Memories - Materials, Devices, Circuits and Systems 4 (2023) 100023

Contents lists available at ScienceDirect

Memories - Materials, Devices, Circuits and Systems

journal homepage: www.elsevier.com/locate/memori

# Hierarchical fuzzy deep learning system for various classes of images

Shashank Kamthan [*,1], Harpreet Singh [2]

*Wayne State University, Detroit, MI 48202, USA*

ARTICLE INFO

*Keywords:*
Fuzzy systems
Hierarchical fuzzy
Image classification
Image classes
Hierarchical systems
Deep learning
Target classification
Target detection
Hierarchical algorithms
Hierarchical structures
Image processing

ABSTRACT

There has been an increasing interest in the development of deep-learning models for the large data processing such as images, audio, or video. Image processing has made breakthroughs in addressing important problems such as genome-wide biological networks, map interactions of genes and proteins, network, etc. With the increase in sophistication of the system, and other areas such as internet of things, social media, web development, etc., the need for classification of image data has been felt more than ever before. It is more important to develop intelligent approaches that can take care of the sophistication of systems. Several researchers are working on the real-time images to solve the problems related to the classification of images. The algorithms to be developed will have to meet the large image datasets. In this paper, the generalized hierarchical fuzzy deep learning approach is discussed and developed to meet such demands. The objective is to design the algorithm for image classification so that it results in high accuracy. The approach is for real-life intelligent systems and the classification results have been shared for large image datasets such as the YaleB database. The accuracy of the algorithm has been obtained for various classes of images using image thresholding. The development of learning algorithms has been validated on corrupted and noisy data and results of various classes of images are presented.

## 1. Introduction

In the last few decades, deep learning techniques have attracted substantial attention within the artificial intelligence community. The accomplishments have been primarily due to enhanced capability of today's computer to gather, store and process large volumes or big datasets. This increased capability led to significant increase in performance, accuracy, and efficiency of the overall system. Therefore, it has become a possibility to gain highly accurate systems for several learning tasks such as image processing, target recognition, object identification and classification [1] within enhanced performance. The most common deep learning approaches are implemented either by fuzzy logic or by neural networks [2] due to their ability to perform under multi-function compositions and ability to drive in multi-stage learning processes.

Neural network defines a computational system developed in reference to biological neural logic i.e., the network interconnects by artificial neurons. Neural network [2,3] adapts the structure based on the training dataset. With the available dataset, the learning process of the model explores the best operating point.

Fuzzy logic is defined by many valued logics i.e., focus on imprecise, uncertain, and approximate reasoning, etc. on contrary to precise or fixed reasoning. The fuzzy logic characterizes the degree of truth and takes any value in range [0,1]. Whereas conventional binary sets characterize values as either 0 or 1. Because of this reason, fuzzy logic provides a facility to acknowledge the concept of partial truth. Since the preface of the fuzzy set theory concept in 1965 by Lotfi Zadeh [4], fuzzy logic acts as a most viable option to handle imprecise and uncertain data and helps in making decisions.

Fuzzy logic derives by mathematical decision depiction model, which supports imprecise, uncertain, or ambiguous data, whereas neural networks involve human factors to resolve real-life issues without modeling mathematically. Both fuzzy logic and neural networks support resolution [2] to non-linear issues, uncertain issues, or unrelated issues. Fuzzy logic incorporates the development of models using membership functions and interconnects them via rule base. Whereas neural networks apply human thinking process to resolve any issue i.e., learning process including specific algorithms and highly depended on available training dataset.

Fuzzy logic has benefited a lot in real-life applications [5–7] such as approximation, systems control, classification, clustering etc. With the

* Corresponding author.
*E-mail addresses:* kamthan.shashank@gmail.com (S. Kamthan), hsingh@eng.wayne.edu (H. Singh).

[1] Shashank Kamthan is working as a Sr. Researcher/ Subject Matter Expert — ADAS and Connected Vehicle System (C-V2X). He has completed M.S. and Ph.D. (electrical engineering) from Wayne State University, MI 48202.

[2] Harpreet Singh is the Professor in electrical engineering at Wayne State University, MI. His area of research is AI in autonomous vehicles, modeling, simulation, and visualization techniques.

https://doi.org/10.1016/j.memori.2022.100023
Received 31 July 2022; Received in revised form 12 November 2022; Accepted 15 December 2022

help of fuzzy logic systems, these applications have shown the flexibility in designing with ease of interconnections and obtaining linguistic results. Fuzzy logic-based systems are white-box implementations that associate transparency in actual implementation and validation.

In general, fuzzy logic systems are universal approximators that estimate accuracy of any continuous functions. Due to transparency and applicability of fuzzy logic, substantial growth has been accomplished in both theory and application in the last several decades. Fuzzy logic has been exercised to more complicated and complex systems but has become an obstacle for the complex and real-life applications [5–7] with high dimensional datasets [8–10]. The limitations due to dimensions can be summarized [9,10] as:

- Exponential increment in fuzzy rule-base with increase in number of input variables
- Exponential growth in number of parameters in mathematical relations of fuzzy systems with increase in input variables
- Increasing dataset necessity to identify the true behavior of the fuzzy system with increase in input variables.

These constraints due to dimensions [9,10] damage the transparency and interpretation of the system, which is the key pillar of fuzzy logic. This leads to the incapability of interpreting and advocating a huge number of fuzzy rule-bases with large numbers of parameters. Most of the real-life applications [7] have limited data availability. Due to this limitation, a huge number of rule-bases and parameters provide an over-fitting result that damages the generalization of the overall fuzzy systems.

To prevail over the limited dimension issue with fuzzy systems, several authors [8–11] [12] proposed designing of fuzzy systems in the form of hierarchical structure, commonly called as hierarchical fuzzy systems. Contrary to conventional fuzzy logic, where a single fuzzy logic system contains the information of full dimension. In hierarchical fuzzy systems, several smaller sub-dimension systems, represented as sub-systems, are interconnected in the form of hierarchy. Each sub system is defined by a fuzzy logic unit. Apart from using hierarchical fuzzy logic for prevailing the dimension issue, most of the researchers leverage hierarchical fuzzy systems to refine the accuracy of the fuzzy system. These refined algorithms of hierarchical fuzzy systems are used in various applications such as data or image classification [1], clustering, path prediction/estimation and tracking system etc.

### 1.1. Limitations of conventional fuzzy systems

For the conventional fuzzy logic, fuzzy terms and membership functions are mainly used to segment input space and identify all the possible interconnections in the form of rules. Every input is categorized by its membership functions. As a repercussion, the number of rules grows exponentially as the number of input variables increases. The matrix below shows the grid structure of two inputs: X1 and X2; and their membership functions (assuming each input has 'm' number of membership functions ranges from [1, 2, …, m]) and each fuzzy value will then be defined as {X1(1), X1(2), …, X1(m)} and {X2(1), X2(2), …, X2(m)}. Fig. 1 shows the combination of two inputs in the form of the matrix. Every combination leads to a rule and all the combinations give the maximum number of rules possible [9,10].

A fuzzy rule-base for the above can be shown by pseudo algorithm as:

*If input X1 has membership functions as X1(i) and input X2 has membership functions as X2(j), then the output is Y(i)(j). Where 'i'= 1,2,3…m and 'j' = 1, 2, 3, … m*

Since both the inputs have 'm' membership functions, to cover the entire domain, the maximum number of rules for a fuzzy system is $m^2$ and is termed as rule base. Now, assume there are 'n' input variables, and each variable has 'm' membership functions, then the total number of possible rules will be ($m^n$). For example: a fuzzy system consists of 10 input variables with each of them having 10 membership functions, containing a total of ($10^{10}$) rules. It is almost infeasible to design conventional fuzzy with that many possible combinations.

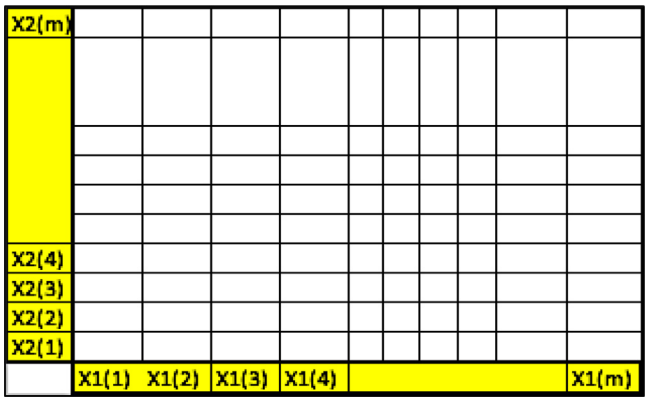



**Fig. 1.** Matrix of 2-input membership functions.

With gradual increase in input variables, there is an associated increasing number of parameters. Assume 'p' is the number of parameters required for every rule-base, so with 10 input variables and each with 10 membership functions requires "p. ($10^{10}$)" parameters.

Another limitation is the increase in training data required with increase in input variables. It is expected that the training dataset shall have at least the same number as the number of parameters. The increase in the number of parameters leads to the increase in data for the input variables.

The above limitations have lots of consequences, such as:

- Loss in transparency and interpretability because of incapability to understand and generalize large number of rules and parameters
- Real-life applications [7], due to limited data availability, lead to overfitting of rules and data. This may damage the generalization of the fuzzy system.
- Requires excessive computational power and large memory to process data/rules

These limitations are the bottleneck for applying and designing fuzzy based systems for large, complicated, and complex applications. This further restricts the conventional fuzzy logic to solve various applications via intelligent tools and with high dimensions successfully.

### 1.2. Hierarchical fuzzy tree structure

Hierarchical fuzzy logic was first introduced in the early 90's by Raju, Zhou and Kisner [8,9,13] to resolve the limitation of dimension of conventional fuzzy logic. In general, a high-dimensional fuzzy system is divided into several low-dimensional fuzzy systems, commonly known as fuzzy logic units. These fuzzy logic units are interconnected in hierarchical manner. In general, Hierarchical fuzzy systems are categorized into various hierarchy levels and each level comprises several sub-systems. Final hierarchical fuzzy sub-systems are described by combining these various sub-systems committing to the final solution. The sub-system at the lowest level acquires only real inputs and their outputs become the inputs to the next level in the hierarchy and the final layer possesses the final solution. This design has attracted a lot of attention from various researchers.

A hierarchical structure [13] shown in Fig. 2 consists of multiple levels and multiple subsystems at each level. The outputs from the current level become the inputs to the next level and output from the final level drives the final solution.

Similarly, Chung and Duan [8,9,12] categorizes hierarchical fuzzy systems in following three forms:

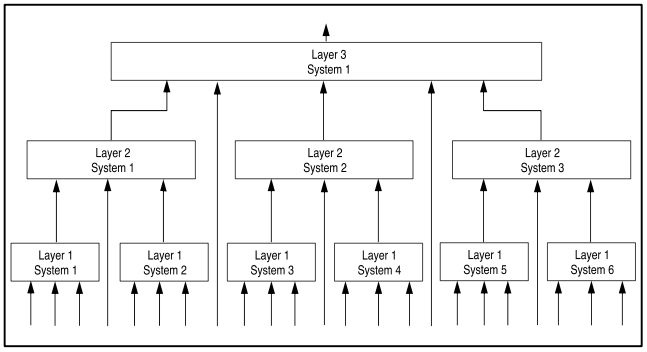



**Fig. 2.** Cascaded hierarchical tree structure.

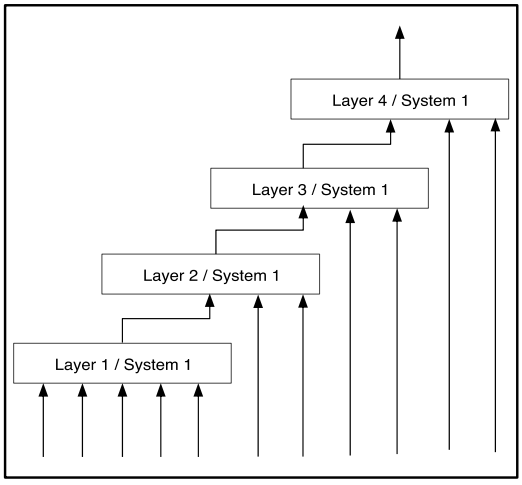



**Fig. 3.** Incremental hierarchical tree structure.

1. Incremental structure, shown in Fig. 3.
2. Aggregated structure, shown in Fig. 4.
3. Cascaded structure, shown in Fig. 2.

Incremental structures are described by multi-stage reasoning. Each stage is represented by one level and every level contains only one sub-system. The lowest level (lth level) consumes the actual inputs, whereas the (l–1) th level consumes output from previous level and part of input variables. To make a precise system, it is important that the most valued input variables should be assigned to the lowest level possible. In the words of Wang and colleagues [3,8,9,12], a sub-system with more valuable information shall be applied first and less valuable information shall be applied later. This way, the output of the lowest subsystem will then be propagated to the upper level within affecting subsequent levels.

Aggregated structures are described by multi-layer and multi-sub-systems, where each level may have single or multiple subsystems. The lowest layer consumes all the real inputs. Other consecutive layers consume the output from the previous level as inputs.

Cascaded structure is the combination of both incremental and aggregated structures, as shown in Fig. 2.

Wang and colleagues [3,8,9,12] proposed methodology for incremental structures. They considered various membership functions including triangular membership functions and gaussian membership functions as a universal approximator that approximates the accuracy for any non-linear functions. For example, suppose there are '$n$' number of input variables, and each input is defined by '$m$' membership functions, and '$L$' is the number of levels, $n_i$ is the number of inputs at $i$th level, which also includes output from previous level $(i-1)$th, in this case, the total number of rules is represented by Eq. (1) as:

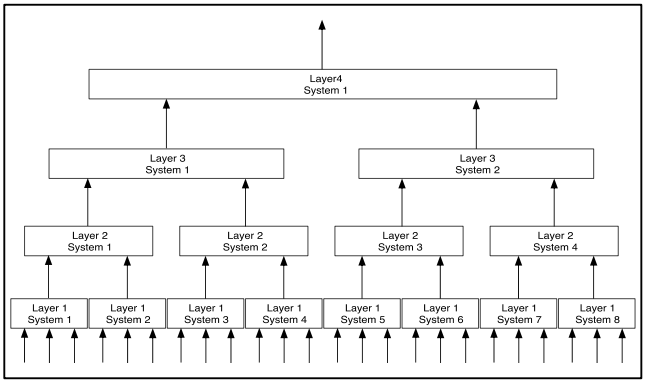



**Fig. 4.** Aggregated hierarchical tree structure.

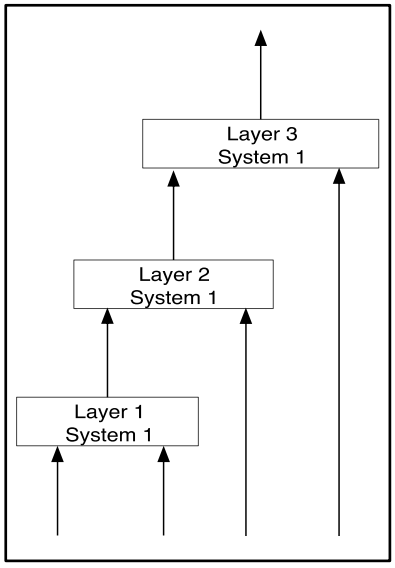



**Fig. 5.** Incremental structure with $N_i = 2$.

$$Number\ of\ Rules\ (maximum) = \sum_{i=1}^{L} m^{n_i} \tag{1}$$

To determine the minimum number of fuzzy rules, let us suppose $N_{mi} = 2$ i.e., every model with two inputs, as shown in Fig. 5. The maximum number of rules when $N_i = L$ (same as conventional fuzzy systems) is $M^N$ and total number of rules with hierarchical fuzzy logic can be expressed in Eq. (2) as below:

$$\sum_{i=1}^{N-1} M^2 = (N-1).M^2 \tag{2}$$

For example, suppose $N = M = 5$, $L = 4$, where $N_1 = N_2 = N_3 = N_4 = 2$, then the number of rules for hierarchical structure is $4*(5^2) = 100$, if we use only one level and one system i.e., equivalent to a conventional fuzzy system, then the number of rules becomes $5^4 = 625$.

The maximum reduction in rules between hierarchical fuzzy logic and conventional fuzzy logic is 625 −100 = 525, which is more than 5 times the actual rule base for conventional fuzzy.

### 1.3. Problem statement

Despite several advancements discussed in literature, still there are several open points and limitations such as narrow systems performance when data is contrived by inherent noise, ambiguity, uncertainty, incompleteness, vagueness. These limitations reduce the effectiveness and performance of the system and thus minimizes the

ability of deep learning techniques to confront real-life applications [7], which can be safety and security critical. This requires the necessity to illustrate data in a better way by collection of additional data, more analysis and clean dataset. Deep learning approaches often have low transparency. This reduced transparency limits the understanding and interpretation to other users.

The aim of this paper is to address the problems and limitations to develop a fuzzy based model, which can handle large volumes of data, especially real sets of images. With ability to maintain composition both at functional level and linguistic level. Fuzzy based models are well suited and have potential to handle data efficiently and provide a high transparency due to its rule base nature. The recent development has been around hierarchical fuzzy systems or fuzzy hierarchy networks.

It is a well-accepted fact that the conventional fuzzy systems have limitations to data dimension [9,10]. This limitation restricts the usage of fuzzy systems to solve complex problems and applications with large data dimensions. In the last few decades, hierarchical fuzzy systems have appeared to be a viable solution to overcome the limitation of conventional fuzzy systems and, so far, bring significant attention. This paper presents an approach to develop hierarchical fuzzy based models with the focus to deal with large volumes of data, without compromising the performance and effectiveness of the overall system.

As per authors' knowledge, no one has developed the hierarchical fuzzy system using image thresholding to process various classes of image dataset. In this paper, the proposed algorithm has been validated on a large, real-time image dataset i.e., YaleB dataset. For improved and better analysis, it will be worthwhile to extend the scope of this work towards other datasets [14–17], for example, video data, visual data, x-view, image-net, etc. This scope would be a feasible option for researchers to work and improve the performance.

## 2. Hierarchical system design

The representation of imprecise and uncertain data for real-life applications poses an urgent and unique challenge. Deep learning techniques such as neural networks, fuzzy logic, fuzzy neural networks etc. present meaningful approaches to handle these challenges of maintaining datasets with large dimensions. To address the issues of generalization, transparency and large dimensions, several researchers [5,6,8–10,18] started the idea of designing fuzzy systems in hierarchical structure. Instead of designing conventional systems with high dimensions, the system will be divided into subsystems with low dimensions and connected with each other in the form of hierarchy. The hierarchical system is represented as a multi-input- single-output system, and on other hand, without losing any generalization, multi-input–multi-output systems can be represented in several multi-input-single-output subsystems. Out of various algorithms available in the literature, hierarchical fuzzy logic, i.e., fuzzy logic system design in a form of hierarchy, introduces as a most effective approach to handle such large dimension dataset in most effective manner. The common applications to hierarchical systems are classification, clustering, planning, and tracking systems etc. Due to the limitation of fuzzy logic rule-base for large datasets, an approach to design fuzzy logic systems arranged in hierarchical architecture has been discussed in this paper, to reduce the number of rule-bases without compromising effectiveness and efficiency of the overall system.

With fixed number of input variables, an algorithm is presented by Radek Sindelar [11] on Hierarchical fuzzy systems suggesting the use of clustering approach to help divide the system into subsystems, identify the center points and correlate with rule-base of various subsystems with the help of clustering. However, the limitation of the algorithm lies with the restriction to a small number of input variables and leveraging a high dimension dataset is not a feasible option. The research also compared the accuracy and effectiveness of various weighted approaches for defuzzification such as maximum, centroid etc. to make the system more efficient.

Hierarchical fuzzy tree structure [8–11] has been described in Section 1 earlier i.e., the interconnections between various subsystems are made in a way such that the output of lower-level subsystems becomes the input to next level subsystems and so on. The proposed algorithm starts with the lower level, where inputs are data from the real images, especially with large dimensions. 2-side gaussian membership function has been preferred because of its smooth, continuous, and differentiable properties. The realization of fuzzy systems has been done in various steps such as: extracting rules from training networks, optimizing network parameters using fuzzy logic, adding fuzzy logic behavior to define required networks, realizing fuzzy membership values, and representing fuzzy blocks for multi-layer networks. The above said steps can be obtained for any of four given algorithms i.e., grid partitioning, back propagation, subtractive clustering, and fuzzy c-mean clustering. In this paper, we choose fuzzy c-mean clustering methods due to its ability to do system design in both Mamdani and Sugeno type fuzzy inference systems. Like conventional fuzzy, for multi-layer networks, hierarchical fuzzy logic units are characterized by three sub-systems: fuzzification, fuzzy inference systems for rules or connections and defuzzification. The "centroid" weighted approach has been considered for defuzzification. The comparison with other weighted approaches such as Maximum etc. is listed in the result section later. The approach suggested a cluster approach to define subsystems, extracted from image data.

### 2.1. Conceptual flowchart

Fig. 6 represents the conceptual flowchart for designing hierarchical fuzzy logic systems [6,8–10] [19]. As discussed previously on hierarchical tree structure, the algorithm for both aggregated and incremental structures have been examined and then the comparison of results has been made to get the closest result in reference to expected behavior.

One of the objectives of designing the hierarchical fuzzy systems is to achieve better performance and accuracy by comparing various tree structures. The algorithm offers the generalized design process consisting of both incremental and aggregated hierarchical tree structures. As shown in flowchart (Fig. 6), either aggregated or incremental tree structure can be picked at a time. In our research, the selection of hierarchical tree structure is made as per user input i.e., depending on the requirements, the user manually selects tree structure. The results of both aggregated and incremental tree structures have been presented separately.

The aggregated tree structure is advantageous when all the inputs are unified, and the system requires uniform transition from one level to the next level of the hierarchy. The incremental tree structure is advantageous when input parameters have varied impact on the system and require variable transition among different levels. The cascaded hierarchical tree structure is the amalgamation of both aggregated and incremental hierarchical tree structures. The ideal selection of hierarchical tree structure is based on the system requirements and other external factors such as the prioritization or ranking of inputs and hardware capabilities. For e.g., if inputs are coming from different sensors and few of them are more critical than others, incremental tree structure is preferred and if all the inputs have the same priority, then aggregated tree structure can be leveraged. This is due to the fact that in aggregated tree structure, all the real inputs are consumed at the lower level of the hierarchy and the hierarchical structure does not require any ranking or prioritization among the various inputs. On the other hand, for incremental tree structure, a segment of real inputs is consumed at every hierarchical level. For the effective incremental tree structure design, it is highly important that the critical or higher priority input shall drive the lowest level of the hierarchical level and lesser priority input segment drives the top hierarchy. This can be achieved by establishing the ranking or prioritization system of the inputs based on their importance and effectiveness on the overall system. Once the input ranking has been identified, the system can be

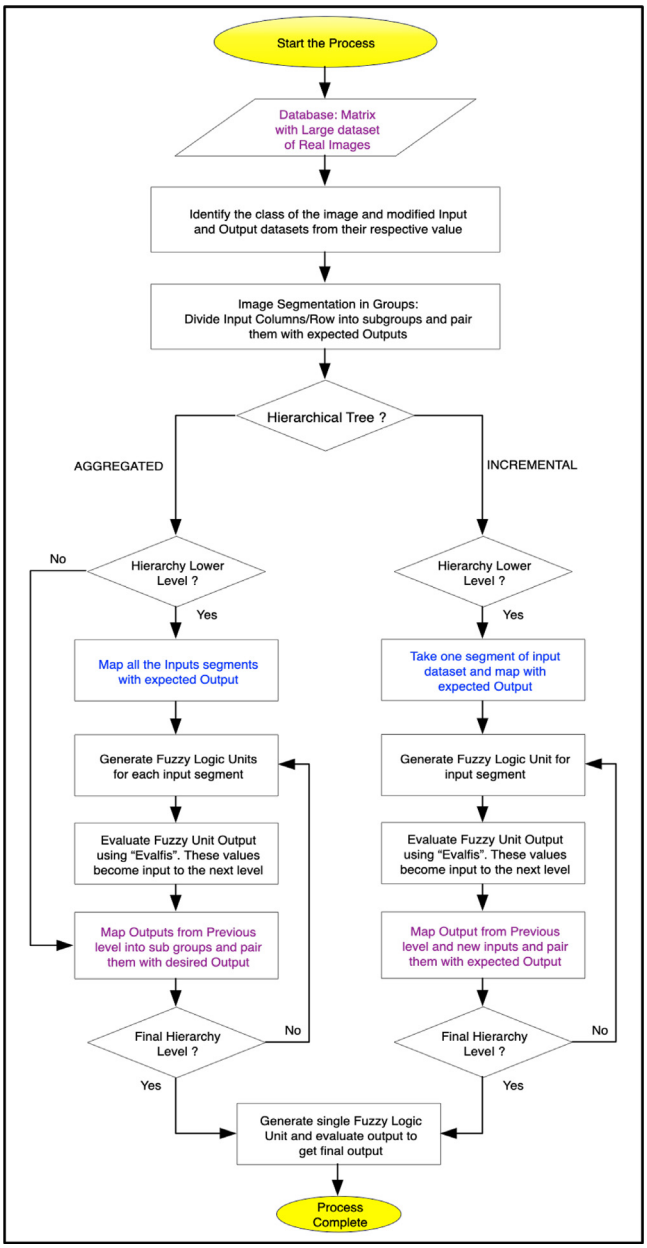



**Fig. 6.** Conceptual hierarchical fuzzy logic implementation for the system.

designed as specified. It will be worthwhile to design and develop the ranking system and segment the inputs before applying the algorithm. It is up to the user discretion on how to utilize the cascaded structure to achieve optimal performance and accuracy. The algorithm for cascaded tree structure can be developed further.

The real input dataset attached to a specific level i.e., lower level of aggregated structure consumes all the real inputs, on the other hand, for incremental structure, all the levels consume real inputs along with output from previous levels. Database, in Section 3, shows the big matrix of image data. Keeping output as expected labels, the data matrix will further be segmented as a portion of column and row matrices, where each segment represents one fuzzy logic unit of the hierarchical systems.

### 2.2. Image segmentation

Segmentation of image [20] is the process to partition a single image in multiple objects, commonly known as image segmentation [9,16,17,20]. The basic usage of image segmentation is to streamline image representation for better analysis i.e., identify objects, detect boundaries etc. This process assigns labels to every pixel but with varied characteristics. The behavior can be extracted from a different set of contours obtained from Image.

The basic structure of image segmentation [20] is image thresholding, which turns gray-scale or RGB image into binary form. There are several methods available in the literature for image thresholding, such as:

1. *Entropy Method — This method uses the entropy of both foreground and background regions using the probability distribution functions. Assuming the foreground has largest entropy compared to background or previous ground*
2. *Balanced Histogram Thresholding — This method uses Iteration thresholding method. The image is classified into two regions front- and back-ground and weighs on histogram to see which side is heavier and then to make it lighter, removes the weight from the heavier side.*
3. *K-Means Clustering — This method is based on iterations. Every set of image pixels is divided into K-clusters that refers to the minimum distance between pixel and cluster center point. In the final step, apply any weighted approach i.e., Centroid, Maximum, Minimum etc. to get a result.*
4. *Otsu's Method (Maximum Variance) — For Image processing, Otsu's method commonly referred after Nobuyuki Otsu is used for Image thresholding. The algorithm provides intensity or threshold values that differentiates between two-pixel classes. This approach determines the intra-or inter-class intensity variance and classifies image thresholding points where the variance is high.*

Image segmentation approach [9,16,20] partitions a single image into multiple objects. It streamlines the representation of images for better analysis in a fuzzy environment. In this paper, all the images in the dataset are available in pixel format. Considering the format of the image, Otsu's method for image thresholding has been preferred. One of the major rationales behind the selection of Otsu's method is its ability to provide multiple threshold values that differentiate various pixel classes. Here Otsu's method is used to identify multi dimension threshold values. These multiple threshold values are then used to modify real inputs and generate new sets of input.

Because of the capability of differentiating image pixel classes, in this paper, Otsu's multi-dimensional method [9,16] has been used to create a generalized algorithm using hierarchical fuzzy logic for a large database of images. The one-dimension Otsu's method has the limitation to have assumption of bi-model distribution. This method applies better when there is a deep and sharp curve. The problems discussed in this paper can be resolved by multi-dimensional methods and thus the reason behind using the same in the algorithm.

It is observed that in various databases available for research, there is a minimal difference in membership values between two image samples. This restricts the segmentation of the data matrix and contradicts the design of the fuzzy logic system rule base. With similar membership values, the expected output values are different. One of the solutions to overcome this problem is to use Otsu's image thresholding algorithm to find multiple threshold values and multiply these values with input membership values to create new input matrix but keeping output membership values remain the same and is shown by Eq. (3) below:

$$\sum_{n=1}^{N} X_n^{New} = \sum_{n=1}^{N} \prod_{p=1}^{P} T_p * X_n \tag{3}$$

Where, '$N$' = number of columns for input matrix per sample; and '$P$' = multi-threshold values from single sample input matrix.

### 2.3. Algorithm design

The flow in hierarchical fuzzy system design [8–10,21] is, except at the lower level, the outputs from previous level will become input to the next level alongside real inputs and so on and so forth. From second level onwards, consider previous output as input to the next level; combine all the values and data segments in sub-systems; and evaluate next values for next level by keeping the same expected output; generate fuzzy logic unit models for every segment. For the last level, it is expected to have only one fuzzy logic unit, which provides final output. In this paper, fuzzy c-mean clustering is considered due to its ability to generate center points that can directly be correlated and used as a fuzzy rule base. With required cluster center points, these regions are contrived to n required points depending on number of rules required per fuzzy logic unit. Thus, center points can be converted directly to rules. The steps for the algorithm to design hierarchical system for different tree structure is define in following steps below:

1. *Consider training data consists of large database with images*
2. *Define following information:*
    a. *Number of hierarchical levels*
    b. *Fuzzy logic units in every level*
    c. *Number of clusters corresponds to required number of rules for every fuzzy logic unit*
3. *Obtain multi-thresholding value from Otsu's method and multiply with each input elements to create new input*
4. *Group the Input dataset in various segments keeping the output remain same as expected. For instance, the input dataset of [165, 1024] can be divided in to 1408 samples of [15, 8] or 704 samples of [15, 16]*
5. *For the first or lower level of hierarchical structure i.e., aggregated tree structure, all the real input adheres to the first or the lower level of hierarchical structure. This layer consists of fuzzy logic unit generated for all the input samples extracted in Step 3, the steps to generate fuzzy logic unit is as follows:*
    a. *Take input segment/ Sample and map with desired output*
    b. *Using MATLAB commands, the syntax to generate Fuzzy inference system is defined below:*
        i. *Fuzzy System = genfis3([Input], [Output],'FIS Type', 'Number of Cluster'); Where –*
            1. *[Input] = Input sample matrix*
            2. *[Output] = Expected output matrix*
            3. *'FIS Type' = Choose type of inference system: Mamdani or Sugeno*
            4. *"Number of Clusters" = Desired number of rule base for fuzzy logic unit*
        ii. *Evaluate the output of the fuzzy logic unit with 'evalfis' command*
        iii. *Store the evaluated output in a buffer*
6. *For the next level(s) of hierarchical structure,*
    a. *Take all the output from previous hierarchy levels and consider them as input to current level*
    b. *Create a group of segments of matrix and map each segment with map with desired output*
    c. *Using MATLAB commands, the syntax to generate Fuzzy inference system is as below:*
        i. *Fuzzy System = genfis3([Input], [Output],'FIS Type', 'Number of Cluster'); Where —*
            1. *[Input] = Input sample matrix*
            2. *[Output] = Expected output matrix*
            3. *'FIS Type' = Choose type of inference system: Mamdani or Sugeno*
            4. *"Number of Clusters" = Desired number of rule base for fuzzy logic unit*
        ii. *Evaluate the output of the fuzzy logic unit with 'evalfis' command*
        iii. *Store the evaluated output in a buffer*
7. *For the last hierarchy level*
    a. *Take all the outputs from previous levels and consider them as input to current level*
    b. *Create a group of segments of matrix and map each segment with map with desired output*
    c. *Using MATLAB commands, the syntax to generate Fuzzy inference system is defined below:*
        i. *Fuzzy System = genfis3([Input], [Output],'FIS Type', 'Number of Cluster'); Where —*
            1. *[Input] = Input sample matrix*
            2. *[Output] = Expected output matrix*
            3. *'FIS Type' = Choose type of inference system: Mamdani or Sugeno*
            4. *"Number of Clusters" = Desired number of rule base for fuzzy logic unit*
        ii. *Evaluate the output of the fuzzy logic unit with 'evalfis' command*
8. *Evaluate the last level of hierarchy to get final output.*

Algorithm to design hierarchical tree structure has been shown below. For aggregated structure, all the real inputs are consumed by lower levels. Next level onwards, each level has output from the previous level as input to a fuzzy logic unit. Final level provides the final output. However, incremental structure has one fuzzy logic unit at every level and except lower level, all the other levels have fuzzy logic units with input as a combination of real input segment and output from previous level. Final layer provides the final output evaluated. MATLAB 'Evalfis' command [22] is leveraged to evaluate the respective outputs from fuzzy units in the given hierarchical tree structure. For the given input values, this command evaluates the fuzzy inference system and returns the output value. The default evaluation option has been selected.

The cascaded structure can be designed using the algorithm by merging both aggregated and incremental design procedures.

The algorithm consists of a Machine learning approach having unsupervised learning with the help of Fuzzy c-mean clustering methodology. The fuzzy c-mean clustering (FCM) method is considered for designing the fuzzy units due to its ability to generate center points and convert the generated center points into a form of rules and later transitioned to rule base. Fuzzy c-mean clustering is a technique in which every data element of the given dataset belongs to every cluster to a certain degree. In terms of fuzzy logic, data elements, leaning towards the center of the cluster, will have higher degree i.e., higher membership values and vice versa. By default, the fuzzy c-mean clustering assigns every data element a unique membership value for every cluster. Using continuous iteration method, the FCM corrects the center of the clusters by adjusting the distance between any two data elements and membership values for every data element. Keeping

in mind the size of the image, maintaining the default clustering option could provide an optimal solution.

***Algorithm:*** *Hierarchical Tree Structure*

***Training Data:*** *Large database of real images*

***Thresholding:*** *Find the image thresholding parameter for each sample and multiply with every element of the same sample. The result becomes the modified input parameters, and the output remains the same.*

***Segmentation:*** *Using Mathematical analysis, create and pool similar inputs and pair them with the expected output*

***Data Grouping:*** *Divide the data in number of columns and rows in reference to same expected output*

***Number of Levels:*** *Define number of hierarchy level required*

***Find Cluster:*** *Find center points (relation between Input and Output) for each group by following syntax*

***Center Points = findcluster ([Input, Output], Total Cluster)***

***Execution:***

- *Define membership functions for every Input and Outputs*
- *Consider new input data extracted by image thresholding using Otsu's method and expected output*
- *Create different sub-groups or segments from input dataset, keeping same output*
- *For: Hierarchy Level = '1' to (Final Level)*
  - ***If (Lower level of hierarchy)***
    - *Generate Fuzzy Inference system for all the sub-groups as follows:*
      ***For (Number of Sub-groups)***
      - *Find center points (IO Mapping)*
      - *Generate Fuzzy inference system FIS = genfis3([In], [Out], 'FIS Type', 'Cluster')*
      - *Evaluate Output using "evalfis"*

      ***End process***
  - ***Else (Level 2 to Final level of hierarchy)***
    - *Take another input sub-group and Output from previous level and map these values with real output*
    - *Create different sub-groups or segments from input (previous level output) dataset, keeping same output*
    - *Generate Fuzzy Inference system for all the sub-groups. The syntax is:*
      ***For I = 1 to Number of Groups***
      - *Find center points (IO Mapping)*
      - *Generate Fuzzy inference system FIS = genfis3([In], [Out], 'FIS Type', 'Cluster')*
      - *Evaluate FIS using "evalfis"*

      ***End process***
    - ***If (Final Level of Hierarchy)***
      - ***Evaluate final output***

***Result:*** *The final output is the output extracted from final level*

This will enhance computational time drastically and eventually lower down the overall performance. Using the iteration method for the image database with the help of MATLAB 'fcm' command [23], the approximate range for total number of clusters has been analyzed and considered for further use. The MATLAB 'genfis' command [22] has been exploited to create center points using a fuzzy c-mean algorithm and to generate a fuzzy inference system by converting these center points into rule base. The algorithm manages every fuzzy unit and packs the output at each hierarchical level independently. There will be numerous ways the algorithm can be managed further. If new data overrides the existing data and forms the same clusters, the generation of new center points or fuzzy clusters is not needed. If the new data does not override with existing dataset, the generation of new center points is needed. This can create a situation to re-design fuzzy inference systems with new clusters. Examples include adding additional hierarchical structure and aggregate output with the existing model or the additional fuzzy unit can be designed separately and later added at the lowest or highest hierarchical level or re-learn complete structure while maintaining the same total number of clusters.

The description of the database used to validate the algorithm is discussed in Section 3. The validation of these algorithms for hierarchical tree structure, for all the image classes discussed in this section, has been shown in Section 4.

## 3. Database – real images

In real-life applications [7], there is always a challenge to handle datasets of large dimensions and there is an urgent need for a strategy to develop distributed systems [5–7]. Various companies such as Google, Amazon, Yahoo, Facebook, LinkedIn etc. already acquire large databases including huge databases of real images for object detection and target identification [17,24] etc. and are now striking the benefit of acquiring and processing such data. The dataset for real-life applications is non-linear and robust in nature, i.e., comprises supervised and non-supervised datasets.

For the development of efficient systems, it is easy to use existing pre-trained models that are available for image classification, image segmentation [15,16,20], handwriting spot, face recognition etc. These pre-existing models contain pre-processed images i.e., images with fixed sizes, normalized image intensities within RGB or Gray range between [0, 255].

To benchmark the algorithm, it is always beneficial to leverage standardized test databases, as it would be convenient for researchers to compare results directly. With many image databases available and currently in use, the choice of dataset shall be specific to the property to be tested i.e., depend on the features of the algorithm. In this paper, Yale face database [24] has been considered, since it provides more images per class and more samples. This database is available publicly but limited to research purposes only.

YaleB database [17,24] consists of 165 images of 15 individuals and all the images are in GIF format. In general, per individual, there are 11 images defined, and the categorization of these images are based on their facial expressions and configurations as: center-light, left light, right light, with glasses, without glasses, happy, normal, sad, sleepy, surprised, wink.

The standard Yale face database [17,24] contains a training and testing set of images both in GIF format and in MAT format. Training dataset is used for learning algorithms and testing dataset is used for validation purposes. MAT format is convenient to extract in MATLAB environment [22] and can be used to image resize and normalization of images. In this paper, training data has been used to develop the hierarchical fuzzy logic system, whereas testing data has been used to evaluate the system to validate the effectiveness and efficiency.

Images are available in two sizes (in grayscale): 32 × 32 and 64 × 64. Depending on the requirement, these images can be resized, normalized etc. In recent times, the Yale database has been transformed to Extended Yale database that has the database of 38 individuals and includes 64 facial expressions and configurations. Fig. 7 shows the sample images of the Yale Face database.

### 3.1. Image data in pixel format

Fig. 8 illustrates various classes of images in pixel format. These classifications have been done using the Yale face database. Each data is a matrix of [165 × 1025], with Input of size [165,1:1024] and output label of size [165,1025:1025].

Each number of rows shows the number of samples, whereas input columns for each row represent one pixel of image in 1-D, in other

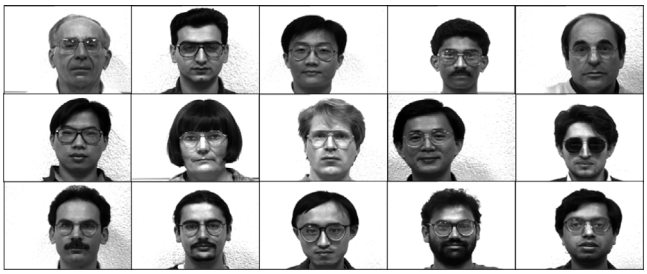


**Fig. 7.** Sample images from Yale face database.

words, the size of the images is 32 × 32 and each image is represented in a row matrix with size [1,1024]. The row matrix represents the one-dimensional matrix of a single image of size 32 × 32, where there will be one row for each image and 32*32 = 1024 columns consist of pixel data.

The dataset of various image classes [9,17,24] has been discussed. Each dataset is represented in different colors. The data is segmented into sub-columns and sub-rows. These segmentations are shown in gray and blue color. The level of the hierarchical system consists of fuzzy logic units designed for every segment and is connected in hierarchical architecture. For example, in the case of aggregated hierarchical structure, all the fuzzy models, designed using these segments, must lie in lower level and output from lower level becomes input to next level and so on. On the other hand, for incremental hierarchical structure, every segment represents a single layer of hierarchical structure but apart from the first layer, output from the previous layer is also added as one of the inputs for the next layer. The four classes of images [9] are shown in Fig. 8 as:

*1. Raw Dataset (Default)*

This class explains the raw image data in grayscale format. In the current example, the size of the data is [165 × 1025], where Input size is [165 × 1024] and Output size is [165 × 1]. Assuming $X_{m,n}$ defines the input matrix and $Y_m$ defines Output matrix, where '$m$' is the number of samples and '$n$' is the total columns required to define one image. The dataset in pixel format is represented in Fig. 8(A).

*2. Values in Fuzzy Range [0,1]*

This class explains the modified image data, but all the membership values are within Range [0, 1]. The raw matrix for Input and Output with size of the data is [165 × 1025]. Assume $X_{m,n}$ defines the raw input matrix and $Y_m$ defines Output matrix, where 'm' is the number of samples and 'n' is the total columns required to define one image. Then the modified Input matrix is represented by Eq. (4) below and the dataset in pixel format is represented in Fig. 8(B):

$$X_{m,n}^{modified} = \sum_{n=1}^{N}\sum_{m=1}^{M}\left(\frac{X_{m,n}}{255}\right) \tag{4}$$

Where, $N$ = total number of columns in input matrix and $M$ = Total number of samples. This is the case of normalization and scaling of all the inputs within range [0,1]. The fuzzy logic works best in the range of [0,1]. The sole purpose is to normalize and scale all the inputs within fuzzy limits, which is between 0 and 1.

*3. Categorized in 4 Membership values*

This class explains the modified image data in grayscale format, but all the membership values are distributed within four membership functions and all the membership values lie in range [0, 255]. The raw matrix for Input and Output with size of the data is [165 × 1025]. Assuming $X_{m,n}$ defines the raw input matrix and $Y_m$ defines Output matrix, where 'm' is the number of samples and 'n' is the total columns required to define one image. Then the modified Input matrix is represented by Eq. (5) below and the dataset in pixel format is represented in Fig. 8(C):

$$X_{m,n}^{modified} = \begin{matrix} 224; \in \{192 \le X_{m,n} \le 255\} \\ 160; \in \{128 \le X_{m,n} < 192\} \\ 96; \in \{64 \le X_{m,n} < 128\} \\ 32; \in \{0 \le X_{m,n} < 64\} \end{matrix} \tag{5}$$

*4. Gaussian Noise*

This class results in a modified image dataset with added gaussian noise with mean 0.0 and variance 0.001 and it is expected that all the membership values lie in range [0, 255]. The raw matrix for Input and Output with size of the data is [165 × 1025]. Assuming $X_{m,n}$ defines the raw input matrix and $Y_m$ defines Output matrix, where 'm' is the number of samples and '$n$' is the total columns required to define one image. Then the modified Input matrix is represented by Eq. (6) below and the dataset in pixel format is represented in Fig. 8(D):

$$X_{m,n}^{modified} = \sum_{n=1}^{N}\sum_{m=1}^{M}(X_{m,n} + U * X_{m,n}) \tag{6}$$

Where '$U$' is the uniform distribution with mean 0.0 and variance 0.001.

One of the objectives of the algorithm is to obtain the accuracy as close to desired behavior. Keeping in mind the value of membership functions and the working range in designing fuzzy inference systems,

**(A) DATABASE WITH DEFAULT FORMAT**

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 24 | 24 | 26 | 24 | 29 | 30 | 29 | 22 | 29 | 48 | 61 | 102 | 143 | 180 | 185 | 167 | 136 | 148 | 241 | 255 | 255 | 221 | 247 | 255 |
| 70 | 68 | 49 | 53 | 50 | 50 | 37 | 33 | 26 | 13 | 17 | 61 | 69 | 109 | 144 | 179 | 195 | 243 | 229 | 254 | 255 | 255 | 255 | 255 |
| 74 | 71 | 64 | 62 | 41 | 43 | 57 | 57 | 56 | 58 | 48 | 56 | 72 | 96 | 223 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 255 |
| 29 | 31 | 32 | 36 | 33 | 35 | 36 | 37 | 18 | 38 | 73 | 96 | 142 | 173 | 198 | 176 | 158 | 158 | 224 | 255 | 255 | 254 | 241 | 255 |
| 42 | 53 | 50 | 62 | 43 | 53 | 50 | 54 | 54 | 48 | 53 | 67 | 88 | 115 | 140 | 182 | 191 | 185 | 221 | 255 | 255 | 255 | 255 | 255 |
| 101 | 94 | 84 | 76 | 74 | 65 | 59 | 77 | 104 | 132 | 150 | 162 | 172 | 176 | 176 | 174 | 172 | 175 | 169 | 161 | 157 | 155 | 155 | 160 |
| 4 | 4 | 5 | 6 | 9 | 7 | 4 | 2 | 1 | 2 | 9 | 23 | 28 | 22 | 17 | 18 | 30 | 26 | 39 | 53 | 45 | 23 | 49 | 68 |
| 59 | 56 | 42 | 47 | 45 | 49 | 37 | 24 | 19 | 24 | 24 | 44 | 65 | 74 | 80 | 79 | 93 | 99 | 151 | 176 | 202 | 180 | 123 | 59 |
| 40 | 44 | 48 | 58 | 51 | 31 | 29 | 29 | 26 | 48 | 62 | 89 | 133 | 160 | 164 | 155 | 165 | 173 | 173 | 180 | 140 | 63 | 70 | 166 |
| 68 | 68 | 65 | 46 | 35 | 28 | 29 | 30 | 34 | 64 | 110 | 139 | 184 | 192 | 185 | 178 | 172 | 167 | 163 | 162 | 162 | 141 | 218 | 127 |
| 56 | 47 | 44 | 48 | 55 | 57 | 43 | 34 | 32 | 43 | 61 | 71 | 97 | 98 | 103 | 142 | 170 | 160 | 218 | 253 | 255 | 255 | 255 | 255 |
| 27 | 30 | 27 | 20 | 21 | 17 | 18 | 19 | 26 | 33 | 50 | 55 | 117 | 156 | 194 | 199 | 189 | 179 | 188 | 218 | 199 | 237 | 243 | 255 |
| 17 | 16 | 12 | 12 | 11 | 12 | 15 | 19 | 40 | 87 | 89 | 113 | 118 | 78 | 73 | 103 | 126 | 142 | 152 | 112 | 254 | 255 | 238 | 255 |
| 24 | 19 | 16 | 17 | 16 | 25 | 60 | 58 | 93 | 111 | 127 | 98 | 78 | 94 | 143 | 137 | 154 | 136 | 255 | 255 | 255 | 250 | 254 | 251 |
| 24 | 30 | 38 | 35 | 28 | 21 | 15 | 16 | 18 | 25 | 83 | 133 | 170 | 205 | 140 | 174 | 182 | 224 | 226 | 227 | 226 | 255 | 255 | 255 |
| 16 | 14 | 13 | 15 | 23 | 69 | 235 | 255 | 254 | 199 | 145 | 186 | 246 | 255 | 255 | 255 | 246 | 246 | 255 | 255 | 255 | 250 | 255 | 255 |

**(B) DATABASE WITH ALL VALUES ARE IN RANGE [0, 1]**

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.09 | 0.09 | 0.10 | 0.09 | 0.11 | 0.12 | 0.11 | 0.09 | 0.11 | 0.19 | 0.24 | 0.40 | 0.56 | 0.71 | 0.73 | 0.65 | 0.53 | 0.58 | 0.95 | 1.00 | 1.00 | 0.87 | 0.97 | 1.00 |
| 0.27 | 0.27 | 0.19 | 0.21 | 0.20 | 0.20 | 0.15 | 0.13 | 0.10 | 0.05 | 0.07 | 0.24 | 0.27 | 0.43 | 0.56 | 0.70 | 0.76 | 0.95 | 0.90 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 0.29 | 0.28 | 0.25 | 0.24 | 0.16 | 0.17 | 0.22 | 0.22 | 0.22 | 0.23 | 0.19 | 0.22 | 0.28 | 0.38 | 0.87 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 0.11 | 0.12 | 0.13 | 0.14 | 0.13 | 0.14 | 0.14 | 0.15 | 0.07 | 0.15 | 0.29 | 0.38 | 0.56 | 0.68 | 0.78 | 0.69 | 0.62 | 0.62 | 0.88 | 1.00 | 1.00 | 1.00 | 0.95 | 1.00 |
| 0.16 | 0.21 | 0.20 | 0.24 | 0.17 | 0.21 | 0.20 | 0.21 | 0.21 | 0.19 | 0.21 | 0.26 | 0.35 | 0.45 | 0.55 | 0.71 | 0.75 | 0.73 | 0.87 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 0.40 | 0.37 | 0.33 | 0.30 | 0.29 | 0.25 | 0.23 | 0.30 | 0.41 | 0.52 | 0.59 | 0.64 | 0.67 | 0.69 | 0.69 | 0.68 | 0.67 | 0.69 | 0.66 | 0.63 | 0.62 | 0.61 | 0.61 | 0.63 |
| 0.02 | 0.02 | 0.02 | 0.02 | 0.04 | 0.03 | 0.02 | 0.01 | 0.00 | 0.01 | 0.04 | 0.09 | 0.11 | 0.09 | 0.07 | 0.07 | 0.12 | 0.10 | 0.15 | 0.21 | 0.18 | 0.09 | 0.19 | 0.27 |
| 0.23 | 0.22 | 0.16 | 0.18 | 0.18 | 0.19 | 0.15 | 0.09 | 0.07 | 0.09 | 0.09 | 0.17 | 0.25 | 0.29 | 0.31 | 0.31 | 0.36 | 0.39 | 0.59 | 0.69 | 0.79 | 0.71 | 0.48 | 0.23 |
| 0.16 | 0.17 | 0.19 | 0.23 | 0.20 | 0.12 | 0.11 | 0.11 | 0.10 | 0.19 | 0.24 | 0.35 | 0.52 | 0.63 | 0.64 | 0.61 | 0.65 | 0.68 | 0.68 | 0.71 | 0.55 | 0.25 | 0.27 | 0.65 |
| 0.27 | 0.27 | 0.25 | 0.18 | 0.14 | 0.11 | 0.11 | 0.12 | 0.13 | 0.25 | 0.43 | 0.55 | 0.72 | 0.75 | 0.73 | 0.70 | 0.67 | 0.65 | 0.64 | 0.64 | 0.64 | 0.55 | 0.85 | 0.50 |
| 0.22 | 0.18 | 0.17 | 0.19 | 0.22 | 0.22 | 0.17 | 0.13 | 0.13 | 0.17 | 0.24 | 0.28 | 0.38 | 0.38 | 0.40 | 0.56 | 0.67 | 0.63 | 0.85 | 0.99 | 1.00 | 1.00 | 1.00 | 1.00 |
| 0.11 | 0.12 | 0.11 | 0.08 | 0.08 | 0.07 | 0.07 | 0.07 | 0.10 | 0.13 | 0.20 | 0.22 | 0.46 | 0.61 | 0.76 | 0.78 | 0.74 | 0.70 | 0.74 | 0.85 | 0.78 | 0.93 | 0.95 | 1.00 |
| 0.07 | 0.06 | 0.05 | 0.05 | 0.04 | 0.05 | 0.06 | 0.07 | 0.16 | 0.34 | 0.35 | 0.44 | 0.46 | 0.31 | 0.29 | 0.40 | 0.49 | 0.56 | 0.60 | 0.44 | 1.00 | 1.00 | 0.93 | 1.00 |
| 0.09 | 0.07 | 0.06 | 0.07 | 0.06 | 0.10 | 0.24 | 0.23 | 0.36 | 0.44 | 0.50 | 0.38 | 0.31 | 0.37 | 0.56 | 0.54 | 0.60 | 0.53 | 1.00 | 1.00 | 1.00 | 0.98 | 1.00 | 0.98 |
| 0.09 | 0.12 | 0.15 | 0.14 | 0.11 | 0.08 | 0.06 | 0.06 | 0.07 | 0.10 | 0.33 | 0.52 | 0.67 | 0.80 | 0.55 | 0.68 | 0.71 | 0.88 | 0.89 | 0.89 | 0.89 | 1.00 | 1.00 | 1.00 |
| 0.06 | 0.05 | 0.05 | 0.06 | 0.09 | 0.27 | 0.92 | 1.00 | 1.00 | 0.78 | 0.57 | 0.73 | 0.96 | 1.00 | 1.00 | 1.00 | 0.96 | 0.96 | 1.00 | 1.00 | 1.00 | 0.98 | 1.00 | 1.00 |

**(C) DATABASE WITH SAMPLED IN 4 MEMBERSHIP FUNCTIONS**

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 160 | 160 | 160 | 160 | 160 | 160 | 224 | 224 | 224 | 224 | 224 | 224 | 224 |
| 96 | 96 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 96 | 160 | 160 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 |
| 96 | 96 | 96 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 96 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 |
| 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 96 | 160 | 160 | 224 | 160 | 160 | 160 | 224 | 224 | 224 | 224 | 224 | 224 |
| 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 96 | 96 | 160 | 160 | 160 | 160 | 224 | 224 | 224 | 224 | 224 | 224 |
| 96 | 96 | 96 | 96 | 96 | 96 | 32 | 96 | 96 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 160 |
| 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 |
| 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 96 | 96 | 96 | 96 | 96 | 160 | 160 | 224 | 160 | 96 | 32 | |
| 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 96 | 160 | |
| 96 | 96 | 96 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 96 | 160 | 160 | 224 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 160 | 224 | 96 |
| 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 96 | 96 | 96 | 160 | 160 | 224 | 224 | 224 | 224 | 224 | 224 | 224 |
| 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 160 | 224 | 224 | 160 | 160 | 160 | 224 | 224 | 224 | 224 | 224 |
| 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 96 | 96 | 96 | 96 | 96 | 96 | 96 | 160 | 160 | 96 | 224 | 224 | 224 | 224 |
| 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 96 | 96 | 96 | 96 | 96 | 160 | 160 | 160 | 160 | 224 | 224 | 224 | 224 | 224 | 224 |
| 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 32 | 96 | 160 | 160 | 160 | 160 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 |
| 32 | 32 | 32 | 32 | 32 | 96 | 224 | 224 | 224 | 224 | 160 | 160 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 | 224 |

**(D) DATABASE WITH ADDED GAUSSIAN NOISE**

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | 17 | 6 | 18 | 26 | 39 | 33 | 13 | 30 | 56 | 54 | 97 | 136 | 183 | 199 | 163 | 145 | 148 | 240 | 248 | 250 | 216 | 251 | 253 |
| 76 | 84 | 35 | 40 | 57 | 57 | 50 | 23 | 22 | 0 | 15 | 59 | 59 | 104 | 134 | 179 | 187 | 244 | 233 | 250 | 255 | 255 | 244 | 255 |
| 79 | 66 | 69 | 45 | 33 | 42 | 60 | 60 | 60 | 57 | 52 | 62 | 71 | 93 | 220 | 255 | 248 | 255 | 234 | 250 | 254 | 255 | 255 | 255 |
| 17 | 27 | 32 | 33 | 28 | 35 | 36 | 28 | 23 | 23 | 69 | 104 | 131 | 158 | 197 | 181 | 152 | 157 | 205 | 255 | 255 | 255 | 234 | 253 |
| 46 | 42 | 57 | 69 | 37 | 49 | 49 | 55 | 56 | 49 | 38 | 63 | 102 | 114 | 138 | 189 | 207 | 190 | 225 | 238 | 255 | 255 | 256 | 246 |
| 102 | 92 | 85 | 66 | 71 | 67 | 65 | 72 | 104 | 126 | 156 | 168 | 178 | 184 | 176 | 177 | 176 | 165 | 174 | 162 | 155 | 155 | 156 | 163 |
| 14 | 3 | 6 | 13 | 17 | 8 | 4 | 13 | 0 | 1 | 8 | 33 | 41 | 35 | 23 | 25 | 24 | 24 | 44 | 53 | 58 | 15 | 54 | 65 |
| 71 | 49 | 36 | 41 | 34 | 56 | 21 | 14 | 24 | 12 | 43 | 32 | 52 | 72 | 95 | 66 | 92 | 105 | 150 | 177 | 197 | 191 | 123 | 53 |
| 37 | 59 | 61 | 53 | 49 | 34 | 29 | 50 | 24 | 41 | 58 | 81 | 136 | 168 | 168 | 170 | 174 | 165 | 165 | 185 | 133 | 65 | 68 | 158 |
| 66 | 64 | 67 | 47 | 37 | 32 | 26 | 23 | 36 | 77 | 120 | 133 | 193 | 184 | 195 | 187 | 152 | 167 | 159 | 150 | 171 | 127 | 206 | 129 |
| 60 | 59 | 30 | 49 | 47 | 65 | 44 | 59 | 35 | 54 | 59 | 77 | 99 | 110 | 109 | 141 | 169 | 166 | 225 | 255 | 255 | 249 | 236 | 255 |
| 32 | 31 | 28 | 10 | 21 | 8 | 10 | 17 | 25 | 22 | 58 | 74 | 116 | 159 | 186 | 190 | 202 | 177 | 184 | 219 | 189 | 237 | 240 | 255 |
| 6 | 21 | 4 | 21 | 7 | 12 | 10 | 0 | 45 | 85 | 87 | 115 | 119 | 87 | 77 | 107 | 132 | 147 | 139 | 126 | 255 | 249 | 231 | 247 |
| 18 | 10 | 15 | 19 | 7 | 25 | 59 | 68 | 112 | 107 | 112 | 100 | 80 | 81 | 152 | 145 | 151 | 135 | 247 | 250 | 255 | 240 | 255 | 254 |
| 10 | 28 | 34 | 22 | 35 | 19 | 31 | 19 | 4 | 32 | 79 | 149 | 176 | 199 | 134 | 165 | 179 | 222 | 219 | 225 | 234 | 254 | 255 | 255 |
| 10 | 16 | 6 | 11 | 30 | 67 | 230 | 253 | 255 | 193 | 165 | 192 | 237 | 255 | 251 | 243 | 243 | 238 | 255 | 249 | 249 | 246 | 246 | 255 |

**Fig. 8.** Various image classes in pixel format.

it is of utmost importance to analyze the threshold for defining functions by which desired accuracy will be achieved. These four image classes present the unique scenarios, where the 'Default' class refers the maximum number of membership functions; the 'In range' class refers the fuzzy working range i.e., [0, 1] and kept al the membership functions within this range; the 'Membership values' class limits the number of membership function per fuzzy inference system; and the 'gaussian noise' class refers the addition of noise on the existing data and validates the proposed algorithm in case of noisy data.

In this paper, we added gaussian noise to the available YaleB dataset and generated a new set of data. MATLAB 'imnoise' command [22] is used to generate new data with gaussian noise with zero mean value and small variance. It will be worthwhile to reevaluate this algorithm in the future.

## 4. Validation and result analysis for various image classes

It is an accepted fact that conventional fuzzy logic holds the limitation of dimensionality i.e., large number of inputs and rules. This limitation restricts the application of conventional fuzzy towards large datasets such as datasets consisting of a large number of real images. Designing a conventional fuzzy logic system for large image datasets is quite involved. To overcome this limitation, hierarchical structure possesses the most viable option. For the validation purposes, a multi-level hierarchical structure has been considered. For aggregated hierarchical tree structure, each hierarchical level consists of a total of $2(N-1)$ fuzzy units, where $N$ is the number of hierarchical levels. The lowest or first level of hierarchical structure inherits the highest number of fuzzy units i.e., $2(N-1)$ fuzzy units. Similarly, the second level inherits $2(N-2)$ fuzzy units; the third level inherits $2(N-3)$ fuzzy units; and so forth and so on. On the other hand, the incremental hierarchical tree structure consists of $M$ hierarchical levels where each hierarchical level inherits only one fuzzy unit, and $M$ = Total number of input parameters/ number of samples per input segment. Example, for image size of [1,1024], the input parameters can be defined as 1024, and if each input segment consists of 16 input parameters, the total number of hierarchical levels = 1024/16 = 64. This is critical because all the hierarchical levels consume part of real inputs.

### 4.1. Verification of fuzzy weighted approach

Hierarchical fuzzy logic implementation resolves the issue of limitation of dimension by simplify and minimize total rule base. However, maintaining and retaining accuracy of the system are very crucial too. Fuzzy logic offers various aggregation methods, out of which 'maximum' and 'weighted average' are the most effective approaches. To make a system more reliable and robust, it is highly important to validate two approaches and identify the best suitable use-case.

For designing the low dimension hierarchical system, Radek Sindelar [11] presented the selection criteria of various aggregated methods and showcased that the two aggregated methods i.e., 'maximum' and 'weighted average' are most effective methods [3,4]. In this paper, the behavior of these two aggregation methods have been compared for hierarchical systems. Since a hierarchical fuzzy system comprises several fuzzy units at various levels of hierarchical structure, each unit has its own rule base that is extracted from required clusters of data. Keeping numerous fuzzy units for one system in mind, it is critical to select the effective aggregation method for defuzzification for every fuzzy unit. The selection of aggregation methods provides better and more accurate results.

Fig. 9 Shows the graphical representation for the comparison of two aggregation methods, i.e., weighted average and maximum, with respect to a conventional fuzzy system. Where 'sFIS' represents the conventional system in blue color and 'hFIS' represents the hierarchical fuzzy system in red color. Fig. 9 presents the maximum aggregation method comparison between conventional fuzzy and hierarchical fuzzy systems. The figure also presents the weighted aggregation method comparison between conventional fuzzy and hierarchical fuzzy systems.

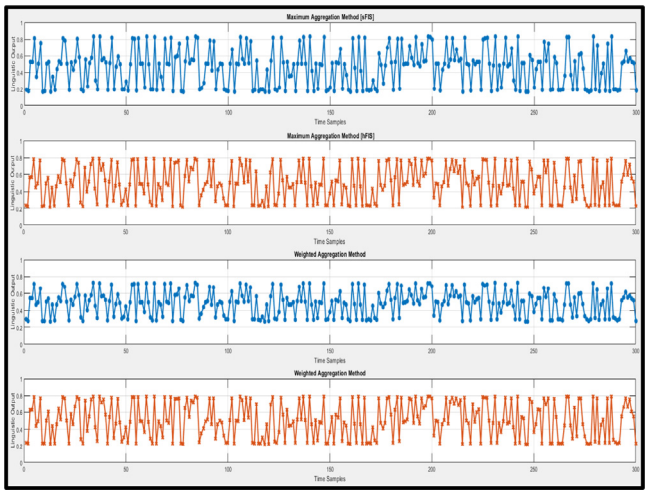



**Fig. 9.** Comparison between weighted approaches.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

It is observed that the aggregated method using weighted average is a clear winner with 0.92 as correlation factor with expected output and gives a closer result compared to maximum aggregate method with correlation factor of 0.84. Note: during the analysis for the above said situations, the rule base is considered the same.

### 4.2. Cumulative distribution

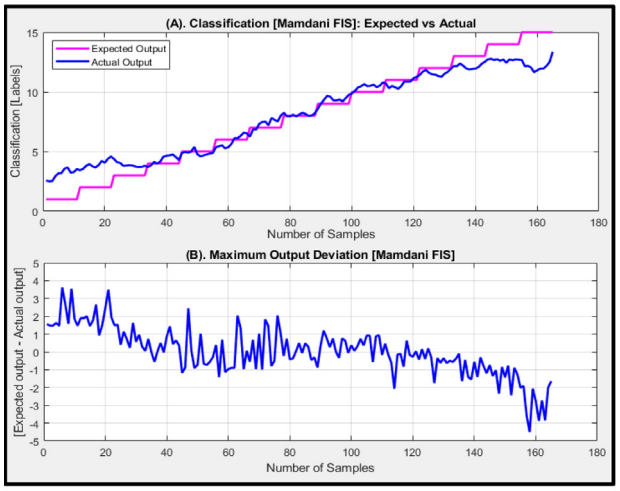
One of the objectives of this research is to design a system using algorithms with higher accuracy. The accuracy between expected output and actual output using various fuzzy inference system methodologies have been compared and shown in Fig. 10 below.

The design of aggregated hierarchical tree structure is considered using both Mamdani and Sugeno type fuzzy inference systems [3,4,8–10]. The analysis using two approaches provides critical information as to which fuzzy inference system type provides a result closer to the desired output. The descriptions of graphs are given as:

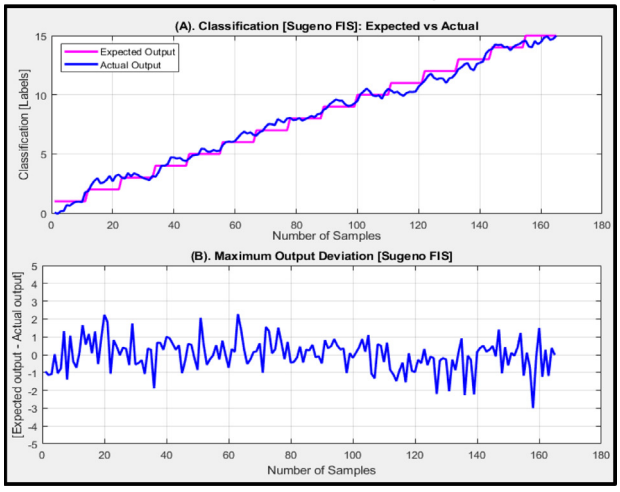
- Mamdani type fuzzy inference system
  - A. Image classification — Expected and Actual output
  - B. Output deviation
- Sugeno type fuzzy inference system
  - A. Image classification — Expected and Actual output
  - B. Output deviation

Fig. 10 represents the classification of images and the deviation between expected and actual output using both Mamdani type and Sugeno type inference systems. Where, Fig. 10(A) presents the analysis using Mamdani inference system and Fig. 10(B) presents the analysis using Sugeno inference system. The deviation is calculated by the relative difference between expected and actual output i.e., [expected output – actual output]. The closer the deviation is to '0', the higher the accuracy of the system will be. Where, the $x$-axis presents the number of image samples and the $y$-axis represents the image classification i.e., labels for the image data.

Please note that the testing dataset from YaleB is considered to validate the performance of the algorithm proposed. This analysis has been performed using aggregated hierarchical tree structure. Number of samples represents the number of images available in the testing dataset to validate. Testing image dataset has been sorted in ascending




(A) Classification and Deviation using Mamdani FIS




(B) Classification and Deviation using Sugeno FIS

**Fig. 10.** Deviation between actual and expected result.

order as per expected label i.e., lowest value (1) to highest value (15). After the sorting is done, the results are compared between expected and actual output. The 'smooth (default)' command [22] in the MATLAB is used on actual output to remove and handle any fluctuations and noise. The following observations have been made:

- Image classification: It is visible from two left plots that the Sugeno type fuzzy inference system offers much closer results to expected labels from YaleB database than then Mamdani type fuzzy inference system. The hierarchical system design using Sugeno approach is better
- Maximum output deviation: Mamdani type fuzzy inference system has higher deviation than Sugeno type fuzzy inference system. This shows that the output difference between Mamdani type fuzzy inference system and YaleB database for every sample is large compared to Sugeno type that provides low deviation. In the figure, the range of deviation through Mamdani approach is [−5, 4] whereas the range of deviation through Sugeno approach is [−3, 2]

Figs. 11–12 shows the empirical cumulative distribution function with respect to expected behavior. Graphs represent different classes with different hierarchical tree structures. Fig. 11 shows the empirical cumulative distribution with respect to expected behavior for aggregated hierarchical fuzzy tree structure for all the mentioned classes

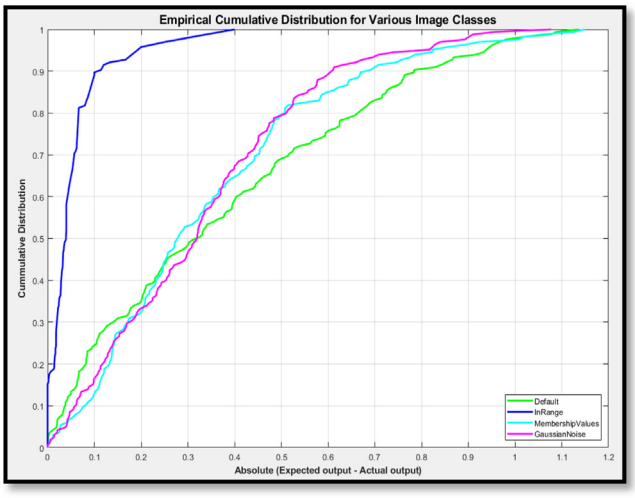



**Fig. 11.** Empirical Cumulative Distribution using aggregated hierarchical tree structure.

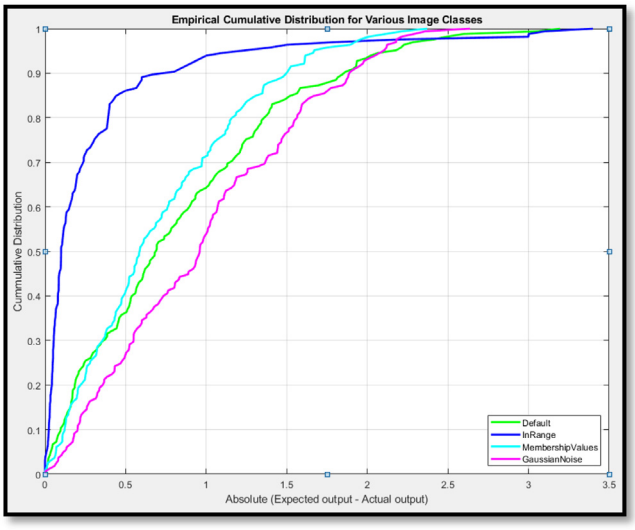



**Fig. 12.** Empirical Cumulative Distribution using incremental hierarchical tree structure.

of images. Fig. 12 shows the empirical cumulative distribution with respect to expected behavior for incremental hierarchical fuzzy tree structure for all the mentioned classes of images.

Closer the graph to $Y$-axis, better the performance of the system. It is observed that both incremental and aggregated hierarchical structure results are closer to the desired behavior.

### 4.3. Performance of hierarchical system algorithms

Figs. 13–14, shows the comparison between expected output versus actual output in raw form.

Fig. 13 shows the overall performance for expected and actual values using aggregated hierarchical fuzzy structure for all the image classes.

Fig. 14 shows the overall performance for expected and actual values using incremental hierarchical fuzzy structure for all the image classes included in this paper.

As stated in the previous section, the YaleB testing dataset has been leveraged for the validation of the accuracy. The YaleB testing data is random in nature and consists of all the image classes included in the YaleB training dataset. Any image samples with varied image labels

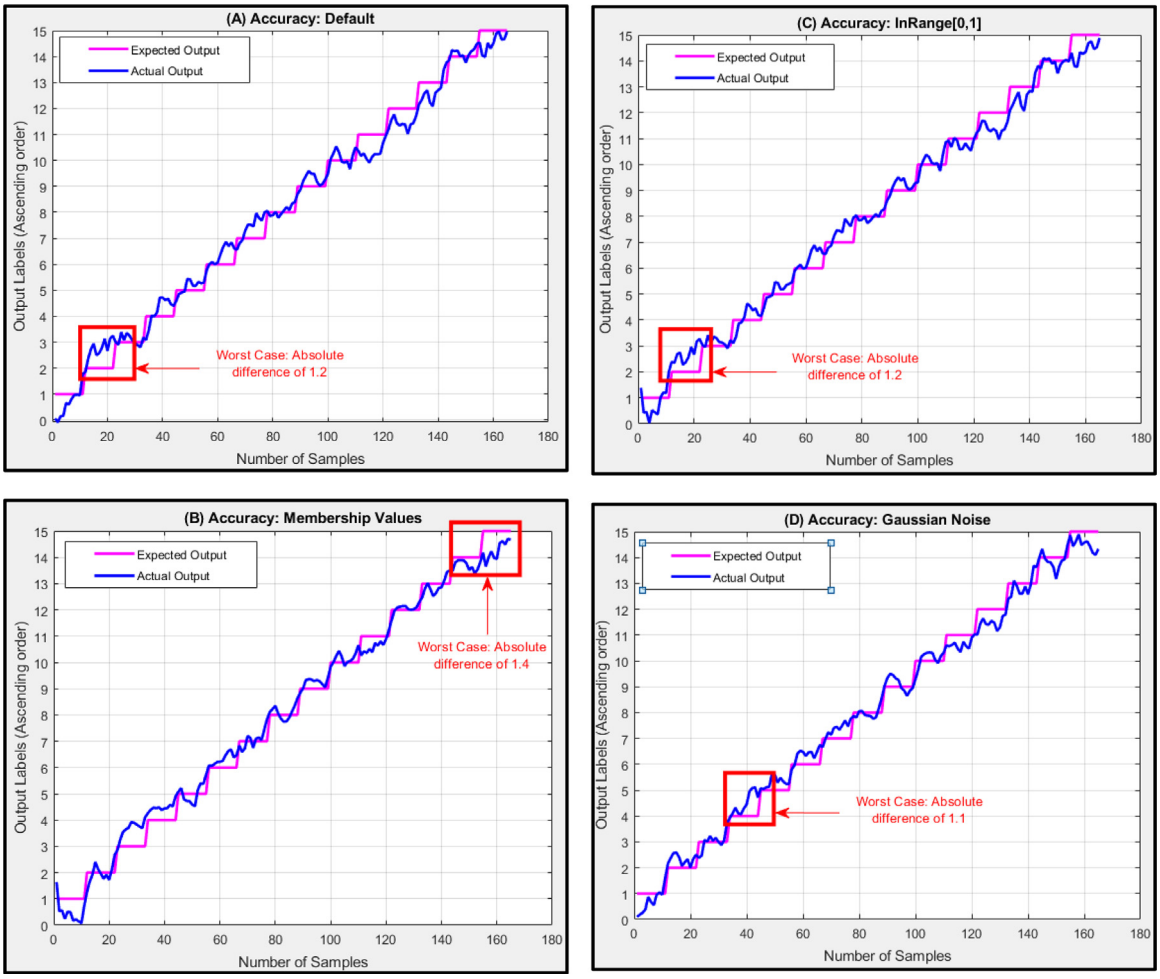



**Fig. 13.** Accuracy for aggregated tree structure.

(i.e., from 1 to 15) can be taken for further validation of this proposed algorithm. For visual purposes, the dataset has been sorted in ascending order as per expected output label and then expanded the selection to the actual output. Every number on the $x$-axis presents the unique image with expected and corresponding actual output.

The performance of the system has been shown in the graphs by observing the behavioral response of actual output with respect to expected output. By comparing all the hierarchical tree structure results, aggregated structure results are better than the incremental hierarchical tree structure. Among various image classes, the aggregated hierarchical tree structure provides the worst-case scenario with the absolute difference between expected output and actual output in range of [1.1, 1.4] whereas, the incremental hierarchical tree structure provides the worst-case scenario with the absolute difference between expected output and actual output in range of [2.2, 4.5].

The input ranking for the incremental hierarchical tree structure is based on randomization. It will be worthwhile to expand this research and add the new ranking and prioritization system for such cases. It is expected that with correct input rankings, the results for incremental hierarchical tree structure can be improved significantly.

### 4.4. Behavioral analysis

Tables 1–2 represents the behavioral analysis of expected and actual behavior for all the image classes with following parameters in MATLAB environment [22]:

- **Variance:** Computes the consistency or dispersion of data over specified vector dimension
- **Standard Deviation (SD):** Measure amount of variation or dispersion within dataset
- **Median:** Value separating data sample in two halves
- **Mean:** Average of all the values in dataset
- **Skewness:** Measure of asymmetry of the probability distribution of real data around its mean value
- **Interquartile Range (IQR):** Measure of statistical dispersion equally distributed between 25th and 75th percentile. It is the measure of variability depending on the division of data in quartiles
- **Kurtosis:** Peak value of a frequency distribution

Where, "Exp" = expected behavior; "Act 1" = actual behavior for default image class; "Act 2" = actual behavior for in-range [0,1] image class, "Act 3" = actual behavior for membership function image class; and "Act 4" = actual behavior for gaussian image class. All the results shown are in reference to desired behavior.

We designed the proposed algorithm using both incremental tree structure and aggregated tree structure separately. Keeping this in mind, the results shown in Tables 1 and 2 are shown separately. However, the image dataset input to the model is the same for both hierarchical structures.

The significance of this analysis is to perform the quality check for the algorithm and do the static analysis to ensure the actual results are

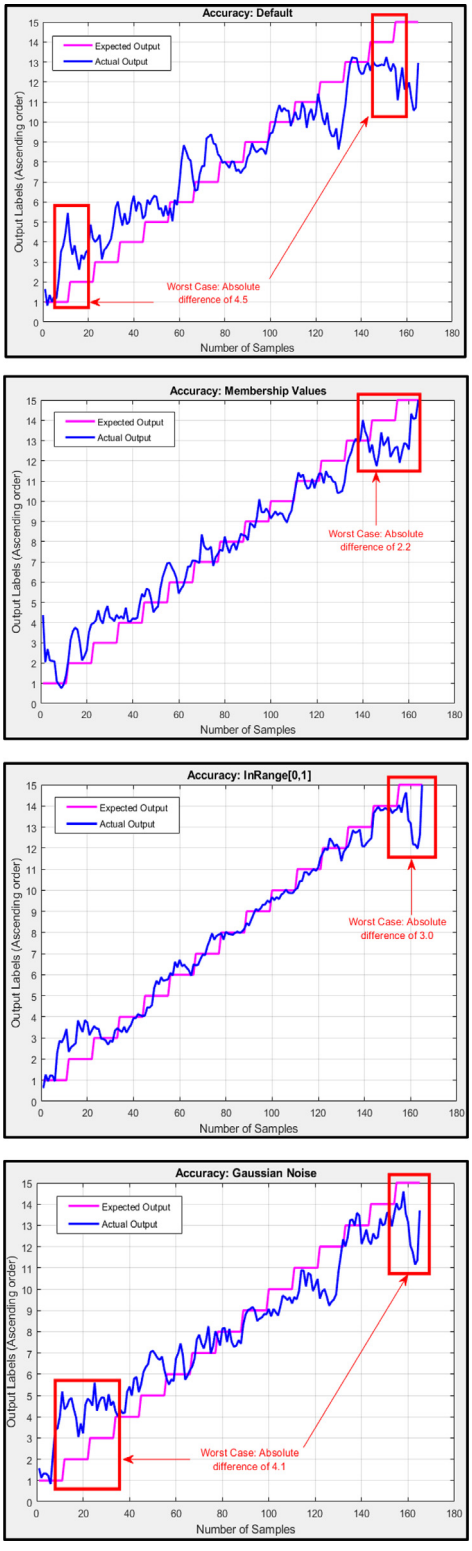



**Fig. 14.** Accuracy for incremental tree structure.

within a certain threshold compared to expected results. This analysis provides internal behavior.

Since all the inputs processed at the lower level in the aggregated hierarchical tree structure, the other higher hierarchical levels hold the degree of symmetry. This analysis can be seen with parameters such as skewness which is '0' for aggregated structure. However, for incremental hierarchical tree structure, the hierarchical levels do not hold the degree of symmetry because every hierarchical level consumes real-input data, which is why variation in skewness can be observed.

**Table 1**
Behavioral analysis of various image classes using aggregated hierarchical structure.

| Parameters | Exp | Act 1 | Act 2 | Act 3 | Act 4 |
|---|---|---|---|---|---|
| Variance | 18.78 | 18.37 | 17.8 | 16.67 | 18.03 |
| SD | 4.33 | 4.33 | 4.33 | 4.33 | 4.33 |
| Median | 8 | 8 | 8 | 8 | 8 |
| Mean | 8 | 8 | 8 | 8 | 8 |
| Skewness | 0 | 0 | 0 | 0 | 0 |
| IQR | 8 | 8 | 8 | 8 | 8 |
| Kurtosis | 1.79 | 1.79 | 1.79 | 1.79 | 1.79 |

**Table 2**
Behavioral analysis of various image classes using incremental hierarchical structure.

| Parameters | Exp | Act 1 | Act 2 | Act 3 | Act 4 |
|---|---|---|---|---|---|
| Variance | 3.60 | 3.78 | 3.84 | 3.89 | 3.09 |
| SD | 1.90 | 1.90 | 2.58 | 1.61 | 1.78 |
| Median | 7.44 | 7.44 | 8 | 8 | 7.6 |
| Mean | 7.05 | 7.05 | 7.8 | 7.66 | 6.93 |
| Skewness | -0.76 | -0.76 | 0.44 | -0.95 | -0.95 |
| IQR | 2.86 | 2.86 | 6.16 | 0.47 | 2.61 |
| Kurtosis | 3.12 | 3.12 | 2.77 | 4.6 | 3.28 |

### 4.5. Comparison

There are several open-source libraries available for various deep learning methods including Convolution neural network (CNN) [9,10,25]. One of the Mathworks toolbox i.e., MatConvNet is considered in this paper. The MatConvNet toolbox is mainly used for computer vision applications. The Convolution neural network comprises both linear and non-linear filtration operations i.e., rectification, convolution, normalization, and pooling. In MatConvNet toolbox, the CNN [25] combines two networks: Simple neural networks (Simple NN) that is relevant for linear network topology, for example, linear sequence of computational blocks; and directed acyclic graph neural network (Diag. NN) that is alternate to Simple NN and allows the network with directed acyclic graph topology. One of the major benefits of using MatConvNet toolbox is the ability to present an efficient platform to several researchers. The YaleB database using CNN model [10,25] has been considered to see the performance of the algorithm. In this paper, standard CNN architecture i.e., 'AlexNet' is considered to train complex models for large image dataset. This architecture allows fast prototyping and efficient CPU and GPU computation.

Table 3 presents the comparison between the output gathered from YaleB CNN model trained via MatConvNet and the output from the hierarchical fuzzy system using aggregated and incremental hierarchical tree structure. The comparison has been shown in a percentage that represents the number of all the cases that lie within 10% of error tolerance. The error tolerance is defined as the difference between the expected output and actual output from the system. Higher the number within 10% of error tolerance, higher the accuracy of the system. For better performance, it is necessary to calibrate the system to the minimum number of clusters where desired accuracy can be met. It is also observed that as the number of clusters is reduced by half, the accuracy diminishes as well.

For effective system design, apart from accuracy of the system, the computational load is an equally important parameter. Table 4 shows the computational load in time [seconds] comparison among

**Table 3**
Accuracy between CNN model and hierarchical fuzzy systems using various tree structures.

| Models and Methodologies | Image Class | Total Clusters per Fuzzy unit | Outputs spread within 10% error tolerance |
|---|---|---|---|
| Outputs from CNN Model computed via MatConvNet Toolbox | All classes | ~ | 91.2% |
| Outputs from hierarchical fuzzy system using aggregated tree structure | Default | 100 | 90% |
| | | 50 | 77% |
| | In Range [0,1] | 100 | 91% |
| | | 50 | 82% |
| | | 100 | 88% |
| | Membership Values | 50 | 78% |
| | Gaussian Noise | 100 | 82% |
| | | 50 | 74% |
| Outputs from hierarchical fuzzy system using incremental tree structure | Default | 100 | 73% |
| | | 50 | 74% |
| | In Range [0,1] | 100 | 74% |
| | | 50 | 83% |
| | Membership Values | 100 | 85% |
| | | 50 | 78% |
| | Gaussian Noise | 100 | 70% |
| | | 50 | 73% |

**Table 4**
Computation time between CNN model and hierarchical fuzzy systems using various tree structures.

| Models and Methodologies | Image Class | Clusters per unit | Computation Time in Seconds |
|---|---|---|---|
| CNN Model via MatConvNet Toolbox | All classes | ~ | 38.21 |
| Hierarchical fuzzy system using aggregated tree structure | Default | 100 | 30.15 |
| | | 50 | 27.76 |
| | In Range [0,1] | 100 | 26.65 |
| | | 50 | 26.89 |
| | Membership Values | 100 | 39.45 |
| | | 50 | 28.12 |
| | Gaussian Noise | 100 | 30.31 |
| | | 50 | 25.81 |
| Hierarchical fuzzy system using incremental tree structure | Default | 100 | 33.28 |
| | | 50 | 29.61 |
| | In Range [0,1] | 100 | 25.67 |
| | | 50 | 28.04 |
| | Membership Values | 100 | 33.50 |
| | | 50 | 30.02 |
| | Gaussian Noise | 100 | 34.06 |
| | | 50 | 29.91 |

CNN and various hierarchical tree structures presented in this paper. Please note that the computation time for hierarchical tree structures can further be reduced or enhanced by calibrating the total number of fuzzy clusters required to design fuzzy units. It is visible in the table that reducing clusters by half reduces the computational time. Depending on the hardware limitations, this number can be adjusted to optimal performance.

From these results, following conclusions can be drawn:

➢ For aggregated hierarchical tree structure, In Range image class possesses a best case with 91% accuracy whereas Gaussian Noise image class provides the worst case with 82%.

**Table 5**
Inference time for hierarchical tree structures.
(A) Inference Time for Aggregated Tree Structure.

| AGGREGATED HIERARCHICAL STRUCTURE | | | |
|---|---|---|---|
| Segments of real inputs | Total inputs per fuzzy unit at lower level | Total fuzzy units (n = levels required to reach to highest level) | Inference Time in Seconds |
| 8 | 128 | $\sum_{1}^{4} 2^{n-1} = 15$ | 1.88 |
| 16 | 64 | $\sum_{1}^{5} 2^{n-1} = 31$ | 1.72 |
| 32 | 32 | $\sum_{1}^{6} 2^{n-1} = 63$ | 1.98 |
| 64 | 16 | $\sum_{1}^{7} 2^{n-1} = 127$ | 3.03 |

(B) Inference Time for Incremental Tree Structure.

| AGGREGATED HIERARCHICAL STRUCTURE | | | |
|---|---|---|---|
| Segments of real inputs | Total inputs per fuzzy unit at lower level | Total fuzzy units (Assume each level has one fuzzy unit) | Inference Time in Seconds |
| 8 | 128 | 8 | 1.78 |
| 16 | 64 | 16 | 1.52 |
| 32 | 32 | 32 | 1.55 |
| 64 | 16 | 64 | 2.15 |

➢ For incremental hierarchical tree structure, Membership values image class possess a best case with 85% accuracy whereas Gaussian Noise image class provides the worst case with 70% accuracy.

➢ For aggregated hierarchical tree structure, the computational time is generally better than convolution neural network

➢ For incremental hierarchical tree structure, the computational time is higher than convolution neural network

➢ Aggregated hierarchical tree structure for In Range image class provides the same accuracy as the convolution neural network. However, the computational time for In Range class is significantly lower than the convolution neural network

➢ Incremental hierarchical tree structure for Membership values image class provides accuracy less than the convolution neural network. However, the computational time for the membership values class is significantly higher than the convolution neural network.

It is observed from Table 3 that the accuracy reduces with a reduced number of clusters. Per our analysis, to obtain higher accuracy, the number of clusters must be higher. On the contrary, it is observed from Table 4 that computational time is higher with higher number of clusters and vice versa. This is up to the user's discretion to ensure the tradeoff between accuracy and computation time.

For the evaluation purposes, the size of the input segments at various levels of the hierarchical tree structures plays a critical role in analyzing the inference time. For aggregated structure all the real inputs consumed at the lower level of the hierarchy and thus the input segment size has the impact only at the lower level of the hierarchical structure. For incremental structure, the size of the input segment impacts all the levels of the hierarchical structure, as all the levels consume part of real inputs. Considering the hierarchical system as

a black box, the inference time is the time taken by the hierarchical system to process inputs at various levels and provide final output.

Table 5 shows the analysis in the form of inference time with respect to the input segments. Assuming the image dataset with each image of size 32 × 32 i.e., a matrix of [1,1024]. In this paper, for designing the aggregated hierarchical tree structure, the total of $2^{n-1}$ fuzzy units has been considered at the lowest level and all the levels above consumes outputs from two fuzzy units at lower level i.e., $2^{n-2}$ fuzzy units required in the next level. Similarly for designing the incremental tree structure, only one fuzzy unit is considered for each hierarchical level.

The hierarchical structure is designed in reference to the number of segments or how many fuzzy units are required to consume all the segments. Considering four layer aggregated hierarchical tree structure, the segments of real inputs determine the number of maximum real inputs for each fuzzy unit at the lowest level. Assuming every fuzzy unit in the next layer acquires outputs from two previous fuzzy units in previous layers, this sums up to the total number of hierarchical fuzzy units. The inference time is defined in reference to the total number of fuzzy logic units and number of inputs per fuzzy unit.

It is observed from Table 5 that the size selection for input segments requires an iteration approach. Too less or too high size leads to the situation of less accuracy or more computation time. There are numerous ways to design any specific hierarchical tree structure. For e.g., the aggregated tree structure can be designed with a random number of fuzzy units at any hierarchical level. The key point is that all the inputs shall be consumed at the lowest level. It is important to consider the number of fuzzy units. Depending on the system capabilities, if the number of fuzzy units are too high, the size of the input segments will be adjusted. It is observed that the inference time for the lowest level of aggregated tree structure and for all the levels of incremental tree structure is similar.

The YaleB database consists of finite and precise image datasets. It is expected that with more uncertain, imprecise, and vague datasets such as real-life human assessments, the proposed hierarchical fuzzy algorithm design will overcome various existing models available in the literature and will provide more accurate outcomes.

## 5. Conclusion

The limitation of dimensionality in conventional fuzzy systems leads to the motivation for the development of hierarchical fuzzy structure. The real-life applications, with large amounts of data or images, leads to an increasing number of rules, parameters and increasing input variables. Therefore, there is always a necessity for a robust, and practical solution that will work in highly complex and complicated issues irrespective of their dimensions. The results have shown that hierarchical fuzzy systems are a very effective and viable option to overcome a lot of limitations of conventional fuzzy and fuzzy neural networks. Despite significant achievement, there are various open points such as handling of intermediate variables between two levels of hierarchical fuzzy structure and, determining few environments where other hierarchical structures such as incremental and cascaded will be more effective. Further investigation including detailed research could help address more complicated and more complex problems with both low and high dimensions. The database taken is for facial recognition, the approach is applicable for other large datasets such as video data, audio data, MNIST, CIFAR, etc. It is hoped that the approach suggested here will be exploited in future for the classification of images for a variety of applications.

## Declaration of competing interest

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

## Data availability

No data was used for the research described in the article.

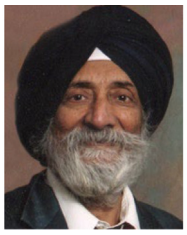
## Acknowledgments

We would like to thank all the reviewers for their thorough and extremely helpful comments in improving the paper.

## References

[1] P.W. Simões, B.I. Narjara, S.C. Ramon, V. Ramon, D.V. Carlos, P.M. Gustavo, L. Edroaldo, et al., Classification of images acquired with colposcopy using artificial neural networks, Cancer Inform. 13 (2014) CIN-S17948.
[2] F.L. Chung, J.C. Duan, On multistage fuzzy neural network modeling, IEEE Trans. Fuzzy Syst. 8 (2) (2000) 125–142.
[3] L.X. Wang, Fuzzy systems are universal approximators, in: Proceedings of Conference on Fuzzy Systems, San Diego, 1992, pp. 1163–1170.
[4] L.A. Zadeh, Fuzzy sets, Inf. Control 8 (3) (1965) 338–353.
[5] S. Kamthan, H. Singh, T. Meitzler, Survivability: a hierarchical fuzzy logic layered model for threat management of unmanned ground vehicles, in: Autonomous Systems: Sensors, Vehicles, Security, and the Internet of Everything, Vol. 10643, International society of optics and photonics, 2018, 106430W.
[6] S. Kamthan, H. Singh, T. Meitzler, UAVs: on development of fuzzy model for categorization of countermeasures during threat assessment, in: Unmanned Systems Technology XIX, Col. 10195, International society for optics and photonics, 2017, 1019518.
[7] H. Singh, M.M. Gupta, T. Meitzler, Z.G. Hou, K.K. Garg, A.M. Solo, L.A. Zadeh, Real-life applications of fuzzy logic, Adv. Fuzzy Syst. (2013) 581879-1. APA.
[8] S. Kamthan, H. Singh, Hierarchical fuzzy logic systems, J. Inst. Eng. (India), Series B 103 (2022) 1167–1175.
[9] S. Kamthan, Hierarchical Intelligent Systems and their Applications To Survivability, (Dissertations), Vol. 3419, Wayne State University, 2021.
[10] S. Kamthan, H. Singh, Hierarchical fuzzy logic for multi-input multi-output systems, IEEE Access 8 (2020) 206966–206981.
[11] R. Sindelar, Hierarchical fuzzy systems, IFAC Proc. Vol. 38 (1) (2005) 245–250.
[12] R. Scherer, L. Rutkowski, A survey of hierarchical fuzzy systems, in: Proceedings of the Fifth Conference Neural Networks and Soft Computing, Zakopane, 2000, pp. 6–10.
[13] G.V.S. Raju, J. Zhou, Adaptive hierarchical fuzzy controller, IEEE Trans. Syst. Man Cybern. 23 (4) (1993) 973–980.
[14] S. Oprea, et al., A review on deep learning techniques for video prediction, IEEE Trans. Pattern Anal. Mach. Intell. 44 (6) (2022) 2806–2826.
[15] M. Xu, C. Li, S. Zhang, P. Le Callet, State-of-the-art in 360 video/image processing: Perception, assessment and compression, IEEE J. Sel. Top. Sign. Proces. 14 (1) (2020) 5–26.
[16] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, Image and video compression with neural networks: A review, IEEE Trans. Circuits Syst. Video Technol. 30 (6) (2019) 1683–1698.
[17] Ralph Gross, Brajovic Vladimir, An image preprocessing algorithm for illumination invariant face recognition, in: International Conference on Audio-and Video-Based Biometric Person Authentication, Springer, Berlin, Heidelberg, 2003, pp. 10–18.
[18] M.L. Lee, H.Y. Chung, F.M. Yu, Modeling of hierarchical fuzzy systems, Fuzzy Sets and Systems 138 (2003) 343–361.
[19] L. Sun, W. Huo, Adaptive fuzzy control of spacecraft proximity operations using hierarchical fuzzy systems, IEEE/ASME Trans. Mechatronics 21 (3) (2016) 1629–1640.
[20] S. Minaee, Y.Y. Boykov, F. Porikli, A.J. Plaza, N. Kehtarnavaz, D. Terzopoulos, Image segmentation using deep learning: A survey, IEEE Trans. Pattern Anal. Mach. Intell. (2021).
[21] S. Kamthan, H. Singh, T. Meitzler, Hierarchical Fuzzy Deep Learning for Image Classification, ELSEVIER, Memories - Materials, Devices, Circuits and Systems, 2021, Accepted, In Publication.
[22] Mathworks Inc, Wei-cheng Wang, Fuzzy Logic Toolbox: For Use with MATLAB: User's Guide, Mathworks Inc., 1998.
[23] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.
[24] Srinivas Gutta, Wechsler Harry, Phillips P. Jonathon, Gender and ethnic classification of face images, in: Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition, IEEE, 1998, pp. 194–199.
[25] M.A. Albahar, Skin lesion classification using convolutional neural network with novel regularizer, IEEE Access 7 (2019) 38306–38313.

**Shashank Kamthan** received Bachelor of Engineering from Rajiv Gandhi Proudyogiki Vishwavidhyalaya, India, Master of Science and Ph.D. in Electrical Engineering from Wayne State University, Detroit, MI, USA. He has more than 12 publications in international journals and conferences. He has over 12 years of extensive experience in design and development of embedded systems and connected vehicle systems. During his stint with the automotive industry, he has worked in the field of model-based development, electrical architecture, connected vehicle systems, embedded software development, autonomous vehicles, artificial intelligence, deep learning etc. At present, he is working full-time as Senior Researcher/Subject Matter Expert at Ford Motor Company. His current areas of interest are target detection, target acquisition, artificial intelligence, connected vehicles, deep learning techniques, autonomous vehicles, classification, mobility, controls, model-based design, etc.



**Harpreet Singh** (M'73, SM'85, LSM'2009) received his B.Sc. Eng. degree from Punjabi University, India, in 1963 and his M.S. and Ph.D. degrees from the University of Roorkee, India, in 1966 and 1971. He taught at the University of Roorkee from 1963 to 1981. Since 1981 he has been with Wayne State University, Detroit, MI, where he has been a professor in the Department of Electrical and Computer Engineering. He has worked in diversified areas of systems, networks, controls, fuzzy logic, DNA Nano-digital circuits, biocircuits, and VLSI design. Dr. Singh has more than 300 publications in international journals and conferences and has received several awards. He has organized several national and international conferences and has supervised more than 25 Ph.D. theses in different areas of electrical and computer engineering. He has been awarded the distinguished Alumni Award of prestigious IIT Roorkee in 2012. He has also been awarded the distinguished alumni award from President of India in the year 1983 from Thapar Engineering College, Patiala. As a founder chair of Life Affinity Group of Southeastern Michigan section, the section received 2017 IEEE Life Member Affinity Group Achievement award.