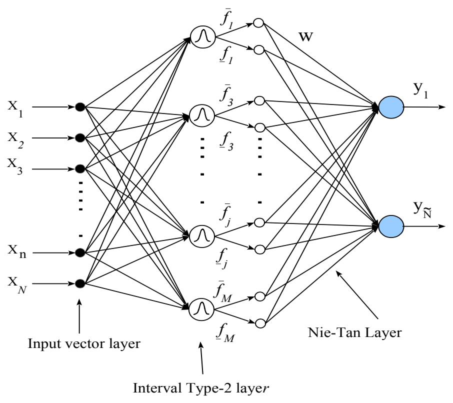


Fig. 2: Interval Type-2 Fuzzy Extreme Learning Machine using a NieTan algorithm for classification problems [28].

The output of a IT2-FELM with a NT layer having an uncertain $[\sigma_{1j}, \sigma_{2j}]$ and fixed center is formulated as follows:

$$\mathbf{T} = \mathbf{H}_{NT}\mathbf{W} \quad (14)$$

where $\mathbf{H}_{NT}$ is the matrix for IT2 firing strengths, and β is the weight vector in the output layer. Each input in $\mathbf{H}_{NT}$ is defined by the IT2 MF $\varphi(\mu, \sigma_{1j}, \sigma_{2j}, \mathbf{x}_p) = \underline{f}_j + \bar{f}_j / \sum_{j=1}^{M} \underline{f}_j + \sum_{j=1}^{M} \bar{f}_j$. where $\mu_j = (\mu_{1j}, \ldots, \mu_{Nj})$ and $\mathbf{W}$ and $\mathbf{T}$ are

$$\mathbf{W} = (w_1 \ldots w_M)_{M \times \tilde{N}} \quad \text{and} \quad \mathbf{T} = (\mathbf{t}_1 \ldots \mathbf{t}_P)_{P \times \tilde{N}} \quad (15)$$

D. Multilayer Extreme Learning Machine (ML-ELM)

Multilayer Extreme Learning Machine (ML-ELM) was initially suggested in [1] as an alternative the reduce the computational load that frequently results from the iterative nature of Back Propagation (BP) learning algorithms that are commonly employed to train Multilayer Neural Networks (ML-NNs) [**?**, 2, 3]. The main advantage of an ML-ELM is the integration of a single learning mechanism that involves several layers for representational learning, followed by a final layer of ELM classification [1]. The basic building block ML-ELM is an ELM-based Autoencoder (ELM-AE, See Fig. 8) that is stacked to build a multilayer structure (deep structure) while performing layer by layer unsupervised learning for feature representation where fine iterative tuning is not required [29].

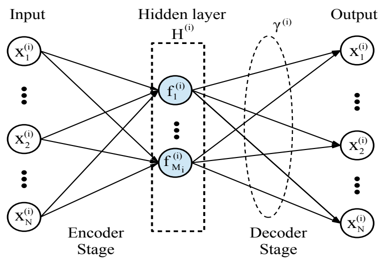


Fig. 3: Architecture of the ith ELM-AE used as the basic building block of a ML-ELM (Taken from [13]).

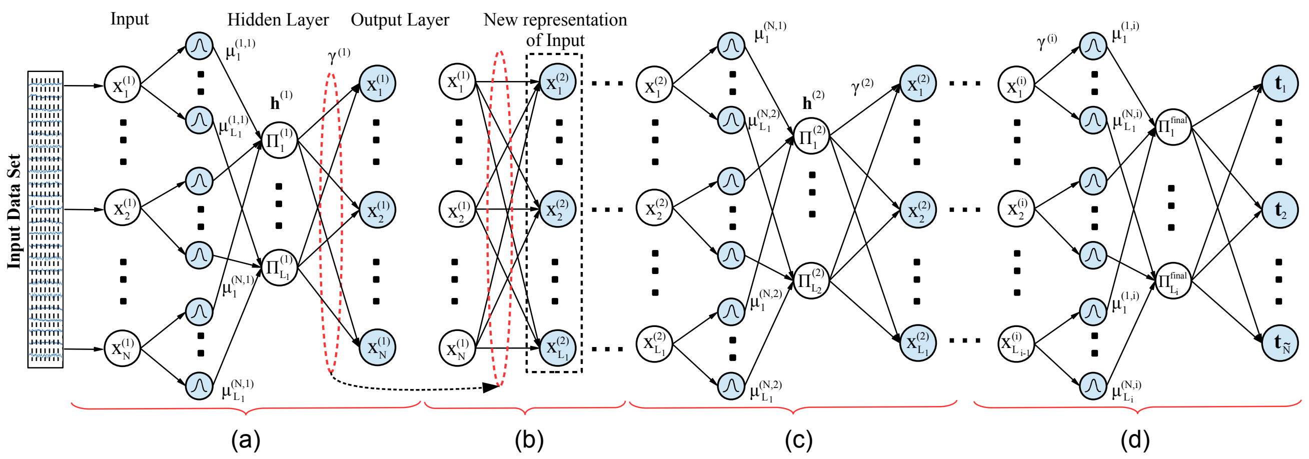


Fig. 4: Architecture of the ML-ELM. (a) ELM-AE outputs the transformation $\Gamma^{(1)}$ for representation learning, (b) where a new representation $\mathbf{x}^{(2)}$ is learned by computing the activation function $g(\mathbf{X}^{(1)}(\Gamma)^{(1)})$, (c) $\mathbf{x}_2$ is used as input to ELM-AE for a second representation learning, and (d) representation learning, the final $\mathbf{x}^{final}$ is used to calculate the output weight β for classification, where $\tilde{N}$ is the max number of target classes.

An ML-ELM is a neural structure that consists of a number of L hidden layers, where for a given input $\mathbf{X}^{(i)} = [\mathbf{x}_1^{(i)}, \dots, \mathbf{x}_N^{(i)}]$, $k = 1, \dots, N$, the ith data transformation $\Gamma^{(i)} = [\gamma_1^{(i)}, \dots, \gamma_N^{(i)}]$ is computed using Eq. (16), and $\gamma^{(i)}$ is the transformation vector that corresponds to the input vector $\mathbf{x}_k^{(i)}$ [5, 13].

$$\mathbf{H}^{(i)}\Gamma^{(i)} = \mathbf{X}^{(i)} \tag{16}$$

in which, $\mathbf{H}^{(i)}$ is the output matrix of the ith hidden layer w.r.t the input $\mathbf{X}^{(i)}$. Data transformation is achieved by projecting $\mathbf{X}^{(i)}$ along the decoder stage weights at each ELM-AE. That means, at the encoder stage, each ELM-AE generates a number of orthogonal random parameters, e.g. random input weights and random biases in hidden nodes for additive nodes [21, 29]. Thus, orthogonal random hidden parameters of each ELM-AE are computed using Eq. (17).

$$\mathbf{h}(\mathbf{x}_k) = g(\mathbf{x}_k\mathbf{A} + \mathbf{b}) = [h_1(\mathbf{x}_k), \dots, h_{M_i}(\mathbf{x}_k)] \tag{17}$$

in which, $\mathbf{H}^{(i)} = [\mathbf{h}(\mathbf{x}_1), \dots, \mathbf{h}(\mathbf{x}_N)]^T$, $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ and $\mathbf{b}^T\mathbf{b} = 1$. Hence, the ith transformation term $\Gamma^{(i)}$ is calculated as:

$$\Gamma^{(i)} = (\mathbf{H}^{(i)})^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}^{(i)}(\mathbf{H}^{(i)})^T\right)^{-1} \mathbf{X}^{(i)} \tag{18}$$

Final representation $\mathbf{X}^{\text{final}}$ is defined as

$$\mathbf{X}^{\text{final}} = g(\mathbf{X}^{(i)}(\Gamma^{(i)})) \tag{19}$$

If layer $M_i = M_{i+1}$, then g can be chosen as a linear function, otherwise g is chosen as nonlinear piecewise function (RBFs or sigmoids). In [13], $\mathbf{X}^{\text{final}}$ was used as the hidden layer output to compute the output weight vector β as [1, 13].

$$\mathbf{X}^{\text{final}}\beta = \mathrm{T} \tag{20}$$

by adding a regularisation factor C and using the pseudoinverse of the final transformation, $\mathbf{X}^{\text{final}}$ as the firing strength matrix, the term β is computed as shown in Eq. (21).

$$\beta = \mathbf{X}^{\text{final}} \left(\frac{1}{C} + \mathbf{X}(\mathbf{X}^{\text{final}})\right)^{-1} \mathbf{T} \tag{21}$$

Unlike the hierarchical ML ELM [1], the methodology reviewed in this section directly uses the final data representation $\mathbf{X}^{\text{final}}$ as hidden layer to calculate the weight vector β.

III. Proposed Multilayer Fuzzy Extreme Learning Machine (ML-FELM)

Similar to ML-ELM, the proposed ML-FELM is based on a hierarchical learning scheme that involves two independent learning mechanisms. First, a number of ELM-based fuzzy autoencoders (FAE for short) is applied to extract a high level of features, followed by an independent ELM-based classifier/regressor that uses the final data transformation Γ^M as its input vector. The basic building block of the ML-FELM is based on the FELM described in Fig. 2(a) and suggested in [30]. According to Fig. 1(a), given a number of data samples $(\mathbf{x}_p, \mathbf{t}_p)$, where $\mathbf{x}_p = [x_{p1}, \dots, \mathbf{x}_{pN}]$, $k = 1, \dots N$, the MF of each jth fuzzy rule is calculated by the sum-product composition which is given by:

$$R^j(\mathbf{x}_p; \mathbf{c}_j, \sigma_j) = \prod_{j=1}^{Li} \mu_j^{k,i}(x_{pk}); \; p = 1, \dots, P \tag{22}$$

where L_i is the number of fuzzy rules, $\mathbf{c}_j = [c_{j1}, \dots, c_{jN}]$ and $\sigma_j = [\sigma_{j1}, \dots, \sigma_{jN}]$. In this paper, a Mamdani inference implication is employed. That means, the output weights used at each ith data transformation are single crisp values. Each MF is a Gaussian function computed as follows:

$$\mu_j^{k,i}(x_k) = exp\left(-\frac{(x_{pk} - c_{jk})^2}{\sigma_{jk}^2}\right) \tag{23}$$

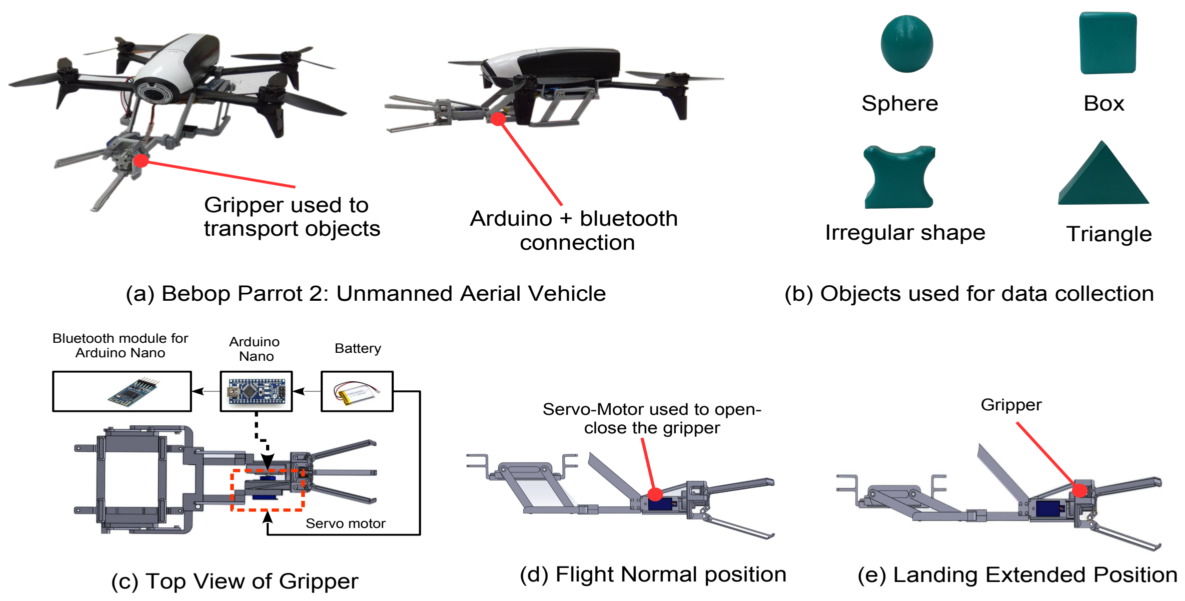

Gripper used to transport objects
Arduino + bluetooth connection
(a) Bebop Parrot 2: Unmanned Aerial Vehicle
Sphere
Box
Irregular shape
Triangle
(b) Objects used for data collection
Bluetooth module for Arduino Nano
Arduino Nano
Battery
Servo motor
(c) Top View of Gripper
Servo-Motor used to open-close the gripper
(d) Flight Normal position
Gripper
(e) Landing Extended Position

Fig. 5: (a) UAV Bebop2 with a robotic arm and a gripper, (b) Geometrical objects used to run all the experiments, (c) Top view of the robotic arm and gripper and (d,e) site views of two different positions of the robotic arm and gripper.

where the output of each FAE is a weighted average:

$$y_{pk}(\mathbf{x}_p) = \frac{\sum_{j=1}^{L_i} R^j(\mathbf{x}_p; \mathbf{c}_j, \sigma_j)\gamma_{jp}^{(i)}}{\sum_{j=1}^{L_i} R^j(\mathbf{x}_p; \mathbf{c}_j, \sigma_j)} \tag{24}$$

in which, $\gamma_p^{(i)} = [\gamma_{j1}^{(i)}, \ldots, \gamma_{jP}^{(i)}]$. Thus, the ith data representation matrix $\Gamma^{(i)} = [\gamma_1^{(i)}, \ldots, \gamma_P^{(i)}]$ for the input $\mathbf{x}_p$ is:

$$\mathbf{H}^{(i)}\Gamma^{(i)} = \mathbf{X}^{(i)} \tag{25}$$

where each $\mathbf{H}^{(i)}$ is defined as:

$$\mathbf{H}(\mu_1^{k,i}, \ldots, \mu_{L_i}^{k,i}, \sigma_1 \ldots, \sigma_{L_i}, \mathbf{x}_1, \ldots, \mathbf{x}_P) \\ = \begin{pmatrix} h_{11} & \cdots & h_{1L_i} \\ \vdots & \vdots & \vdots \\ h_{P1} & \cdots & h_{PL_i} \end{pmatrix}_{P \times L_i}$$

each $h_{pi} = R^j(\mathbf{x}_p; \mathbf{c}_j, \sigma_j) / \sum_{j=1}^{L_i} R^j(\mathbf{x}_p; \mathbf{c}_j, \sigma_j)$. Therefore, $\Gamma^{(i)}$ is calculated as follows:

$$\Gamma^{(i)} = (\mathbf{H}^{(i)})^T \left(\frac{\mathbf{I}}{C_{\text{final}}} + \mathbf{H}^{(i)}(\mathbf{H}^{(i)})^T \right)^{-1} \mathbf{X}^{(i)} \tag{26}$$

Similar to ML-ELM, the final representation is given by $\mathbf{X}^{\text{final}} = g(\mathbf{X}^{(i)}(\Gamma^{(i)}))$. if $L_i = L_{i+1}$ for all i, the activation function g is chosen as a linear piecewise function, otherwise as a nonlinear piecewise function. Finally, $\mathbf{X}^{\text{final}}$ is fed into an FELM classifier to calculate the output weight vector β.

$$\mathbf{H}^{\text{final}}\beta = \mathrm{T} \tag{27}$$

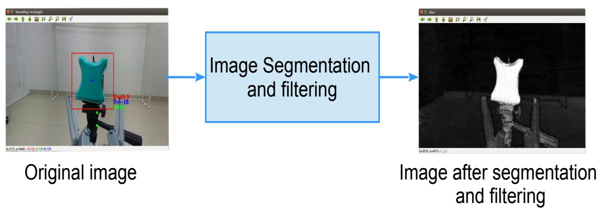

Original image
Image Segmentation and filtering
Image after segmentation and filtering

Fig. 6: Image preprocessing.

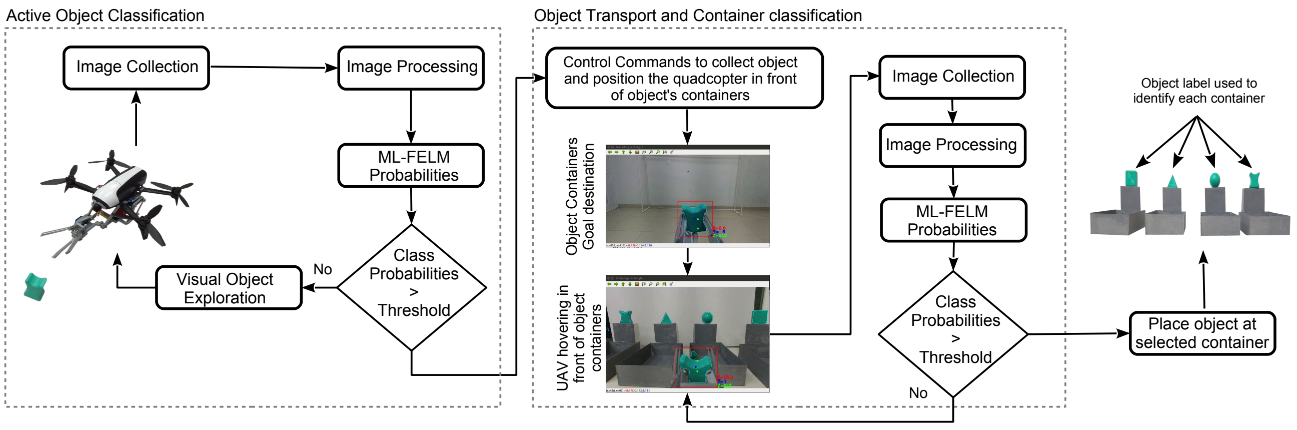
and β is calculated by Eq. (28)

$$\beta = (\mathbf{H}^{\text{final}})^{\dagger}\mathbf{T} = \mathbf{H}^{\text{final}} \left(\frac{1}{C_{\text{final}}} + \mathbf{H}^{\text{final}}(\mathbf{H}^{\text{final}})^T \right)^{-1} \mathbf{T} \tag{28}$$

IV. Methods

A. *Robotic Platform*

In this work, a UAV Bebop2 was used to collect images and run all the experiments (See Fig. 5(a)). The UAV transmits the live video stream of its front facing camera and pose information to a central computer over WiFi. As shown in Fig. 5(a-d), a robotic arm was also constructed to collect and transport four different objects. The output of the proposed ML-FELM is used to control the position of the UAV and the robotic arm. The UAV transmits images to a central computer where active object classification and the associated methodology used to plan the UAV trajectory to transport each object are carried out.



Active Object Classification
Image Collection
Image Processing
ML-FELM Probabilities
Class Probabilities > Threshold
No
Visual Object Exploration
Object Transport and Container classification
Control Commands to collect object and position the quadcopter in front of object's containers
Object Containers Goal destination
UAV hovering in front of object containers
Image Collection
Image Processing
ML-FELM Probabilities
Class Probabilities > Threshold
No
Place object at selected container
Object label used to identify each container

Fig. 7: Architecture for active UAV control that consists of active object classification and object transport.

As illustrated in Fig. 5, in order to control the robotic arm, a microcontroller Arduino one connected through Bluetooth and was employed to open-close the final gripper of the robotic arm. To implement the ML-FELM, control commands to guide the position of the UAV, the processing of each image and the communication between the central computer and the robotic arm, the Robotic Operating System (ROS) middleware using Ubuntu was used.

B. Data Collection and Preprocessing

To train the proposed ML-FELM, different data sets were collected using the on board camera of the UAV Bebop2 (Fig. 5(a)) at different angles and positions. A final data set of 36386 images were collected with a resolution of 856×480 pixels corresponding to four different objects, i.e. box (9114 images), circle (9042 images), an irregular shape (9050 images) and a triangle (9180 images) as shown in Fig. 5(b). As illustrated in Fig. 6, each image collected by the onboard camera was preprocessed to remove out the background. This process was carried out using two OPEN-CV built-in functions. First an HSV filter to specifically select a color was applied to segment each object. Secondly, a blurring kernel filter is implemented to obtain the final image to be classified.

C. Methodology for Active Classification and Transport of Objects using the ML-FELM

As described in Fig. 7, the classification and transport of each object is divided into two main stages. First, the UAV hovers in front of a random object while actively collecting and processing a predefined number of images (See Fig. 6). Each preprocessed image is fed into the ML-FELM, and its classification outcome is stored. The probability of each classification is obtained as the number of times the output of the ML-FELM ($Y_{ML-FELM}$) is an object o_i divided by the total number of images used to identify an object .

$$P(c|o_j) = \frac{number\ of\ times\ Y_{ML-FELM}\ is\ o_i}{number\ of\ collected\ images} \tag{29}$$

where $j = 1, \ldots, C$, such as C is the number of object classes, and c the current class. Thus, the classification of the current object is obtained with the current maximum a posterior (MAP) estimate as:

$$\hat{c} = \arg\max_{c} P(c|o_i) \tag{30}$$

If the value of $\hat{c}$ is higher than a predefined threshold, then this information is used to guide the drone to identify the container that corresponds to the classified object. On the contrary, this process is repeated until the value of $\hat{c}$ is satisfied. In order to determine which is the correct container, four objects with the same shape that those objects that are recognised and transported by the UAV are place at the top of each container (See Fig. 7). In other words, the output of the ML-FELM is actively used to control not only the current position of the UAV, but also the gripper opening and closing. In a like-manner to the first stage, a number of predefined images are collected from the label of each container, and then preprocessed to feed the ML-FELM. A new value of $\hat{c}$ is then computed and used to guide the UAV to the goal destination, where the robotic arm drops the selected object.

V. Results

In order to evaluate the accuracy of the proposed ML-FELM, in this section two different experiments are suggested. First a number of four popular benchmark data sets about classification problems are suggested. Secondly, the experimental results for active classification and transport of objects using a UAV are presented and compared to other ML approaches.

A. Benchmark data sets for classification Problems

As described in Table I, in this section four classification problems are used to evaluate the ML-FELM. For the cross-validation purposes, ten random experiments are conducted, where for training and testing two subsets are created correspondingly as indicated in Table I.

TABLE I: SPECIFICATION OF BENCHMARK CLASSIFICATION PROBLEMS.

Datasets	**# Attributes**	**# Classes**	**# Observations**	
			Training	Testing
Australian	14	2	345	345
Samitage	36	6	4,400	2,035
Abalone	8	2	2,000	2,177
Letter Recognition	16	26	10,000	10,000

TABLE II: PERFORMANCE COMPARISON FOR REAL-WORLD BENCHMARK CLASSIFICATION PROBLEMS.

Data Sets	Models	Training (%)		Time	Testing (%)		# hidden units
		Mean	SD	Sec	Mean	SD	
Australian	ML-FELM	83.48	0.283	3.07	87.94	0.002	$[10, 10, 50]$
	ML-ELM	84.41	0.201	2.97	87.82	0.002	$[10, 10, 50]$
	ML-KELM	86.88	0.303	6.21	87.12	0.001	345
	FELM	85.22	0.120	2.11	86.08	0.094	540
Satimage	ML-FELM	91.04	0.243	28.3	90.19	0.033	$[40,\ 850]$
	ML-ELM	90.89	0.009	15.0	91.90	0.019	$[50,\ 800]$
	ML-KELM	92.04	0.045	29.2	93.28	19.81	4400
	FELM	93.19	0.012	15.2	89.41	0.01	500
Abalone	ML-FELM	58.99	0.23	4.12	57.20	0.09	$[70, 70, 150]$
	ML-ELM	59.10	0.25	1.19	56.79	0.12	$[70, 70, 150]$
	ML-KELM	58.80	0.54	2.54	57.62	0.05	2,000
	FELM	58.11	0.67	0.01	55.86	0.61	25
Letter Recognition	ML-FELM	97.89	0.230	409	93.12	0.24	$[210, 210, 1900]$
	ML-ELM	96.01	0.190	62.3	93.42	0.66	$[210, 210, 1900]$
	ML-KELM	95.01	0.290	101	94.49	0.19	10,000
	FELM	95.81	0.01	19.2	92.88	0.09	2,000

The average classification accuracy of ML-FELM, ML-ELM, ML-KELM and FELM is compared in Table I as well as the number of fuzzy rules (hidden units for the case of fuzzy approaches) per each problem. For all ML structures, a two-layer feature extraction is applied, followed by a classification FELM as shown in the last column of Table I. The number of outputs for each FELM is equal to the number of classes of each data set. From our experiments, it was found the optimum value for C_i for the Australian, Satimage, Abalone and Letter Recognition data are $[1^{-2}, 10^4, 10^8]$, $[1^{-2}, 10^3, 950]$, $[0.9, 2.4, 10^3]$ and $[10^{-3}, 10^2, 9.8]$ respectively. As can be noted from Table I, although the associated computational load of ML-FELM is higher for the treatment of large data sets, its model accuracy is comparable and in some cases higher than that obtained by a ML-ELM and ML-KELM. Especially ML-FELM outperforms its single-hidden-layer counterpart the FELM.

TABLE III: PERFORMANCE COMPARISON FOR OBJECT CLASSIFICATION.

Models	Training		Testing	# hidden units
	Mean (%)	Time	Mean (%)	
ML-FELM	97.47	78.4s	97.30	$[1740, 1600, 1600]$
ML-ELM	95.32	45.1s	94.16	$[1740, 1600, 1600]$
ML-FELM-II	98.22	134s	97.80	$[1740, 1600, 1600]$
ML-KELM	94.51	62.2s	93.50	$[25470, 25470, 25470]$
CNN	99.97	2200s	99.12	-

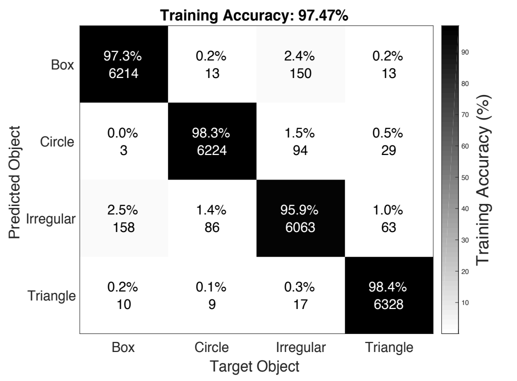
B. Active Classification and transport of objects

This section is dedicated to describe the experimental setup and results obtained for the classification and transport of four different objects using the UAV and ML-FELM. For cross-validation purposes, the object data set was split into two subsets, i.e. 70% for training and 30% for testing. In order to compare the accuracy of the ML-FELM, a ML-ELM, ML-KELM, and a Convolutional Neural Network (CNN) with a structure that consists of convolution($48 \times 48 \times 32$)-pooling($24 \times 24 \times 32$)-convolution($22 \times 22 \times 32$)-convolution($20 \times 20 \times 64$)-pooling($10 \times 10 \times 64$)-classifier($64000 - 500 - 4$). A third ML structure that consists of two FAEs + IT2-FELM which is used as a feature representation classifier is also implemented as a comparison method and that is called ML-FELM-II. It was found that a value for $C_i = [1^3, 1^7, 1^{49}]$ and $C_i = [1^2, 1^{14}, 1^{40}]$ provides the highest trade-off between model simplicity and testing accuracy for the ML-FELM and ML-ELM respectively. As described in Table II. The optimal configuration for each ML-ELM approach follows the arrangement [AE1/FAE1,AE2/FAE2, ELM/FELM classifier] where each input represents the number of hidden units used by each model. As can be noted from Table III, although the best accuracy is achieved by the CNN, this is compensated by the learning time and model simplicity of the ML-FELM which is much smaller. According to the experiment results, adding an IT2-FELM enhances he model accuracy of a ML structure that includes two FAEs (or ML-FELM-II) as shown in the results presented in TABLE III.

VI. Conclusions and Future Work

In this paper, a ML-FELM that is based on the concept of Multilayer ELM and T1 FLS theory is suggested for active object classification and their transport. The learning approach of the ML-FELM involves two main steps, first a number of Fuzzy Autoencoders are stacked in order to achieve a high feature representation of the input data. Secondly, a FELM of Mamdani type is used for feature classification. In order to evaluate the efficiency of the proposed ML-FELM, two different experiments were suggested. First, a number of popular data sets about classification problems are used to compare the performance of the ML-FELM with respect to other existing ML-ELM methods. Secondly, an image data set is collected to train the ML-FELM whose optimised model is used in real time experimets to classify and transport a number of four different objects. According to our experiments, the ML-FELM is a fuzzy ML technique that provides a similar trade-off between model simplicity and model accuracy for solving classification problems and feature representation.

Future work involves the development of new online learning techniques for ML-FELM methods, as well as the application of evolutionary computation. This also includes the development of new Fuzzy Autoencoders able to provide a high level of feature representations in the field of image processing and classification.



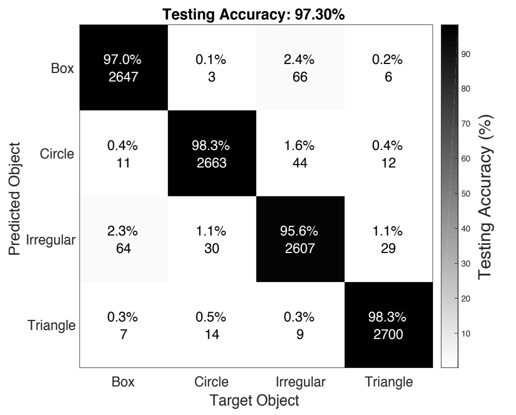


Fig. 8: Confusion matrix for the average training and testing accuracy.

References

[1] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong, "Representational learning with extreme learning machine for big data," *IEEE intelligent systems*, vol. 28, no. 6, pp. 31–34, 2013.

[2] P. Vidnerová and R. Neruda, "Deep networks with rbf layers to prevent adversarial examples," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2018, pp. 257–266.

[3] C.-M. Vong, C. Chen, and P.-K. Wong, "Empirical kernel map-based multilayer extreme learning machines for representation learning," *Neurocomputing*, 2018.

[4] M. Chen, Y. Li, X. Luo, W. Wang, L. Wang, and W. Zhao, "A novel human activity recognition scheme for smart health using multilayer extreme learning machine," *IEEE Internet of Things Journal*, 2018.

[5] C. M. Wong, C. M. Vong, P. K. Wong, and J. Cao, "Kernel-based multilayer extreme learning machines for representation learning," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 3, pp. 757–762, 2018.

[6] C. Gautam, A. Tiwari, S. Suresh, and A. Iosifidis, "Multi-layer kernel ridge regression for one-class classification," *arXiv preprint arXiv:1805.07808*, 2018.

[7] J. Kim, J. Kim, G.-J. Jang, and M. Lee, "Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection," *Neural Networks*, vol. 87, pp. 109–121, 2017.

[8] K. Han, D. Yu, and I. Tashev, "Speech emotion recognition using deep neural network and extreme learning machine," in *Fifteenth annual conference of the international speech communication association*, 2014.

[9] A. Rubio-Solis and G. Panoutsos, "Iterative information granulation for novelty detection in complex datasets," in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2016, pp. 953–960.

[10] M. Yousefi-Azar and M. D. McDonnell, "Semi-supervised convolutional extreme learning machine," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 1968–1974.

[11] G. Altan and Y. Kutlu, "Hessenberg elm autoencoder kernel for deep learning," *Journal of Engineering Technology and Applied Sciences*, vol. 3, no. 2, pp. 141–151, 2018.

[12] K. Phurattanaprapin and P. Horata, "Extended hierarchical extreme learning machine with multilayer perceptron," in *Computer Science and Software Engineering (JCSSE), 2016 13th International Joint Conference on*. IEEE, 2016, pp. 1–5.

[13] 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.

[14] G.-B. Huang and C. K. Siew, "Extreme learning machine: Rbf network case." in *ICARCV*, vol. 2, 2004, pp. 1029–1036.

[15] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[16] H.-J. Rong, G.-B. Huang, N. Sundararajan, and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 4, pp. 1067–1072, 2009.

[17] A. Rubio-Solis and G. Panoutsos, "Fuzzy uncertainty assessment in rbf neural networks using neutrosophic sets for multiclass classification," in *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2014, pp. 1591–1598.

[18] Y. Qu, C. Shang, W. Wu, and Q. Shen, "Evolutionary fuzzy extreme learning machine for mammographic risk analysis." *International Journal of Fuzzy Systems*, vol. 13, no. 4, 2011.

[19] Z.-L. Sun, K.-F. Au, and T.-M. Choi, "A neuro-fuzzy inference system through integration of fuzzy logic and extreme learning machines," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 5, pp. 1321–1331, 2007.

[20] A. Rubio-Solis and G. Panoutsos, "Interval type-2 radial basis function neural network: A modeling framework," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 2, pp. 457–473, 2015.

[21] A. Rubio-Solis, G. Panoutsos, and S. Thornton, "A data-driven fuzzy modelling framework for the classification of imbalanced data," in *2016 IEEE 8th International Conference on Intelligent Systems (IS)*. IEEE, 2016, pp. 302–307.

[22] A. Rubio-Solis, P. Melin, U. Martinez-Hernandez, and G. Panoutsos, "General type-2 radial basis function neural network: A data-driven fuzzy model," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 2, pp. 333–347, 2018.

[23] A. R. Solis and G. Panoutsos, "Granular computing neural-fuzzy modelling: A neutrosophic approach," *Applied Soft Computing*, vol. 13, no. 9, pp. 4010–4021, 2013.

[24] A. Rubio-Solis, U. Martinez-Hernandez, and G. Panoutsos, "Evolutionary extreme learning machine for the interval type-2 radial basis function neural network: A fuzzy modelling approach," in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2018, pp. 1–8.

[25] A. Rubio-Solis and G. Panoutsos, "An ensemble data-driven fuzzy network for laser welding quality prediction," in *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[26] A. Rubio-Solis, G. Panoutsos, C. Beltran-Perez, and U. Martinez-Hernandez, "A multilayer interval type-2 fuzzy extreme learning machine for the recognition of walking activities and gait events using wearable sensors," *Neurocomputing*, 2019.

[27] C.-F. Juang, R.-B. Huang, and W.-Y. Cheng, "An interval type-2 fuzzy-neural network with support-vector regression for noisy regression problems," *IEEE Transactions on fuzzy systems*, vol. 18, no. 4, pp. 686–699, 2010.

[28] D. Wu, "Approaches for reducing the computational cost of interval type-2 fuzzy logic systems: overview and comparisons," *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 1, pp. 80–99, 2012.

[29] L. L. C. Kasun, Y. Yang, G.-B. Huang, and Z. Zhang, "Dimension reduction with extreme learning machine," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3906–3918, 2016.

[30] G.-B. Huang, N.-Y. Liang, H.-J. Rong, P. Saratchandran, and N. Sundararajan, "On-line sequential extreme learning machine." *Computational Intelligence*, vol. 2005, pp. 232–237, 2005.