Deep Learning and Interpolation for Featured-Based Pattern Classification

Yongfeng Zhang

Supervisors: Dr. Changjing Shang
Prof. Qiang Shen

Ph.D. Thesis

Department of Computer Science
Institute of Mathematics, Physics and Computer Science
Aberystwyth University

October 25, 2016

Declaration and Statement

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed Yongfeng Zhang (candidate)

Date 25/10/2016

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated. Where **correction services**[1] have been used, the extent and nature of the correction is clearly marked in a footnote(s).

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed Yongfeng Zhang (candidate)

Date 25/10/2016

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed Yongfeng Zhang (candidate)

Date 25/10/2016

[1]This refers to the extent to which the text has been corrected by others.

Abstract

Deep machine learning has received significant attention over the past decade, especially in terms of dealing with information that may span large scales. By employing a hierarchical architecture, consisting of simple computational nodes of similar characteristic, such a network helps to partition large data structures into relatively smaller, more manageable units, and to discover any dependencies that may exist between the resulting units. However, the process of running this type of network which has a layered structure, to perform tasks such as feature extraction, and subsequent feature pattern-based recognition, typically involves significant computation. To tackle this problem, two approaches are proposed in this thesis. The first novel approach developed is for image classification, by integrating deep learning and feature interpolation, supported with advanced learning classification techniques. The recently introduced Deep Spatio-Temporal Inference Network (DeSTIN) is employed to carry out limited original feature extraction. Simple interpolation is then employed to artificially increase the dimensionality of the extracted feature sets for accurate classification, without incurring heavy computational cost. The work is tested against the popular MNIST dataset of handwritten digits, demonstrating the potential of the proposed work. The second approach, which is a substantially simplified 2-layer learning network, is introduced that exploits unsupervised learning for pattern representation, capable of extracting effective features efficiently. Experimental results, in comparison with the use of popular deep learning networks, again on the application to handwritten digit classification demonstrate that the proposed approach is of significant potential in dealing with real-world problems.

The generation of effective feature pattern-based classification rules from data is essential to the development of intelligent classifiers which are readily comprehensible to the user. Unfortunately, a sparse rule base may be generated when there is missing information in the experienced dataset. This hinders classification systems that work based on such sparse knowledge effectively performing their tasks in many real-world applications, where complete historical data cannot be assumed. This thesis further proposes an innovative approach by integrating fuzzy rule interpolation within a data-driven classification mechanism, such that conclusions can be approximately derived even if no matched rule can be found from a given sparse rule base when given a certain observation. The proposed technique is simple conceptually, directly

exploiting the recently developed fuzzy rule interpolation techniques. However, the resulting integrated system offers a powerful means to develop robust classifiers, significantly enhancing the effectiveness of intelligent classification systems, as demonstrated by systematic comparative experimental results and also, by an application to the challenging problem of mammographic risk analysis.

Acknowledgements

I would like to express my uppermost gratitude to my supervisors: Dr. Changjing Shang and Prof. Qiang Shen, for their motivation, encouragement, and guidance, which have been essential at all stages of my research.

I would like to express my deepest appreciation to Aberystwyth University and Chinese Scholarship Counsel for their generous financial support.

I am also very thankful to Dr. Jingping Song, Dr. Pan Su, Dr. Yitian Zhao, Dr. Minfeng Huang and Dr. Chen Gui for their constant support.

I would like to thank all my colleagues in the Department of Computer Science. I am especially grateful to Dr. Neil S. Mac Parthaláin, Dr. Richard Jensen, Dr. Ren Diao, Dr. Chengyuan Chen, Dr. Nitin Kumar Naik, Dr. Shangzhu Jin, Dr. Lu Lou, Ling Zheng, Tianhua Chen, Fangyi Li and Zhenpeng Li. I have enjoyed many useful and entertaining discussions with them.

I am extremely grateful to Shangxian United, Chenyu Liu, Yue Ling, Xianyao Ge, Ziqiang Zhong, Ling Qin, Nanlong Shi, Yuleng Zeng, Xiangyu Yu and Qi Huang for their continuous support.

I am grateful to Lichao Sun, Chao Dong, Yang Jiang, Yi Liu, Anxun Sun, Jiajun Tang, Defeng Jiang.

My sincere gratitude goes to my entire family: my parents Zhenfa Zhang and Yunfeng Li, my sister Na Zhang and her husband Gang Li, my lovely nephew Xianghong Li.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Data-Driven Learning

The human can efficiently and robustly represent information, so mimicking this process has been a core challenge in artificial intelligence research for decades. Humans are exposed to myriad of sensory data received every second of the day and are somehow able to capture critical aspects of this data in a way that allows for its future use in a concise manner. Over 50 years ago, Richard Bellman, introduced dynamic programming theory [1] and pioneered the field of optimal control, asserted that high dimensionality of data is a fundamental hurdle in many science and engineering applications.

The main difficulty that arises, particularly in the context of pattern classification applications, is that the learning complexity grows exponentially with a linear increase in the dimensionality of the data, this phenomenon is known as the curse of dimensionality. The mainstream approach of overcoming "the curse" has been to pre-process the data in a manner that would reduce its dimensionality to that which can be effectively processed, for example by a classification engine. One dimensionality reduction scheme often referred to is feature extraction. It can be argued that the intelligence behind many pattern recognition systems has shifted to the human-engineered feature extraction process, which at times can be challenging and highly application-dependent. Moreover, if incomplete or erroneous features are extracted, the classification process is inherently limited in performance.

Recent neuroscience findings have provided insight into the principles governing information representation in the mammalian brain, leading to ideas for designing systems that represent information. One of the key findings has been that the neocortex, which is associated with many cognitive abilities, does not explicitly pre-process sensory signals, but rather allows them to propagate through a complex hierarchy of modules that, over time, learn to represent observations based on the regularities they exhibit. This discovery motivated the emergence of the sub-field of deep machine learning, which focuses on computational models for information representation that exhibit similar characteristics to that of the neocortex.

A number of influential and successful deep learning models have been proposed, including Deep Belief Networks (DBNs) [2], Stacked Autoencoders (SAEs) [3], Convolutional Neural Nets (CNNs) [4], and Deep Spatio-Temporal Inference Network (DeSTIN) [5]. Unfortunately, the process of running this type of network for feature extraction typically involves information processing and passing through a good number of layers, demanding significant computational effort. For example, as a representative application of DeSTIN, the existing work for handwritten digit recognition employs a network of 4 layers [6]. This may introduce considerable overheads on computation and therefore, may offset the potential benefit on the efficiency gained by the entire feature extraction process. An alternative approach is desirable.

1.2 Fuzzy Rule-Based Classification

1.2.1 Fuzzy Inference Systems

A fuzzy inference system (FIS) is a way of formulating the mapping from given inputs to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made, or patterns can be discerned. The concept of FISs is based on fuzzy logic, fuzzy IF-THEN rules [7] and fuzzy reasoning, which jointly enable modelling complex systems in a way naturally used by humans [8]. The general architecture of an FIS is well-known in the literature [9, 10, 11], consisting of four conceptual components: *fuzzifier*, *rule base*, *inference engine* and *defuzzifier*.

- *fuzzifier*: Converting the crisp input to a linguistic variable using the membership functions stored in the fuzzy knowledge base.

- *rule base*: Containing a selection of fuzzy IF-THEN rules.
- *inference engine*: Performing the inference procedure upon the rules and given facts to derive an inferred output or conclusion.
- *defuzzifier*: Converting the fuzzy output of the inference engine to a crisp value using membership functions analogous to the ones used by the *fuzzifier*.

With crisp inputs and output, an FIS implements a non-linear mapping from its inputs space to output space. This mapping is achieved by a number of fuzzy IF-THEN rules, each of which describes the local behaviour of the mapping. In particular, the antecedent of a rule represents a fuzzy region in the input space, while the consequence indicates the inferred consequent in the output region.

There are two ways to construct a fuzzy rule base for a given problem. The first class of FISs directly translates expert knowledge to fuzzy rules, so that these FISs are called fuzzy expert systems or fuzzy controllers [12, 13, 14]. Since rules are fuzzy representations of expert knowledge, these FISs may offer a high semantic level and a good generalisation capability. However, the complexity of large real-world problems may lead to an insufficient accuracy in the solutions found. Such drawback leads to the other class of FISs, which is a data-driven fuzzy system. The fuzzy rules are obtained from data by machine learning techniques rather than expert knowledge [15, 16, 17, 18].

1.2.2 Fuzzy Rule Interpolation

Given a fuzzy rule base generated in either of the above two ways, there are a number of fuzzy inference mechanisms, such as compositional rule of inference [19] and similarity-based fuzzy reasoning [20, 21, 22, 23, 24], that can be utilised for deriving a conclusion from a given observation. However, dense rule bases are compulsory for these methods. Briefly, a dense rule base is a rule base where the input universe of discourse is covered completely. Given such a rule base and an observation that is at least partially covered by the rule base, the conclusion can be inferred from certain rules that intersect with the observation. However, for the case where a fuzzy rule base (termed: sparse rule base [25]) contains "gaps", if a given observation has no overlap with the antecedent values of any rule, conventional fuzzy inference methods cannot derive a conclusion. A system implemented with such an incomplete

rule base is hereafter referred to as a sparse rule based system. Obviously, classical fuzzy reasoning methods can no longer be applied in certain cases due to the fact that a traditional rule-based inference mechanism will fail when no fuzzy rule may be found to match the given observation. This becomes a major drawback from the viewpoint of using fuzzy systems in solving many real-world problems where typically past data and knowledge learned do contain significant gaps.

Fuzzy rule interpolation (FRI) [26, 27, 28, 29, 30, 31] is of particular significance to support reasoning in the presence of insufficient knowledge. Given a sparse rule base, if an observation has no overlap with the antecedent of any rules, no rule can be invoked in classical fuzzy inference and therefore, no consequence can be derived. Fortunately, the use of FRI techniques enables inference to be performed in such cases. Moreover, with the help of FRI, the complexity of a rule base can be reduced by omitting fuzzy rules which may be approximated from their neighbouring rules.

1.3 Major Contributions

Although potentially powerful, using traditional deep learning architecture as the underlying technique for feature extraction introduces significant computation, which may well offset the potential benefit on the efficiency gained by the entire feature extraction process.

An alternative approach proposed in chapter 3 is to employ DeSTIN to extract only a small number of original features, and then to use interpolation to artificially create a more informative feature set of a higher dimensionality, thereby improving the representation of the underlying images to be classified. This will help increase the classification accuracy without sacrificing the efficiency of the overall system.

Another approach as proposed in chapter 4 offers a substantially simplified 2-layer machine learning network. This network exploits unsupervised learning for pattern representation, and is capable of extracting effective features efficiently. In so doing, the problem that feature extraction involves significant computation can be tackled.

In most fuzzy inference systems, the completeness of the fuzzy rule base is required to generate meaningful output when classical fuzzy inference methods are applied. This emphasises the need for a dense rule base for fuzzy inference that covers all possible inputs. Regardless of the way in which a rule base is constructed, be it

by human experts or by an automated agent, often incomplete or sparse rule bases are available. A dense rule base is especially impracticable in a multidimensional environment where the number of rules increases exponentially as the input variables and the fuzzy linguistic labels associated with each variable increase. In this situation, the classical fuzzy reasoning techniques such as compositional rule of inference (CRI) cannot generate an acceptable output for such cases. One simple solution to handle incomplete or sparse fuzzy rule bases and to infer reasonable output is through the application of fuzzy rule interpolation (FRI) methods.

Inspired by this observation, chapter 5 presents a direct utilisation of FRI in enhancing the reasoning robustness of a fuzzy rule-based classifier which utilises a sparse rule base, namely whose explicit knowledge has been induced from limited experienced data that does not cover the entire problem domain. The unified approach has been successfully applied on mammographic risk analysis from images.

1.4 Thesis Structure

This section outlines the structure of the remainder of this thesis.

Chapter 2: Background

This chapter provides a review of the most recent methods for deep machine learning, the most notable approaches such as DBNs, CNNs, DeSTIN are included for comparison. This chapter also provides a background introduction to fuzzy inference systems (FISs) / fuzzy rule-based systems (FRBSs), compositional rule of inference (CRI) and fuzzy rule interpolation (FRI). It provides a comprehensive review of typical CRI and FRI methods that have been developed in the last two decades.

Chapter 3: Interpolating DeSTIN Features for Image Classification

This chapter presents a novel approach for image classification, by integrating advanced machine learning techniques and the concept of feature interpolation. In particular, a recently introduced learning architecture, the Deep Spatio-Temporal Inference Network (DeSTIN) [5], is employed to perform feature extraction for support vector machine (SVM) based image classification. The system is supported

by use of a simple interpolation mechanism, which allows the improvement of the original low-dimensionality of feature sets to a significantly higher dimensionality with minimal computation. This in turn, improves the performance of SVM classifiers while reducing the computation otherwise required to generate directly measured features. The work is tested against the popular MNIST dataset of handwritten digits [32]. Experimental results indicate that the proposed approach is highly promising, with the integrated system generally outperforming that which makes use of pure DeSTIN as the feature extraction preprocessor to SVM classifiers. This chapter and parts thereof have been published initially in [33], with further and more in-depth versions in [34, 35, 36] .

Chapter 4: Clustering Supported Learning Network for Pattern Recognition

Deep machine learning has received significant attention over the past decade, especially in terms of dealing with information that may span large scales. By employing a hierarchical architecture, consisting of simple computational nodes of similar characteristic, the goal of such a network is to partition large data structures into relatively smaller, more manageable units, and to discover any dependencies that may exist between such units. However, the process of running this type of network which has a layered structure, for feature extraction itself typically involves significant computation. To tackle this problem, a substantially simplified 2-layer machine learning network is introduced that exploits unsupervised learning for pattern representation, capable of extracting effective features efficiently. Experimental results on application to handwritten digit classification demonstrate that the proposed approach is of significant potential in dealing with real-world problems. The techniques described in this chapter are currently under review for journal publication.

Chapter 5: Enriching Data-Driven Fuzzy Rule-Based Classification with Fuzzy Rule Interpolation

The generation of effective feature pattern-based classification rules from data is essential to the development of intelligent classifiers which are readily comprehensible to the user. Unfortunately, a sparse rule base may be generated when there is missing information in the experienced dataset. This hinders classification systems that work based on such sparse knowledge effectively performing their tasks in many

real-world applications, where complete historical data cannot be assumed. This chapter proposes an innovative approach by integrating fuzzy rule interpolation within a data-driven classification mechanism such that conclusions can be approximately derived even if no matched rule can be found from a given sparse rule base when given a certain observation. The proposed technique is conceptually simple, but it offers a powerful means to develop robust classifiers, significantly enhancing the effectiveness of intelligent classification systems, as demonstrated by systematic comparative experimental results.

Mammographic risk analysis from images is an important area of research as it provides an important indicator for the likelihood of a woman developing breast cancer, which is the leading cause of death of women in their 40's in the EU and US [37, 38]. Like many areas which deal with image data, there are large amounts of redundancy and noise in the data. With the use of feature selection, these extraneous features can be removed. Additionally, with the aid of the proposed classifier learner, a unified approach to mammographic risk analysis is formulated which can increase the accuracy of risk analysis and thus reduce the potential for misdiagnoses. This unified technique and application is also currently under review for journal publication.

Chapter 6: Conclusion

This chapter summarises the key contributions made by the thesis as well as a discussion of topics which form the basis for future research.

Appendix A

This appendix lists the publications arising from the work presented in this thesis, containing both published papers, and those currently under review for journal publication.

Chapter 2

Background

This chapter reviews the relevant literature, which sets the background of developments in this thesis. The rest of the chapter is organised as follows. Section 2.1 provides a review of typical deep machine learning methods that have been developed in the last two decades. Section 2.2 gives a brief introduction of fuzzy set theory. Section 2.3 explains the fuzzy inference systems (FISs) or fuzzy rule-based systems (FRBSs). Section 2.4 briefly discusses compositional rule of inference (CRI) which is an important part of the proposed integrated dynamic framework. Typical CRI methods are discussed. Section 2.5 explains fuzzy rule interpolation (FRI) which is the backbone of the proposed integrated framework. An overview of the most popular methods for fuzzy rule interpolation are presented. Finally, Section 2.6 summarises this chapter.

2.1 Deep Machine Learning

The human can efficiently and robustly represent information. Indeed, humans are exposed to myriads of sensory data received every second of the day and are somehow able to capture critical aspects of this data in a way that allows for its future use in a concise manner [39, 40, 41, 42]. Mimicking this ability has been a core challenge in artificial and computational intelligence research for decades [43, 44]. However, for intelligent systems, e.g., those performing autonomous pattern classification, the learning complexity grows exponentially with a linear increase in the data dimensionality [45]. In order to overcome this problem of so-called curse of

dimensionality [1], the mainstream approach in machine learning and related areas has been to pre-process the data in a manner that would reduce its dimensionality [2, 46, 47, 48]. After pre-processing, the data can be effectively processed for a given application. As a result, the intelligence behind many pattern recognition systems has shifted to human-engineered feature extraction processes, which at times can be themselves computationally challenging and highly application-dependent [49, 50, 51, 52].

Recent neuroscience research has found that the neocortex, which is associated with many cognitive abilities, does not explicitly pre-process sensory signals, but rather allows them to propagate through a hierarchy of modules, where each module tries to capture the regularities in the observations it exhibits [53, 54]. This important finding has provided insight into principles governing information representation in the mammalian brain, paving an innovative way for the ideas of designing intelligent systems that represent and process information. The research field of deep machine learning subsequently emerges, focussing on computational models for information representation that exhibits similar characteristics to that of the neocortex [55, 56, 57]. In such work, apart from the spatial information of real-life data, the temporal component can also play a key role.

A sequence of observed data patterns often conveys a certain meaning to the observer, whereby independent fragments of this sequence may be hard to decipher in isolation. Meaning is often inferred from events or observations that are received closely in time [58, 59, 60]. Therefore, modelling the temporal component of the observations is important for effective information representation and processing. Capturing spatio-temporal dependencies, based on regularities in the observations, is regarded as a fundamental goal for deep learning systems [61]. Following this approach, it would be possible to train a hierarchically structured network, on a given set of observations, and then to extract signals from this network to a relatively simple classification engine for the purpose of robust pattern recognition [62, 63]. Robustness here refers to the ability to exhibit classification invariance to a diverse range of uncertainty and imprecision, including noise and distortions involving scale, rotation, displacement, etc.

2.1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [4, 64] are the first truly successful deep learning approach particularly designed to process two-dimensional data, such as

images and videos. CNNs are inspired by the time-delay neural networks (TDNNs). TDNNs are designed for use on speech and time-series processing [65]. TDNNs reduce learning computation requirements by sharing weights in a temporal dimension. The choice of topology or architecture provides a CNN with the ability of using spatial relationships to reduce the number of parameters which must be learned and improving general feed forward back propagation training.

In CNNs, the inputs to the lowest layer of the hierarchical structure are small patches of the image (dubbed a local receptive field). CNNs allow information to propagate through the network, layer by layer, whereby at each layer digital filtering is applied in order to obtain salient features of the data observed. As the local receptive field allows the processing unit access to elementary features such as oriented edges or corners, CNNs provide a level of invariance to shift, scale and rotation.

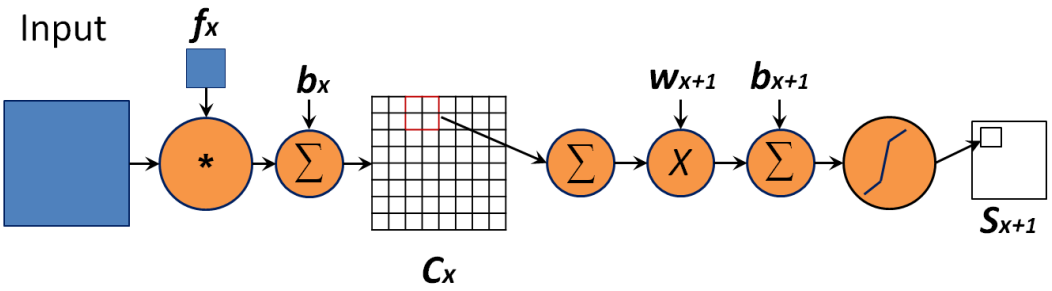


Figure 2.1: Convolution and subsampling process

Figure 2.1 shows the convolution and sub-sampling process. Essentially, a set of N small filters is convolved with the input image, the coefficients of N small filters are either trained or pre-determined using some criteria. The convolution process consists of convolving an input with a trainable filter f_x then adding a trainable bias b_x to produce the convolution layer C_x. This initial stage is followed by a sub-sampling (typically a 2×2 averaging operation) that further reduces the dimensionality and offer some robustness to spatial shifts. The sub-sampling consists of summing a neighbourhood (four pixels), weighing by scalar w_{x+1}, adding trainable bias b_{x+1}, and passing through a sigmoid function to produce a roughly 2x smaller

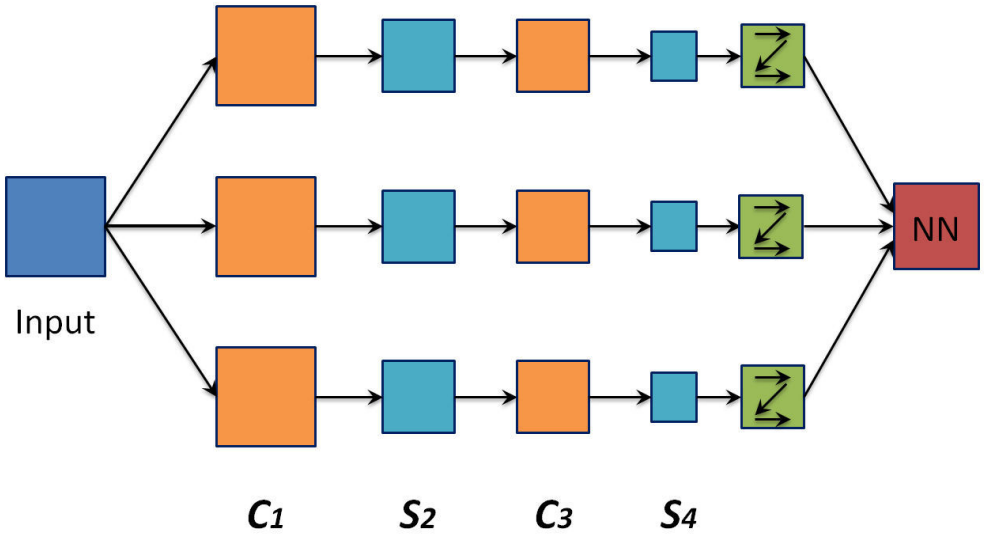


Figure 2.2: Conceptual example of convolutional neural network

feature map S_{x+1}. Some variants of this exist with as few as one map per layer [66] or summations of multiple maps [4].

The activation function is nearly linear when the weighting is small; other weighting can cause the activation output to resemble an AND or OR function. The new feature map is then passed through another sequence of convolution, sub-sampling and activation function flow, this process can be repeated an arbitrary number of times, as illustrated in Figure 2.2. The input image is convolved with three trainable filters and biases as in Figure 2.1 to produce three feature maps at the C_1 level. Each group of four pixels in the feature maps are added, weighted, combined with a bias, and passed through a sigmoid function to produce the three maps at S_2. The outputs of S_2 are again filtered to produce the C_3 level. The hierarchy then produces S_4 in a manner analogous to S_2. Finally these pixel values are rasterized and presented as a single vector input to the conventional neural network at the output.

Note that one or more of the previous layers can be combined to feed to the subsequent layers; for example, in [4] the initial six feature maps are combined to form 16 feature maps in the subsequent layer. A method dubbed "feature pooling" (the S layers in Figure 2.2) allows CNNs invariance to object translations [54]. However, feature pooling is not trained or learned by the system, it is hand crafted by the network organizer.

The intimate relationship between the layers and spatial information in CNNs renders them well suited for image processing and understanding, and they generally perform well at autonomously extracting salient features from images. In some cases Gabor filters have been used as an initial pre-processing step to emulate the human visual response to visual excitation [67]. In more recent work, researchers have applied CNNs to various machine learning problems including face detection [66, 68], document analysis [69], and speech detection [70]. CNNs have recently been trained with a temporal coherence objective to leverage the frame-to-frame coherence found in videos, though this objective need not be specific to CNNs [71].

The strength of CNNs is reflected in the fact that it can deliver high accuracy in image recognition problems. The main limitation of CNNs is it involves high computational cost. They are quite slow to train (for complex tasks) and a lot of training data are needed.

2.1.2 Deep Belief Networks

Deep Belief Networks (DBNs), as probabilistic generative models, were initially introduced in [2]. Generative models provide a joint probability distribution over observation and labels and both *P(Observation|Label)* and *P(Label|Observation)* can be estimated. While traditional neural nets, as discriminative models, are limited to estimate *P(Label|Observation)*. Furthermore, DBNs tackle the weaknesses of deeply-layered neural networks that back-propagation are applied, namely: (1) labelled data are needed for training, (2) time consuming (i.e. slow convergence), (3) poor local optima are obtained when parameter selection are inadequate.

Restricted Boltsmann Machines (RBMs) [72], a type of neural network, form several layers of DBNs (see Figure 2.3). RBMs are composed of a visible layer and a hidden layer, connections are formed between these two layers, while units within a layer are not connected. The outputs of the visible units are used to train the hidden units that try to capture the correlations in the observation data. The top two layers form an associative memory. Aside from associative memory, directed top-down generative weights connect the layers of a DBN. An unsupervised greedy layer-by-layer manner, enabled by contrastive divergence [73], occurs during the pre-training to obtain generative weights. Due to the ease of learning these connection weights, RBMs are chosen as a building block over more traditional and deeply layered sigmoid

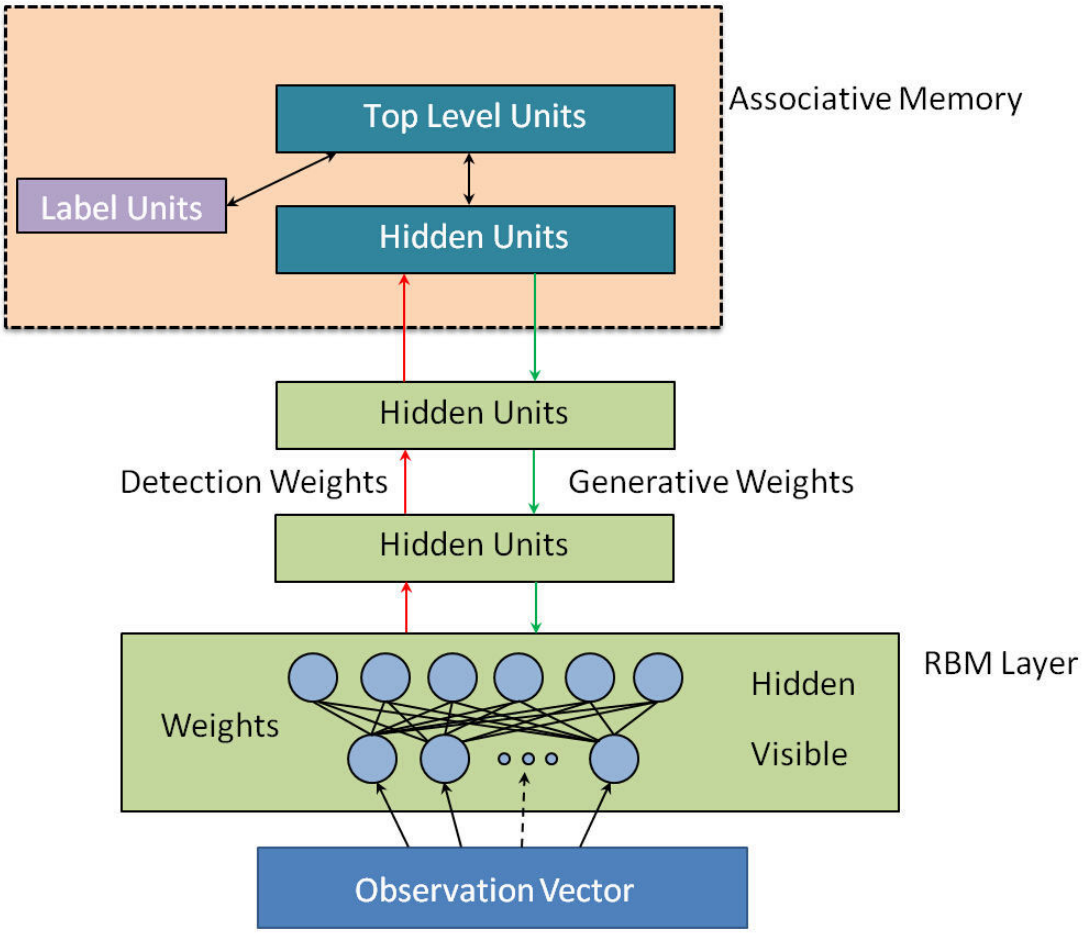


Figure 2.3: Deep belief network

belief networks. The training phase of contrastive divergence can be illustrated as following: the visible units receive $\vec{v}$, the observation data, and forward values to hidden units. Then the visible unit inputs are stochastically found in an attempt to reconstruct the original input. Finally, hidden unit activations, $\vec{h}$, can be reconstructed one step based on the new visible neuron activations. Performing these back and forth steps is a process known as Gibbs sampling. The weight update is based on the difference in the correlation of visible inputs and the hidden activations. Note that only a single step is needed to approximate maximum likelihood learning, so the training time is significantly reduced. Each layer added to the network improves the log-probability of the training data, which means true representation power increases.

The detection weights and generative weights are tied together at the top two layers. The top layer receives the output of the lower layer as a reference clue or

link to associate with its memory contents. Discriminative performance is of great concern in classification tasks. By utilizing labelled data through back-propagation, the discriminative performance can be improved after pre-training. A set of labels is attached to the top layer, and the associative memory is expanded, this helps to clarify category boundaries in the network. A new set of bottom-up recognition weights are learned through the category boundaries. A DBN may be fine tuned, and such networks often perform better than those trained exclusively with back-propagation [74]. In contrast to traditional feed-forward neural networks, back-propagation for DBNs is only required to perform a local search on the weight space (parameter space). This fact intuitively explains that DBN can speed training and convergence in less time.

A more thorough analysis of DBNs' use with unsupervised tasks as well as continuous valued inputs was presented in [3] shortly after DBNs were introduced. Further testing had been carried out to illustrate the resilience of DBNs on problems with increasing variation [48, 75].

Recently, Convolutional Deep Belief Networks (CDBNs) [52] were introduced to expand the flexibility of DBNs. DBNs lack the ability to inherently embed information about 2D structure of an input image, i.e. an image matrix is simply vectorized to act as the inputs. In contrast, with the introduction of convolutional RBMs, the spatial relationship of neighbouring pixels is utilized by CDBNs, so CDBNs as a translation invariant generative model scales well with high dimensional images. Another shortcoming of DBNs is the temporal relationships between observation vectors can not be explicitly learned. Currently, stacking temporal RBMs [76] or generalizations of these, dubbed temporal convolution machines [77] have been proposed to learn sequences. Audio signal processing problems offer an important application area of such sequence learner [78].

Recent works pertaining to DBNs include the use of stacked auto-encoders in place of RBMs in traditional DBNs [3, 48, 79]. This effort produced deep multilayer neural network architectures that can be trained with the same principles as DBNs but are less strict in the parameterization of the layers. Unlike DBNs, auto-encoders use discriminative models from which the input sample space cannot be sampled by the architecture, making it more difficult to interpret what the network is capturing in its internal representation. However, it has been shown that denoising autoencoders, which utilize stochastic corruption during training, can be stacked to

yield generalization performance that is comparable to (and in some cases better than) traditional DBNs [79]. The training procedure for a single denoising autoencoder corresponds to the goals used for generative models such as RBMs.

2.1.3 Recently Proposed Deep Learning Architectures

Several computational architectures have been introduced to try to model the neocortex. Inspired by sources such as [80], these models attempt to map various computational phases in image understanding to areas in the cortex. Although these models have been refined over time, a hierarchical structure has remained, because the hierarchical structure is the core component in the central concept of visual processing. Based on studies of visual cortical cells of cats, Hubel and Weisel proposed the simple-to-complex cell organization [81]. CNNs as well as other deep-layered models (such as Neocognitron [82, 83, 84] and HMAX [63, 85]), utilize similar organizations, yet a stronger mapping of their architecture to biologically-inspired models yields more "explicit" cortical modes. Diverse mechanisms have been attempted to solve problems of learning and invariance. In particular, temporal analysis, in which temporal information is considered an important element of learning process, is utilized by Numenta Corporation to create Hierachical Temporal Memory (HTM)[54].

Based on concepts described in [86], HTMs are designed as a hierarchical structure. Essentially, an input image feeds to the lowest level of the hierarchy, each unit in the lowest layer receives a small region of the image as its input and tries to represent the visual information. Higher levels of the hierarchy incorporate the representation constructs of multiple lowest receptive fields, so higher levels correspond to larger regions. HTMs bear similarities to other deeply-layered models in terms of the visual information representation. In addition to the visual information representation across layers of hierarchy, temporal information is created by translation or scanning of the input itself to each layer. With a specific focus on the learning phase, the lowest layer compiles the most common input patterns and indices are assigned to the input patterns, probability transitions from one input sequence to another form the temporal relationships. Then graph partitioning techniques are used to cluster these temporal relations. When the learning stage concludes in the first layer, the second layer assigns the indices to the inputs from its child modules in the first layer and learns the most common input patterns at a higher level. Characterization obtained

in the higher layer can then be provided to the lower level as the feed back. In turn, the inference formulation in lower level incorporates this boarder representation information. This process is repeated at each layer of the hierarchical structure until the network is trained. Given the beliefs at the top layer of the hierarchy, image recognition can be performed, a Bayesian belief propagation algorithm [42] is used to identify the most likely input pattern.

There are other architectures that share this similar idea, such as Hierarchical Quilted SOMs of Miller Lommel [62] and Neural Abstraction Pyramid of Behnke [87].

In order to achieve robust information representation, a framework called the Deep Spatial-Temporal Inference Network (DeSTIN) [5] was recently introduced. The entire hierarchy is composed of a common cortical circuit (or node). The node in each layer is independent to all other nodes and operates in parallel. This kind of structure makes DeSTIN highly suitable for implementation on parallel processing platforms.

As the hierarchy is presented with data, each node tries to capture the patterns in the data independently. A belief state is used to represent a possible pattern, and is incrementally updated as new data comes in. Two constructs are used to implement the belief state update: *P(observation|state)*, which represents how likely system states are for segments of the observation, is constructed unsupervised and driven purely by observations. Another construct *P(subsequent state|current state, feedback)*, which represents how likely state to state transitions are given feedback from above, modulates *P(observation|state)* and embeds the dynamics in the pattern observations. To estimate the observation distribution, an on-line clustering method is carefully applied. Based on the frequency, state transitions can be obtained. Unfortunately, training this type of network demands significant computational effort.

2.1.4 Deep Learning Applications

The effectiveness of deep learning methods in a variety of application domains have been demonstrated in several studies [32, 64, 67, 70, 78, 88]. Much research has been done in the following area: MNIST handwriting challenge, face detection, speech recognition and detection, general object recognition, natural language processing, robotics.

Interest in deep machine learning has not been limited to academic research. Recently, the Defense Advanced Research Projects Agency (DARPA) has announced a research program exclusively focused on deep learning. Several private organizations, including Numenta and Binatix, have focused their attention on commercializing deep learning technologies with applications to broad domains.

2.1.5 Summary of Comparison of Typical Deep Learning Methods

For all the outlined typical deep learning methods, comparisons are summarised based on the identified criteria in Table 2.1. This comparison shows that DeSTIN can explicitly capture temporal and spatial information, supervised learning is not needed and can be updated online. Two criteria: spatial mapping, and supervised learning are fulfilled by CNN. Updating online and temporal mapping are not supported by CNN The DBN method combines both supervised and unsupervised learning.

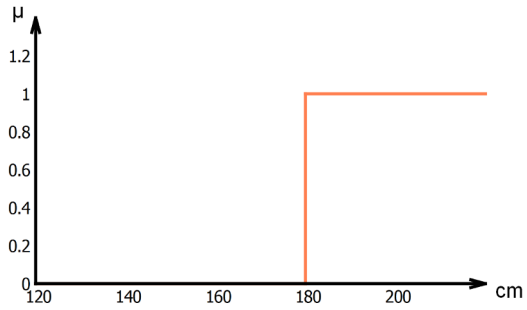
Table 2.1: Summary of Typical Deep Learning Methods with regards to Evaluation Criteria

	Temporal Mapping	Spatial Mapping	Supervised Learning	Online
DeSTIN	Explicit	Explicit	None	Yes
CNN	None	Explicit	SL	No
DBN	None	None	SL and UL	No

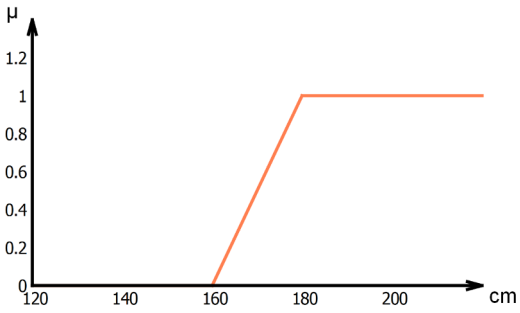
2.2 Fuzzy Set Theory

The modelling of imprecise and qualitative knowledge, as well as the transmission and handling of uncertainty at various stages are possible through the use of Fuzzy Sets (FSs) [89]. Fuzzy logic is capable of supporting human type reasoning in natural form [90]. It is the earliest and most widely reported constituent of soft computing (SC). The development of fuzzy logic has led to the emergence of soft computing [91].

Fuzzy sets are a further development of the mathematical concept of a set. A fuzzy set is an extension of a crisp set, where the latter allows only full membership or no membership at all, whereas the former allows partial membership. In a crisp



(a) Crisp characteristic function



(b) Fuzzy membership function

Figure 2.4: Fuzzy membership functions for variable "height"

set, membership or non-membership of an element is described by a characteristic function in the binary pair {0, 1}. Fuzzy set theory extends this concept by defining partial membership. A fuzzy set is characterised by a membership function (MF) that takes values in the interval [0, 1]. In this case, a given element can be a member of more than one fuzzy set at a time.

As an example, consider the concept tall. In a crisp set, all of the people with height 180 cm or more are considered tall, and all of the people with height of less than 180 cm are considered not tall. The crisp set characteristic function is shown in Figure 2.4a, while the corresponding fuzzy set with a smooth membership function is shown in Figure 2.4b, where X and Y axes denote the height and its corresponding membership value, respectively. The membership function curve defines the transition from not tall and shows the degree of membership for any given height.

Let X be the universe, a fuzzy set, A, in X is a set of ordered pairs

$$A = \{(x, \mu_A(x)) | \mu_A(x) \in [0,1], x \in X\} \tag{2.1}$$

Such a fuzzy set is a collection of objects with graded membership, where $\mu_A(x)$ is termed the grade of membership of x in A. The closer the value of $\mu_A(x)$ is to 1, the more x belongs to the set A.

Essentially, a membership function is a function that defines how each point in the input space is mapped to a membership value between 0 and 1. Various types of membership functions can be used, including triangular, trapezoidal, Gaussian curves, polynomial curves, etc. In particular, due to the fact that triangular and trapezoidal fuzzy sets are commonly used in many fuzzy rule interpolation (FRI) approaches [26, 27, 30, 31, 92]. Triangular and trapezoidal membership functions are defined respectively by three and four parameters and given by

$$f(x:a,b,c) = \begin{cases} 0 & if\ x < a \\ \dfrac{x-a}{b-a} & if\ a \le x \le b \\ \dfrac{c-x}{c-b} & if\ b \le x \le c \\ 0 & if\ x > c \end{cases} \tag{2.2}$$

where a and c denote the left and right extreme points (with membership values of 0), and b denotes the normal point (with a membership value of 1).

$$f(x:a,b,c,d) = \begin{cases} 0 & if\ x < a \\ \dfrac{x-a}{b-a} & if\ a \le x < b \\ 1 & if\ b \le x \le c \\ \dfrac{d-x}{d-c} & if\ c < x \le d \\ 0 & if\ x > d \end{cases} \tag{2.3}$$

where a and d denote the left and right extreme points (with membership values of 0), and b and c denote the normal points (with membership values of 1).

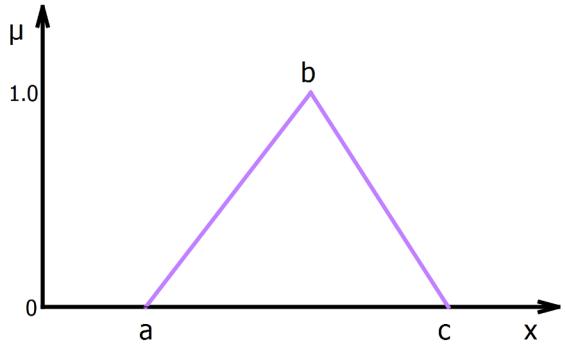


Figure 2.5: A triangular membership function example

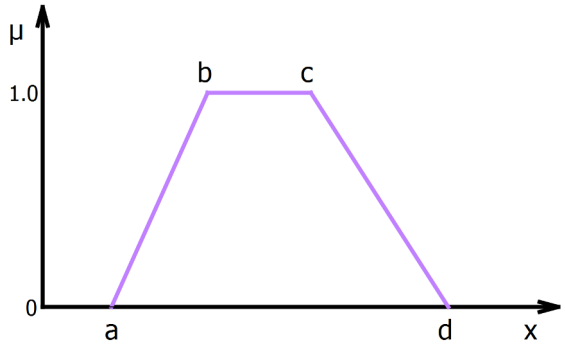


Figure 2.6: A trapezoidal membership function example

The *support* of a fuzzy set A is defined by

$$supp(A) = \{x \in X | \mu_A(x) > 0\} \tag{2.4}$$

and the *core* of a fuzzy set A is defined by

$$core(A) = \{x \in X | \mu_A(x) = 1\} \tag{2.5}$$

An important property of fuzzy sets is their convexity. A fuzzy set A on X is *convex* if and only if

$$\mu_A(\lambda x_1 + (1-\lambda)x_2) \geq min(\mu_A(x_1), \mu_A(x_2)) \tag{2.6}$$

for all $x_1, x_2 \in X$ and all $\lambda \in [0,1]$.

An equivalent representation to the above standard definition is: a fuzzy set A is said to be *convex* if and only if

$$\mu_A(z) \geq min(\mu_A(x), \mu_A(y)), \forall (x,y,z) \in X \text{ and } z \in [x,y] \tag{2.7}$$

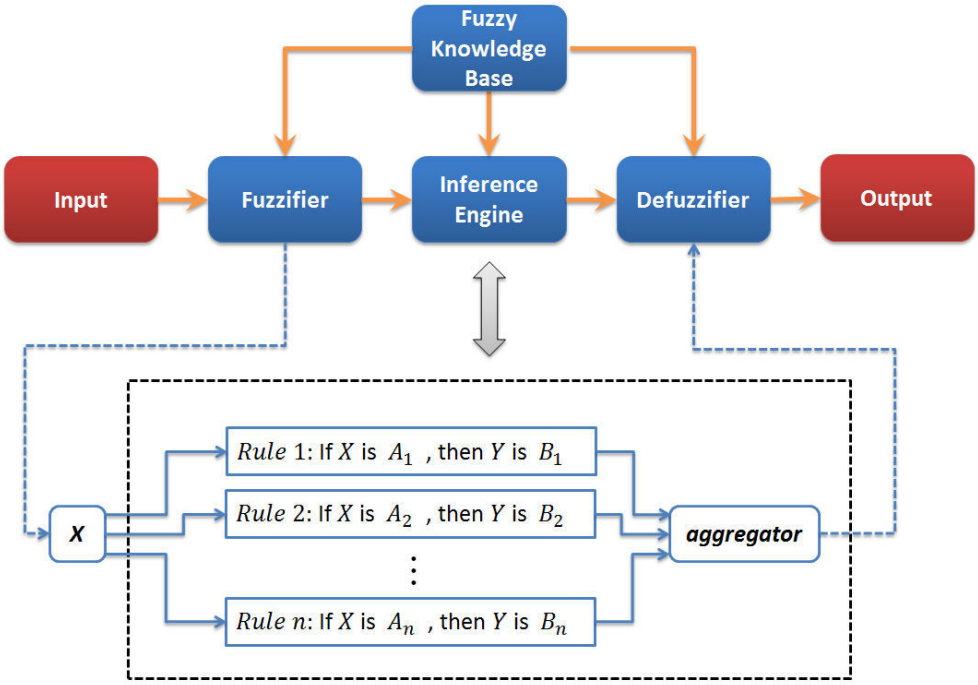


Figure 2.7: Generic fuzzy inference system

where z is a point between x and y.

A fuzzy set A is said to be *normal* if and only if

$$\mu_A(x) = 1, \exists x \in X \tag{2.8}$$

2.3 Fuzzy Inference System

The process of fuzzy inference is basically an iteration of a computer paradigm based on fuzzy set theory, fuzzy IF-THEN rules and logical operations. Each iteration takes inputs which can be an observation or a previously inferred crisp or fuzzy result. These inputs are then used to "fire" the rules in a given rule base. From this, the output is the aggregation of the inferred results from all of the fired rules. The general structure of fuzzy inference is illustrated in Figure 2.7.

The *fuzzifier* maps discrete or real-valued inputs into corresponding fuzzy memberships. This is required in order to build rules that can be considered in terms of linguistic variables. The *fuzzifier* takes input values and determines the degree to which they belong to each of the fuzzy sets by means of membership functions.

The *rule base* contains linguistic rules that are provided by experts. It is also possible to extract rules from numerical data. Once the rules have been established, the fuzzy inference system (FIS) can be viewed as a system that maps an input vector to an output vector.

The *inference engine* defines the mapping from input fuzzy sets into output fuzzy sets. It determines the degree to which the antecedent is satisfied for each rule. If the antecedent of a given rule has more than one part, fuzzy operators are applied to obtain a number that represents the result of the antecedents for that particular rule. Furthermore, if one or more rules fire simultaneously, outputs for all rules are then aggregated. During the aggregation process, fuzzy sets that represent the output of each rule are combined into a single fuzzy set.

The *defuzzifier* maps output fuzzy sets into a crisp or discrete output. Given a fuzzy set that encompasses a range of output values, the *defuzzifier* returns a single value. Several methods for defuzzification can be used in practice, including: centroid, maximum, etc.

2.4 Compositional Rule of Inference (CRI)

Fuzzy systems use a fuzzy rule base (set of rules) to contain knowledge that is exploited to make inference by the inference mechanism. A fuzzy rule base is fully covered at level α, if all input universes are covered by rules at level α. Such fuzzy rule bases are also called dense or complete rule bases. In practice, it means that for all the possible observations there exists at least one rule, whose antecedent part overlaps the input data at least partially at level α. If this condition is not satisfied, the rule base is considered sparse, i.e. containing gaps.

In order to draw conclusions from a dense rule base, one needs a mechanism that can produce an output from a collection of rules. The most commonly used inference process for dense rule bases is the compositional rule of inference (CRI) [19, 89], as shown in Figure 2.8. For a given observation, in order to obtain a meaningful inference result based on CRI, there are two basic approaches: First Infer - Then Aggregate (FITA) and First Aggregate - Then Infer (FATI). In the FITA approach, for a given observation, first inference is performed using CRI on each of the rules in the rule base, and then combine all these intermediate results. Whereas, in the FATI

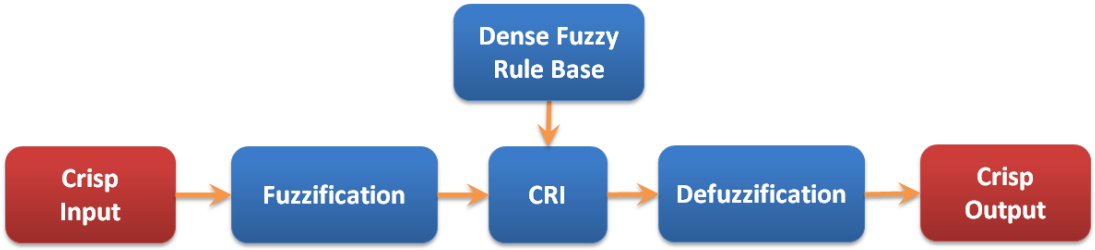


Figure 2.8: Compositional rule of inference system

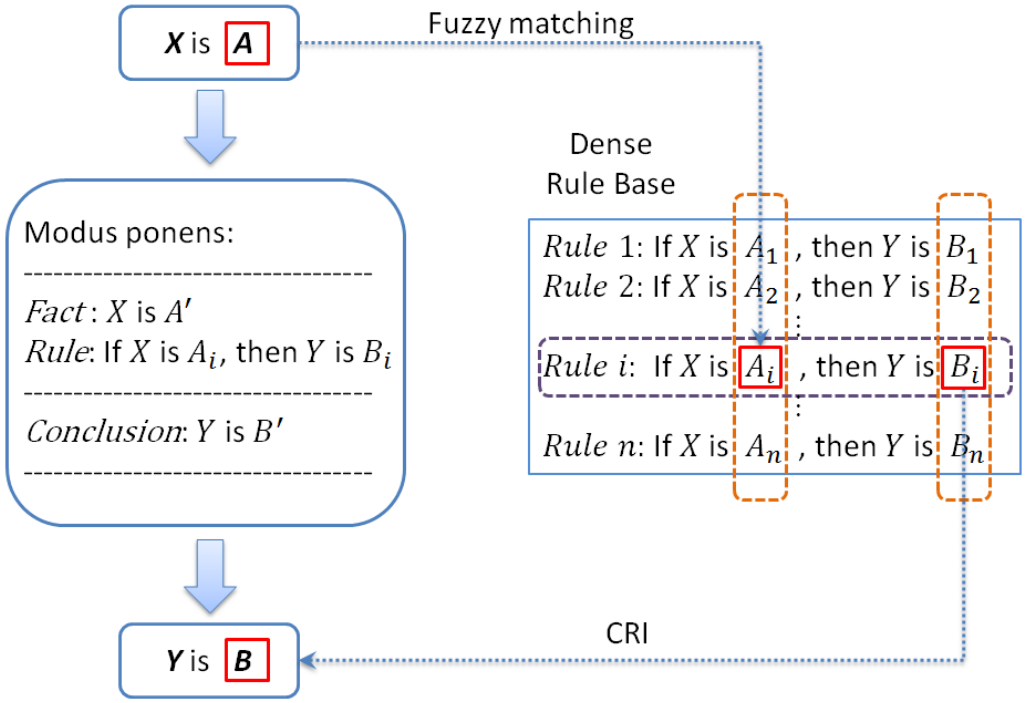


Figure 2.9: Compositional rule of inference example

approach, all the rules are first aggregated by forming an overall fuzzy relation R which is the combination of all the fuzzy implication relations and then inference is performed on the given observation, so compared with FITA, FATI performs the inference only once.

An inference process based on CRI changes the membership function grades of the right hand sides of the corresponding rules either by reducing or by increasing

the membership grades [93]. CRI is also called generalised modus ponens (GMP). For example, here reasoning is performed with one rule using CRI based on FATI. With a single rule and an observation, an inference result can be deduced as follows:

$$\begin{gathered} \textit{Rule : If x is A, then y is B} \\ \textit{Observation : x is } A^{'} \\ \textit{Consequence : y is } B^{'} \end{gathered}$$

where $A, A^{'} \subset X, B \subset Y$ are fuzzy sets defined in the universes of discourse X and Y, $x \in X$, and $y \in Y$. The fuzzy rule is interpreted as an implication:

$$R: A \rightarrow B$$

When input observation $A^{'}$ is given to the inference system, the output consequence would be calculated:

$$B^{'} = A^{'} \; o \; R = A^{'} \; o \; (A \rightarrow B)$$

Where o is the composition operator. This inference procedure is called compositional rule of inference as shown in Figure 2.9. Here the inference mechanism is determined by two factors: 1. implication operators such as *min*, *product*, etc. and 2. composition operator such as *max-min*, *max-product*, etc. Therefore, it is clear that an inference process based on CRI includes several stages. More specifically, it includes implication, composition and combination for FITA, and implication, combination and composition for FATI.

There are many methods to select from in order to implement the required implication, composition and combination operators to perform CRI. The most common fuzzy inference methods based on CRI are Mamdani and Takagi-Sugeno-Kang (TSK) fuzzy inference methods. The Mamdani fuzzy inference is the most commonly seen. This method was introduced by Mamdani and Assilian in 1975 [14]. Another well-known inference method is Takagi-Sugeno-Kang (TSK) method. This inference method was introduced by Takagi,Sugeno and Kang in 1985 [94, 95].

The main difference between the two popular Mamdani inference method and TSK inference method is the way the crisp output is generated from the fuzzy inputs.

While Mamdani system uses the technique of defuzzification of a fuzzy output, TSK system uses weighted average to compute the crisp output. The expressive power and interpretability of Mamdani output is reduced in the TSK systems since the consequents of the rules are not fuzzy [96]. However, TSK has better processing time since the weighted average replaces the time consuming defuzzification process. Due to the interpretable and intuitive nature of the rule base, Mamdani inference systems are widely used in particular for decision support applications [96].

2.4.1 Mamdani Fuzzy Inference Systems

From the introduction of fuzzy sets by Zadeh in 1965 [89], fuzzy logic has become a significant area of interest for researchers in artificial intelligence. In particular, Mamdani was the pioneer who investigated the use of fuzzy logic for interpreting the human derived control rules, and therefore his work was considered a milestone application of this theory [97]. The original Mamdani fuzzy inference system was proposed as the first attempt to control a steam engine and boiler combination by a set of linguistic control rules obtained from experienced human operators. A fuzzy system with two inputs x and y (antecedents) and a single output z (consequent) is described by a linguistic IF-THEN rule in Mamdani form as [14]:

$$\textit{If } x \textit{ is } A \textit{ and } y \textit{ is } B, \textit{ then } z \textit{ is } C$$

where A and B are fuzzy sets in the antecedent and C is a fuzzy set in the consequent.

Figure 2.10 is an illustration of how a two-rule Mamdani fuzzy inference system derives the overall output z when subjected to two crisp inputs x and y. If *min* and *max* are chosen as the *T-norm* and *T-conorm* operators, respectively, and use the original *max-min* composition, then the resulting fuzzy reasoning is shown in Figure 2.10, where the inferred output of each rule is a fuzzy set scaled down by its firing strength via *max*. Other variations are possible if different *T-norm* and *T-conorm* operators are used. For example, using *product* and *max* for *T-norm* and *T-conorm* operators, respectively results in the *max-product* composition.

In general, to compute the output for the given input observation, a Mamdani inference system follows the following steps:

1. Determining a set of fuzzy rules

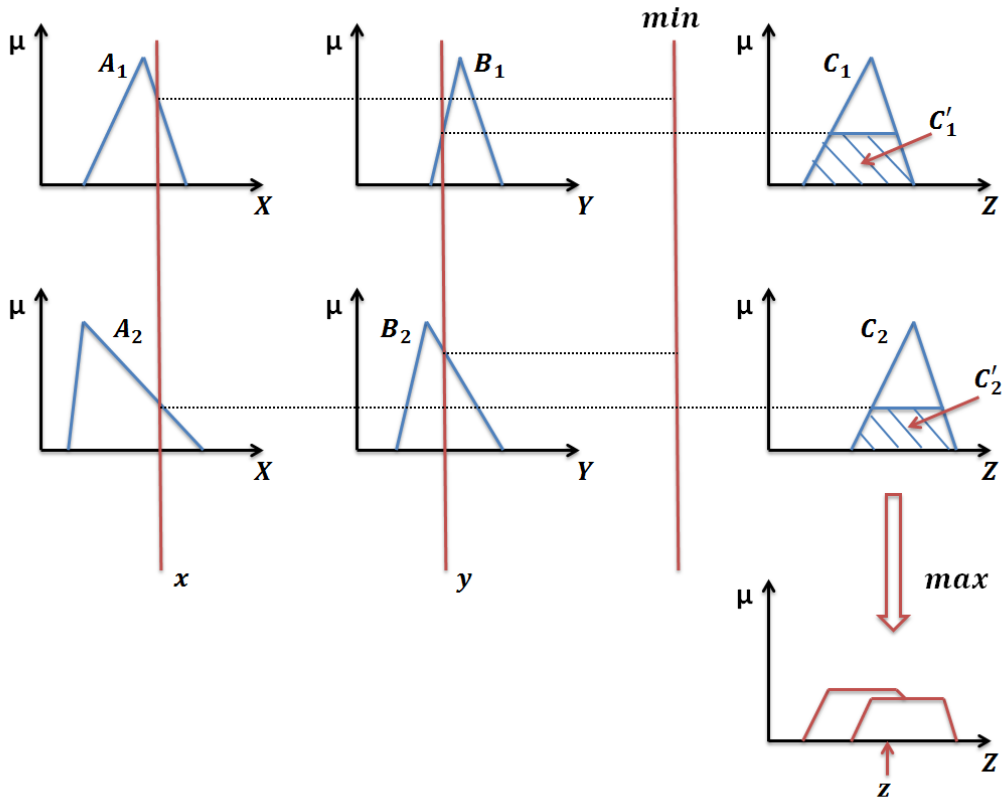


Figure 2.10: Mamdani fuzzy inference system

2. Fuzzifying the inputs using the input membership functions

3. Combining the fuzzified inputs according to the fuzzy rules to establish a rule strength

4. Finding the consequence of the rule by combining the rule strength and the output membership function

5. Combining the consequences to get an membership function

6. Defuzzifying the output distribution (this step is involved only if a crisp output (class) is needed, if it's a prediction or regression problem, this step is not needed)

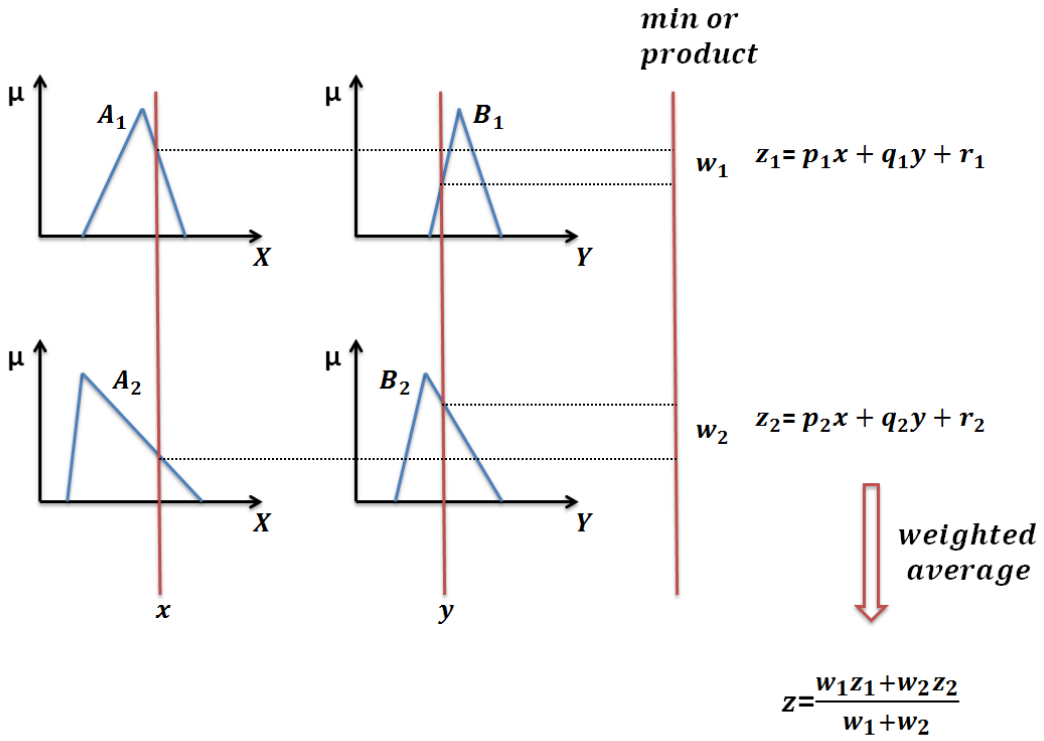


Figure 2.11: Takagi-Sugeno-Kang (TSK) fuzzy inference system

2.4.2 Other Types of Fuzzy Inference Systems

2.4.2.1 Takagi-Sugeno-Kang (TSK) Fuzzy Inference Systems

Tagaki, Sugeno, and Kang [94, 95] investigated a new approach to fuzzy inference models with the emphasis upon systematic methods of generating fuzzy rules from given sets of input-output data. A fuzzy system with two inputs x and y (antecedents) and a single output z (consequent) is described by a linguistic *If-Then* rule in Takagi-Sugeno-Kang form as:

$$\textit{If } x \textit{ is } A \textit{ and } y \textit{ is } B, \textit{ then } z = f(x,y)$$

where fuzzy antecedent variables A and B, give rise to the consequent crisp function $z = f(x, y)$.

The first-order TSK fuzzy inference procedure is shown in Figure 2.11. Here, both rules have crisp outputs so the final output can be calculated through weighted

average, therefore Mamdani model's time-consuming process of defuzzification can be avoided. Conversely, this generalization could lead to the loss of MF linguistic meanings except the sum of firing strengths (that is, $\sum_i w_i$) is close to unity [98]. A TSK fuzzy inference model is not a strict compositional rule of inference model, where the matching of fuzzy sets can still be used to find the firing strength of each rule [98] which is shown in the antecedent part of Figure 2.11. While the final output is always a crisp output whether it is based on weighted average or weighted sum; this does not seem logically correct because a fuzzy model should be able to transmit the fuzziness from inputs to outputs. Nevertheless, TSK fuzzy inference is a common option for data-oriented fuzzy modelling when simplified defuzzification is required.

2.4.2.2 Type-2 Fuzzy Inference Systems

Quite often, the knowledge used to construct rules in a fuzzy inference system (FIS) is uncertain. This uncertainty leads to rules having uncertain antecedents and/or consequences, which in turn translates into uncertain antecedent and/or consequent membership functions (MFs).

Most of these types of uncertainty translate into difficulties about fuzzy set MFs. Type-1 fuzzy sets are not able to model such types of uncertainty because their MFs are crisp. On the contrary, type-2 fuzzy sets are able to model such uncertainty, because their MFs are themselves fuzzy.

The structure of a type-2 FIS is very similar to the structure of a type-1 FIS, which is shown in Figure 2.12. A type-2 FIS is characterised by IF-THEN rules, but its antecedent and/or consequent sets are now type-2 fuzzy sets. It includes fuzzifier, rule base, inference engine, and output processing. For a type-1 FIS, the output processing block only contains the defuzzifier.

The *fuzzifier* maps the crisp input into a fuzzy set. In general, this fuzzy set can be a type-2 set or a singleton where the input fuzzy set only has a single point of non-zero membership.

For the *rule base*, the distinction between type-1 and type-2 is associated with the nature of the MFs, which is not important while forming rules. For this reason, the structure of the rules remains exactly the same in type-2 FIS, the only difference

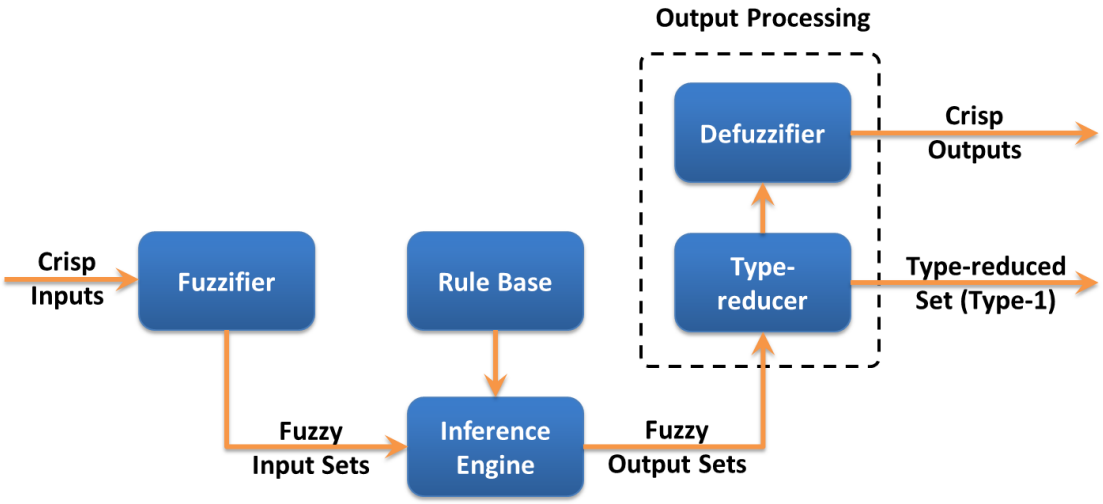


Figure 2.12: Type-2 fuzzy inference system

being that some or all of the involved sets are of type-2. However, it is not necessary that all the antecedents and consequences be type-2 fuzzy sets. As long as one antecedent or the consequent set is type-2, it is a type-2 FIS.

The *inference engine* in a type-1 FIS combines rules and gives a mapping from input type-1 fuzzy sets to output type-1 fuzzy sets. Multiple antecedents in rules and multiple rules are connected by the *T-norm* (corresponding to intersection of sets) and the *T-conorm* (corresponding to the union of sets), respectively. Similarly, the inference engine in a type-2 FIS combines rules and gives a mapping from input type-2 fuzzy sets to output type-2 fuzzy sets with the use of intersections and unions of type-2 fuzzy sets.

In a type-1 FIS, the *defuzzifier* produces a crisp output from the fuzzy set that is the output of the *inference engine*, i.e., a type-0 (crisp) output is obtained from a type-1 set. In the type-2 case, an operation analogous to type-1 defuzzification results in a type-1 set from a type-2 set, which is the output of the *inference engine*. This operator is called type-reducer and the resultant set is called a "type-reduced set". This type-reduced set can be further defuzzified by the *defuzzifier* to obtain a crisp output. The most natural way of doing this seems to be by finding the centroid of the type-reduced set [99, 100], however, there exist other possibilities like choosing the highest membership point in the type-reduced set [101, 102].

2.5 Interpolative Reasoning Methods

Fuzzy systems use fuzzy rule bases to make inference. If the input domain is covered completely by the rule bases, such fuzzy rule bases are called dense rule bases [103]. In dense rule bases, for all the possible observations there exists at least one (at least partially) fired rule, whose antecedent part overlaps the input data. When an observation occurs, a consequence can be inferred by using conventional fuzzy reasoning methods such as Mamdani [14, 104] and TSK [94, 95]. On the contrary, for a sparse rule base, that is, the input domain is covered incompletely by the rule base, there is an empty space between two membership functions of antecedents [105]. In this case, conventional fuzzy reasoning methods may encounter difficulty if an observation occurs in the empty space (which is also termed a "gap"), resulting in no rule fired and thus, no consequence derived. In general, the "empty space" is above a certain minimum confidence threshold if membership functions like Gaussian are used.

The reasons for sparse or incomplete rule bases are various but have several aspects [25]: Originally, fuzzy systems were constructed from IF-THEN rules provided by human experts. More recently, learning techniques have increasingly been developed and applied to the construction of fuzzy IF-THEN rules from numerical data. However, both ways of constructing rule bases can result in sparse rule bases. In the former case, an incomplete rule base may be the consequence of missing expertise for certain system configurations. In the latter case, it may be that data used in the construction of the rule base does not sufficiently represent the input parameters. Fuzzy inference methods are often criticised when the number of inputs is large. The size of the rule base and the complexity of the inference algorithm grow exponentially with the number of inputs. A possible solution to reduce complexity is to omit redundant rules. This can, however, lead to incomplete rule bases [106]. "Gaps" can be defined between rule bases intentionally, in order to avoid high complexity in large systems.

In the case where a fuzzy rule base contains "gaps", conventional fuzzy reasoning methods can no longer be used. This fact is due to the failure of traditional inference mechanisms in the case when observations find no fuzzy rule to fire. This cannot be allowed when using a fuzzy system in any practical application and such a system is considered useless. This problem was initially outlined in the "tomato classification" problem [107], shown in Figure 2.13.

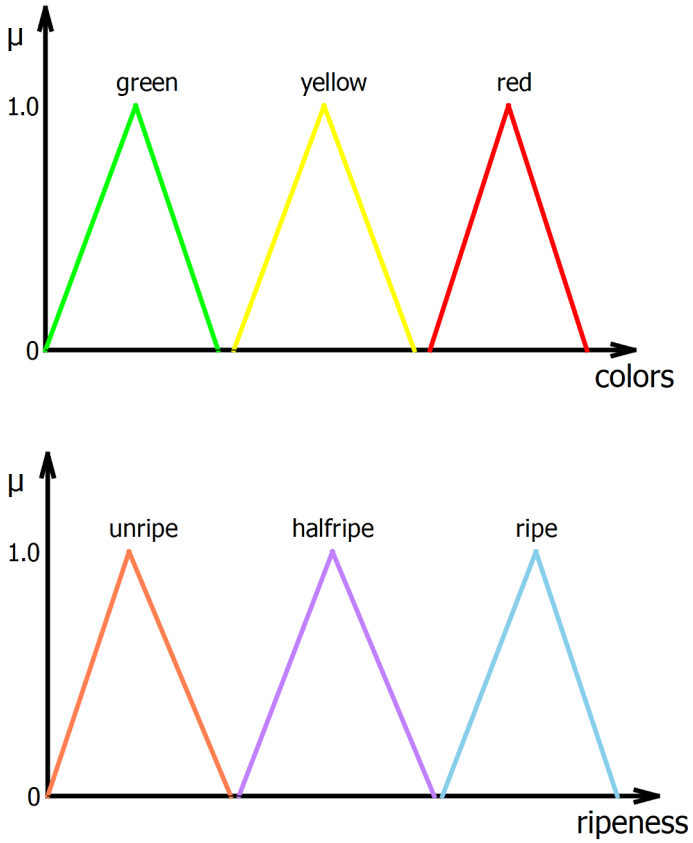


Figure 2.13: Fuzzy reasoning assumption of the tomato classification problem

Rule 1 : If a tomato is red, then the tomato is ripe.
Rule 2 : If a tomato is green, then the tomato is unripe.
Observation : This tomato is yellow.
Conclusion : ???

The intuitive consequence of a human being would be that this tomato is half ripe. However, the membership function "yellow" has no overlap with the membership functions "red" or "green". Therefore, none of the conventional fuzzy inference mechanisms is able to reach such a conclusion.

Motivated by this, fuzzy interpolative reasoning mechanisms are proposed for performing fuzzy inference with systems comprising insufficient knowledge or sparse rule bases. Even when a given observation has no overlap with the antecedent values of any existing rules, FRI may still derive a useful conclusion. The techniques of FRI not only support inference in such situations, but also help to reduce the complexity

of fuzzy models by eliminating the rules which may be approximated from their neighbouring rules.

A number of important FRI approaches have been proposed in the literature [108, 109, 110]. In terms of the underlying methodology, most of these approaches can be divided into two groups: single step rule interpolation and intermediate rule-based interpolation.

The first group of approaches directly interpolates a rule whose antecedent is identical to the given observation and thus, the consequence of the interpolated rule is the logical result of the observation. The most typical approach in this group is the first proposed FRI technique [28], denoted the KH (Kóczy and Hirota) approach, which is based on the Decomposition Principle [10, 111, 112]. According to these principles, each fuzzy set can be represented by a series of α-cuts $\alpha \in (0, 1]$. Given α, the α-cut of the interpolated consequent fuzzy set can be calculated from the α-cuts of the (newly observed) antecedent fuzzy sets and all of the fuzzy sets involved in the rules used for interpolation. Having found the α-cuts of the consequent fuzzy set for all $\alpha \in (0, 1]$, the consequent fuzzy set is then assembled by applying the Resolution Principle.

The second group of approaches reaches the target in two steps. In the first step these approaches interpolate an artificial intermediate rule. The antecedent of this intermediate rule is expected to be very close to the given observation. As a result, the interpolation problem becomes similarity reasoning [24, 113, 114]. The estimated conclusion is then derived in the second step according to the similarity between the observation and the antecedent of the artificial intermediate rule. The scale and move transformation-based FRI approach (T-FRI) [30, 31], which has been adopted as the foundation for the work in this thesis, belongs to this group.

As the two representatives for these two groups, the KH and T-FRI approaches are respectively reviewed in the following sections, with a focus on the T-FRI method as it is to be used as the basis in the implementation of this work due to its popularity. Other approaches are also to be outlined below..

2.5.1 The KH Approach

The KH approach [28] determines the conclusion by its α-cuts in such a way that the proportional distance between the estimated conclusion and the consequent sets of

the rules which are used should be the same as the distance between the observation and the antecedents of those rules, for all important α-cuts. The α-cut A_α of an fuzzy set A is a crisp set, denoted: $A_\alpha = \{x|A(x) \geq \alpha, \alpha \in (0,1]\}$.

2.5.1.1 Base Case of the KH Approach

The starting ideas are the Extension Principle and Resolution Principle. The former states that the solution of a problem for fuzzy sets can be found in the form of solving first for arbitrary α-cuts that are crisp sets and then extending the solution to the fuzzy case. The latter describes the decomposition of fuzzy sets to α-cuts

$$\mu_A(x) = sup\{\alpha : x \in A_\alpha\} \tag{2.9}$$

Every fuzzy set can be approximated with the use of the family of its α-cuts. Theoretically, all infinite cuts should be treated separately. In most practical cases, however, if the membership function is piecewise linear, it is often sufficient to calculate its α-cuts for only a few important or typical values [105], e.g., $\alpha = 0$ and $\alpha = 1$.

An important concept in the KH approach is the "less than" relation between two convex and normal fuzzy sets. Fuzzy set A_1 is said to be less than fuzzy set A_2, denoted by $A_1 \prec A_2$, if $\forall \alpha \in (0,1]$, the following conditions hold:

$$inf\{A_{1\alpha}\} < inf\{A_{2\alpha}\},\ sup\{A_{1\alpha}\} < sup\{A_{2\alpha}\} \tag{2.10}$$

where $A_{1\alpha}$ and $A_{2\alpha}$ are the α-cut sets of A_1 and A_2, respectively, $inf\{A_{i\alpha}\}$ is the infimum of $A_{i\alpha}$, and $sup\{A_{i\alpha}\}$ is the supremum of $A_{i\alpha}$, $i = 1, 2$.

For simplicity, suppose that two single-antecedent fuzzy rules are given as follows:

$$R_1 : \textit{If } x \textit{ is } A_1, \textit{ then } y \textit{ is } B_1$$
$$R_2 : \textit{If } x \textit{ is } A_2, \textit{ then } y \textit{ is } B_2$$

They are said to be *neighbouring rules* if and only if: (1) $A_1 \prec A_2$ or $A_2 \prec A_1$; and (2) there is no individual rule "If x is $A^{'}$, then y is $B^{'}$" such that $A_1 \prec A^{'} \prec A_2$ if $A_1 \prec A_2$, or $A_2 \prec A^{'} \prec A_1$ if $A_2 \prec A_1$.

To implement interpolation in the region between the antecedents of these two rules, i.e., to generate an approximated conclusion when an observation A^* located

between fuzzy sets A_1 and A_2 is hereby given. The neighbouring rules in a given rule base are therefore said to flank the observation. For the above two rules, this means that $A_1 \prec A^* \prec A_2$ or $A_2 \prec A^* \prec A_1$.

The KH approach uses the following equation to determine the interpolated result:

$$\frac{d(A^*, A_1)}{d(A^*, A_2)} = \frac{d(B^*, B_1)}{d(B^*, B_2)} \tag{2.11}$$

where A_1, A_2 are the antecedents of the two flanking rules, A^* is a given observation, B_1, B_2 are the consequences of those rules, B^* is the estimated conclusion, and $d(.,.)$ is typically the Euclidean distance between two fuzzy sets (though other distance metrics may be also used).

According to the Decomposition Principle, a convex and normal fuzzy set A can be represented by a series of α-cut intervals, each denoted as $A_\alpha, \alpha \in (0,1]$. In this case, Equation 2.11 can be written as:

$$\frac{d(A^*_\alpha, A_{1\alpha})}{d(A^*_\alpha, A_{2\alpha})} = \frac{d(B^*_\alpha, B_{1\alpha})}{d(B^*_\alpha, B_{2\alpha})} \tag{2.12}$$

where given any $\alpha(\alpha \in (0,1])$, the lower and upper distances between α-cuts $A_{1\alpha}$ and $A_{2\alpha}$ are defined:

$$\begin{cases} d_L(A_{1\alpha}, A_{2\alpha}) = d(inf\{A_{1\alpha}\}, inf\{A_{2\alpha}\}) \\ d_U(A_{1\alpha}, A_{2\alpha}) = d(sup\{A_{1\alpha}\}, sup\{A_{2\alpha}\}) \end{cases} \tag{2.13}$$

Note that the Euclidean distance between intervals can be defined in different ways but they all lie between $d_L(A_{1\alpha}, A_{2\alpha})$ and $d_U(A_{1\alpha}, A_{2\alpha})$. From Equation 2.13, Equation 2.12 can be rewritten as

$$\begin{aligned} \frac{d_L(A^*_\alpha, A_{1\alpha})}{d_L(A^*_\alpha, A_{2\alpha})} &= \frac{d_L(B^*_\alpha, B_{1\alpha})}{d_L(B^*_\alpha, B_{2\alpha})} \\ &= \frac{d_L(inf\{B^*_\alpha\}, inf\{B_{1\alpha}\})}{d_L(inf\{B^*_\alpha\}, inf\{B_{2\alpha}\})} \\ &= \frac{inf\{B^*_\alpha\} - inf\{B_{1\alpha}\}}{inf\{B_{2\alpha}\} - inf\{B^*_\alpha\}} \end{aligned} \tag{2.14}$$

Equation 2.14 can then be solved as follows:

$$\begin{aligned} inf\{B_\alpha^*\} &= \frac{inf\{B_{1\alpha}\}d_L(A_\alpha^*,A_{2\alpha}) + inf\{B_{2\alpha}\}d_L(A_\alpha^*,A_{1\alpha})}{d_L(A_\alpha^*,A_{2\alpha}) + d_L(A_\alpha^*,A_{1\alpha})} \\ &= \frac{\frac{inf\{B_{1\alpha}\}}{d_L(A_\alpha^*,A_{1\alpha})} + \frac{inf\{B_{2\alpha}\}}{d_L(A_\alpha^*,A_{2\alpha})}}{\frac{1}{d_L(A_\alpha^*,A_{1\alpha})} + \frac{1}{d_L(A_\alpha^*,A_{2\alpha})}} \end{aligned} \tag{2.15}$$

where $sup\{B_\alpha^*\}$ can be calculated in the same way, resulting in

$$\begin{cases} inf\{B_\alpha^*\} = \dfrac{\frac{inf\{B_{1\alpha}\}}{d_L(A_\alpha^*,A_{1\alpha})} + \frac{inf\{B_{2\alpha}\}}{d_L(A_\alpha^*,A_{2\alpha})}}{\frac{1}{d_L(A_\alpha^*,A_{1\alpha})} + \frac{1}{d_L(A_\alpha^*,A_{2\alpha})}} \\ sup\{B_\alpha^*\} = \dfrac{\frac{sup\{B_{1\alpha}\}}{d_U(A_\alpha^*,A_{1\alpha})} + \frac{sup\{B_{2\alpha}\}}{d_U(A_\alpha^*,A_{2\alpha})}}{\frac{1}{d_U(A_\alpha^*,A_{1\alpha})} + \frac{1}{d_U(A_\alpha^*,A_{2\alpha})}} \end{cases} \tag{2.16}$$

Alternatively, let

$$\begin{cases} \lambda_L = \dfrac{d_L(A_\alpha^*,A_{1\alpha})}{d_L(A_{2\alpha},A_{1\alpha})} \\ \lambda_U = \dfrac{d_U(A_\alpha^*,A_{1\alpha})}{d_U(A_{2\alpha},A_{1\alpha})} \end{cases} \tag{2.17}$$

The same solution can then be obtained but represented differently as follows:

$$\begin{cases} inf\{B_\alpha^*\} = (1-\lambda_L)inf\{B_{1\alpha}\} + \lambda_L inf\{B_{2\alpha}\} \\ sup\{B_\alpha^*\} = (1-\lambda_U)sup\{B_{1\alpha}\} + \lambda_U inf\{B_{2\alpha}\} \end{cases} \tag{2.18}$$

From this, $B_\alpha^* = [inf\{B_\alpha^*\}, sup\{B_\alpha^*\}]$ results. The estimated conclusion B^* can then be constructed by using the representation principle of fuzzy sets:

$$B^* = \bigcup_{\alpha\in(0,1]} \alpha B_\alpha^* \tag{2.19}$$

The most important advantage of the KH approach is its low computational complexity that ensures the fast response performance for real-time applications. Despite the rapid development of α-cut based FRI, there is a drawback in this group of methods.

Theoretically, all possible α-cuts (an infinite number) should be considered in performing the interpolation. However, the existing approaches in this group only take a finite number of α-cuts into consideration (usually 3 or 4). The resulting points are then connected by linear pieces to produce an approximation of the accurate conclusion.

2.5.2 The T-FRI Approach

The T-FRI approach [30, 31] can handle both interpolation and extrapolation of multiple multi-antecedent rules with triangular, complex polygon, Gaussian and bell-shaped fuzzy MFs. It has the following properties:

• It can handle both interpolation and extrapolation which involve multiple fuzzy rules, with each rule consisting of multiple antecedents.

• It guarantees the uniqueness as well as normality and convexity of the resulting interpolated fuzzy sets.

• It preserves piece-wise linearity such that interpolation can be computed using only characteristic points which describe a given polygonal convex fuzzy set, thereby ignoring any non-characteristic points and saving computation effort.

• It has been applied to problems such as truck backer-upper control and computer activity prediction.

2.5.2.1 Representative Value

A key concept used in the T-FRI approach is the representative value (*Rep*) of a given fuzzy set, it captures important information such as the overall location of a fuzzy set. Consider an arbitrary polygonal convex fuzzy set A with k points, which can be denoted as $A = (a_0, \ldots, a_{k-1})$. Given such an arbitrary polygonal convex fuzzy set, its general *Rep* is defined by

$$Rep(A) = \sum_{i=0}^{k-1} w_i a_i \tag{2.20}$$

where w_i is the weight assigned to the point a_i.

In general, the specification of the weights is necessary for a given application. Different definitions can be adopted for deriving different *Rep* values. The simplest

case is that all points take the same weighting value, i.e.,

$$Rep(A) = \frac{1}{k}\sum_{i=0}^{k-1} a_i \tag{2.21}$$

An alternative is the *weighted average Rep*, where the weights increase upwards from the bottom support to the top support, to reflect the relative significance of the fuzzy membership values. For instance, assuming the weights increase upwards from 0.5 to 1, such a *Rep* is defined by

$$Rep(A) = \frac{\sum_{i=0}^{\lceil \frac{k}{2} \rceil - 1} \frac{1+\mu_i}{2}(a_i + a_{k-i-1})}{\sum_{i=0}^{\lceil \frac{k}{2} \rceil - 1} \frac{1+\mu_i}{2}} \tag{2.22}$$

where μ_i is the membership value of a_i. Note that artificial odd points can be created to construct evenly paired odd points (as indicated previously), so $\mu_i = \mu_{k-i-1}$ can always be assumed.

One of the most widely used defuzzification methods, the *centre of core*, can also be utilised as an alternative. The *centre of core Rep* is solely determined by those points with a fuzzy membership value of 1:

$$Rep(A) = \frac{1}{2}(a_{\lceil \frac{k}{2} \rceil - 1} + a_{k - \lceil \frac{k}{2} \rceil}) \tag{2.23}$$

Based on the generated *Rep* values, the interpolation process is discussed in the following three cases. For simplicity, only rules involving triangular-shaped membership functions are considered.

2.5.2.2 The T-FRI Approach with Two Single-Antecedent Rules

Suppose that two neighbouring rules $A_1 \Rightarrow B_1, A_2 \Rightarrow B_2$ and an observation A^*, which is located between fuzzy sets A_1 and A_2, are given as follows:

$$\begin{gathered} R_1 : \textit{If } x_1 \textit{ is } A_1, \textit{ then } y_1 \textit{ is } B_1 \\ R_2 : \textit{If } x_2 \textit{ is } A_2, \textit{ then } y_2 \textit{ is } B_2 \\ O : x \textit{ is } A^* \\ C : y \textit{ is } B^* \end{gathered}$$

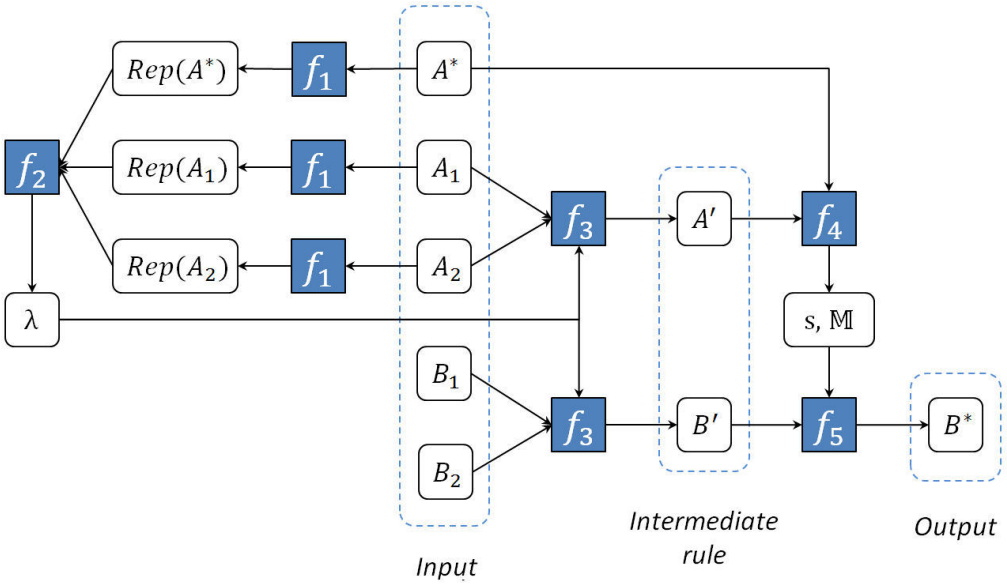


Figure 2.14: T-FRI with two single-antecedent rules

The desired conclusion B^* can be derived by interpolation. An intermediate rule $A' \Rightarrow B'$ is first constructed by manipulating these two given rules, where the intermediate term A' and the observation A^* have the same *Rep*, and so do the intermediate term B' and the desired B^*. Then B' is converted into B^* using scale and move transformations, which have been used to transform A' to A^*.

The interpolation process is illustrated in Figure 2.14. Given fuzzy sets A^*, A_1 and A_2, three parameters $Rep(A^*)$, $Rep(A_1)$ and $Rep(A_2)$ are produced with the function f_1. Next, the relative placement relation between the observation A^* and the antecedents (A_1 and A_2) of the two neighbouring rules is calculated by the function f_2, resulting in λ. From this, an intermediate rule $A' \Rightarrow B'$ is generated by applying the function f_3 with parameter λ to both the antecedents and consequences of the neighbouring rules. Then, the similarity degree between A' and A^* is computed by a predefined similarity measure. Specifically, scale rate s and move ratio $\mathbb{M}$ are exploited in scale and move transformation-based interpolation to represent the similarity degree, which is achieved by the function f_4. Finally, the estimated conclusion B^* is obtained by applying the function f_5 to B' while imposing the same similarity degree.

Intermediate Rule

The relative placement factor λ of the observation A^*, with respect to its two neighbouring rule antecedents A_1 and A_2, is defined by

$$\begin{aligned}\lambda &= \frac{d(A_1,A^*)}{d(A_1,A_2)} \\ &= \frac{d(Rep(A_1),Rep(A^*))}{d(Rep(A_1),Rep(A_2))}\end{aligned} \tag{2.24}$$

where $d(A_1,A_2) = d(Rep(A_1),Rep(A_2))$ represents the distance between two fuzzy sets A_1 and A_2, which is defined by

$$\begin{aligned}d(A_1,A_2) &= d(Rep(A_1),Rep(A_2)) \\ &= Rep(A_2) - Rep(A_1)\end{aligned} \tag{2.25}$$

where $Rep(A_1) \neq Rep(A_2)$ because $A_1 \prec A_2$ or $A_2 \prec A_1$. Such a factor reflects the relative location of the interpolated rule regarding the two neighbouring rules.

By using the simplest linear interpolation, the antecedent of the intermediate rule $A^{'} = (a_0^{'}, a_1^{'}, a_2^{'})$ can be calculated as follows:

$$\begin{cases} a_0^{'} = (1-\lambda)a_{10} + \lambda a_{20} \\ a_1^{'} = (1-\lambda)a_{11} + \lambda a_{21} \\ a_2^{'} = (1-\lambda)a_{12} + \lambda a_{22} \end{cases} \tag{2.26}$$

which are collectively abbreviated to

$$A^{'} = (1-\lambda)A_1 + \lambda A_2 \tag{2.27}$$

In so doing, the *Rep* of the calculated $A^{'}$ is guaranteed to be equal to that of A^*. The consequence of the intermediate rule $B^{'} = (b_0^{'}, b_1^{'}, b_2^{'})$ can then be obtained similar to the calculation of $A^{'}$ using the same λ:

$$\begin{cases} b_0^{'} = (1-\lambda)b_{10} + \lambda b_{20} \\ b_1^{'} = (1-\lambda)b_{11} + \lambda b_{21} \\ b_2^{'} = (1-\lambda)b_{12} + \lambda b_{22} \end{cases} \tag{2.28}$$

with abbreviated notation

$$B^{'} = (1-\lambda)B_1 + \lambda B_2 \tag{2.29}$$

Scale and Move Transformations

As $A^{'} \Rightarrow B^{'}$ is derived from $A_1 \Rightarrow B_1$ and $A_2 \Rightarrow B_2$, it is feasible to perform fuzzy reasoning with this new rule without further reference to its originals. Given such an intermediate rule and an observation, the conclusion can be calculated with respect to the following intuition:

The more similar $A^{'}$ to A^*, the more similar $B^{'}$ to B^*.

Suppose that a certain degree of similarity between the antecedent parts $A^{'}$ and A^* is established, it is intuitive to require that the consequent parts $B^{'}$ and B^* attain the same similarity degree. Hence, the following two transformations are used to ensure this.

Scale Transformation The similarity degree between $A^{'}$ and A^* is first measured by scale rate s, which is defined by

$$s = \frac{a_2^* - a_0^*}{a_2^{'} - a_0^{'}} \tag{2.30}$$

Let $A^{''} = (a_0^{''}, a_1^{''}, a_2^{''})$ denote the second intermediate term generated by the scale transformation. By using s, the current support $(a_0^{'}, a_2^{'})$ is transformed into a new support $(a_0^{''}, a_2^{''})$, while keeping the *Rep* and the ratio of the left-support $(a_0^{''}, a_1^{''})$ to the right-support $(a_1^{''}, a_2^{''})$ of $A^{''}$ the same as those of its original, such that

$$\begin{cases} a_2^{''} - a_0^{''} = s(a_2^{'} - a_0^{'}) \\ \dfrac{a_0^{''} + a_1^{''} + a_2^{''}}{3} = \dfrac{a_0^{'} + a_1^{'} + a_2^{'}}{3} \\ \dfrac{a_1^{''} - a_0^{''}}{a_2^{''} - a_1^{''}} = \dfrac{a_1^{'} - a_0^{'}}{a_2^{'} - a_1^{'}} \end{cases} \tag{2.31}$$

$A^{''}$ can then be calculated by solving Equation 2.31:

$$\begin{cases} a_0^{''} = \dfrac{a_0^{'}(1+2s) + a_1^{'}(1-s) + a_2^{'}(1-s)}{3} \\ a_1^{''} = \dfrac{a_0^{'}(1-s) + a_1^{'}(1+2s) + a_2^{'}(1-s)}{3} \\ a_2^{''} = \dfrac{a_0^{'}(1-s) + a_1^{'}(1-s) + a_2^{'}(1+2s)}{3} \end{cases} \tag{2.32}$$

This measure reflects the similarity degree between $A^{'}$ and A^{*}: the closer is s to 1, the more similar is $A^{'}$ to A^{*}. It is therefore used to act as, or to contribute to, the desirable similarity degree in order to transform $B^{'}$ to B^{*}.

Move Transformation The similarity degree is further measured by move ratio $\mathbb{M}$. By using $\mathbb{M}$, the current support $(a_0^{''}, a_2^{''})$ of $A^{''}$ is moved to (a_0^{*}, a_2^{*}) while keeping its *Rep*, resulting in the observation A^{*}. The move ratio $\mathbb{M}$ is defined by

$$\mathbb{M} = \begin{cases} \dfrac{a_0^{*} - a_0^{''}}{\dfrac{a_1^{''} - a_0^{''}}{3}} & if\ a_0^{*} \geq a_0^{''} \\ \\ \dfrac{a_0^{*} - a_0^{''}}{\dfrac{a_2^{''} - a_1^{''}}{3}} & otherwise \end{cases} \tag{2.33}$$

Given $\mathbb{M}$, A^{*} can then be retrieved as:

$$\begin{cases} \begin{cases} a_0^{*} = a_0^{''} + \mathbb{M}\dfrac{a_1^{''} - a_0^{''}}{3} \\ a_1^{*} = a_1^{''} - 2\mathbb{M}\dfrac{a_1^{''} - a_0^{''}}{3} \\ a_2^{*} = a_2^{''} + \mathbb{M}\dfrac{a_1^{''} - a_0^{''}}{3} \end{cases} & if\ \mathbb{M} \geq 0 \\ \\ \begin{cases} a_0^{*} = a_0^{''} + \mathbb{M}\dfrac{a_2^{''} - a_1^{''}}{3} \\ a_1^{*} = a_1^{''} - 2\mathbb{M}\dfrac{a_2^{''} - a_1^{''}}{3} \\ a_2^{*} = a_2^{''} + \mathbb{M}\dfrac{a_2^{''} - a_1^{''}}{3} \end{cases} & otherwise \end{cases} \tag{2.34}$$

This reflects the similarity degree between $A^{'}$ and A^{*}: the closer is $\mathbb{M}$ to 0, the more similar is $A^{'}$ to A^{*}.

Having obtained the similarity degree between $A^{'}$ and A^{*}, the interpolated conclusion B^{*} can therefore be obtained by transforming $B^{'}$ with the same scale rate s and move ratio $\mathbb{M}$.

2.5.2.3 The T-FRI Approach with Two Multi-Antecedent Rules

Two multi-antecedent rules interpolation is a generalisation of the two single antecedent rules interpolation. Given an observation such that

$$O : x_1 \text{ is } A_1^*, \cdots, x_j \text{ is } A_j^*, \cdots, x_M \text{ is } A_M^*$$

Suppose that two neighbouring rules are used for interpolation with respect to the given observation, which are represented by

$$R_1 : \text{If } x_1 \text{ is } A_{11}, \cdots, x_j \text{ is } A_{1j}, \cdots, x_M \text{ is } A_{1M}, \text{ then } y \text{ is } B_1$$
$$R_2 : \text{If } x_1 \text{ is } A_{21}, \cdots, x_j \text{ is } A_{2j}, \cdots, x_M \text{ is } A_{2M}, \text{ then } y \text{ is } B_2$$

where M is the number of antecedent variables.

When one rule involves multiple antecedent variables, each antecedent dimension will have its own parameter values for λ, s and $\mathbb{M}$. Obviously, all these values contribute to the construction of the intermediate term $B^{'}$ and the desired B^*. The following equations aggregate all of these values in order to construct the intermediate term $B^{'}$. The interpolated conclusion B^* can then be obtained by using $s^{'}$ and $\mathbb{M}^{'}$, where

$$\lambda^{'} = \frac{1}{M}\sum_{j=1}^{M}\lambda_j \tag{2.35}$$

$$B^{'} = (1-\lambda^{'})B_1 + \lambda^{'}B_2 \tag{2.36}$$

$$s^{'} = \frac{1}{M}\sum_{j=1}^{M}s_j \tag{2.37}$$

$$\mathbb{M}^{'} = \frac{1}{M}\sum_{j=1}^{M}\mathbb{M}_j \tag{2.38}$$

and M is the number of antecedent variables.

The process of the T-FRI with two multi-antecedent rules is illustrated in Figure 2.15. In this figure, there are M repeated components which are identical to the core of the two single-antecedent rules interpolation (as shown in Figure 2.14). Each of these components does exactly the same as the common core of the single-antecedent situation. That is, the relative placement factors $\lambda_j (j = 1, ..., M)$ are

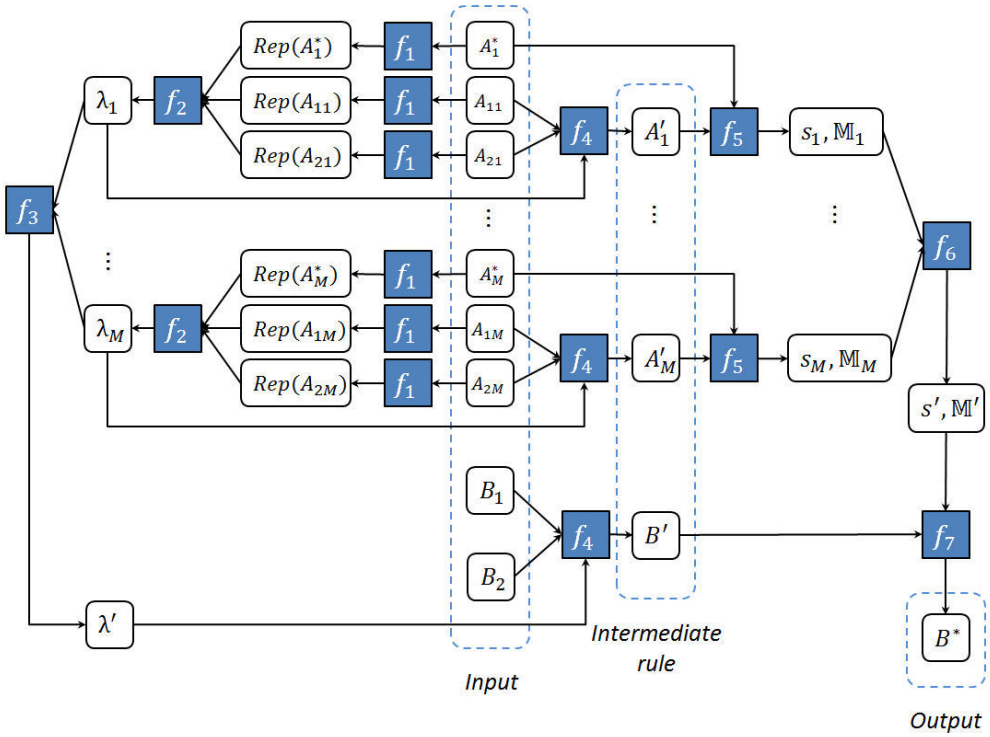


Figure 2.15: T-FRI with two multi-antecedent rules

calculated from each term of the observation A_j^* and the corresponding two fuzzy sets A_{1j} and A_{2j} . The function f_3 is then introduced to combine all these λ_j to a single parameter $\lambda^{'}$, resulting in the consequence of the intermediate rule. Similarly, the scale rates s_j and the move ratios $\mathbb{M}_j (j = 1, ..., M)$ are combined to $s^{'}$ and $\mathbb{M}^{'}$ by using the function f_6.

2.5.2.4 The T-FRI Approach with Multiple Multi-Antecedent Rules

In order to implement interpolation or extrapolation with multiple multi-antecedent rules, the first step is to choose $N (N \geq 2)$ rules from a given rule base. Then, an intermediate rule is constructed based on the selected rules. Once the intermediate rule is worked out, the remainder of the process remains the same as that described in the previous sections. The key steps in generating an intermediate rule are briefly introduced as follows.

Closest N Rules Selection

Without loss of generality, suppose that a rule R_i and an observation O are represented by

$$R_i : \textit{If } x_1 \textit{ is } A_{i1}, \cdots, x_j \textit{ is } A_{ij}, \cdots, x_M \textit{ is } A_{iM}, \textit{ then } y \textit{ is } B_i$$

$$O : x_1 \textit{ is } A_1^*, \cdots, x_j \textit{ is } A_j^*, \cdots, x_M \textit{ is } A_M^*$$

where A_{ij} denotes the j-th antecedent fuzzy set of Rule R_i , A_j^* denotes the observed fuzzy set of variable x_j , and B_i denotes the consequent fuzzy set of Rule R_i with $j \in \{1, \ldots, M\}$, M being the number of antecedent variables.

The distances d_{ij} between the pairs of A_{ij} and A_j^* can be calculated as follows:

$$\begin{aligned} d_{ij} &= d(A_{ij}, A_j^*) \\ &= d(Rep(A_{ij}), Rep(A_j^*)) \end{aligned} \tag{2.39}$$

The distance d_i between the rule R_i and the observation O is deemed to be the average of all antecedent variables' distances:

$$d_{ij}^{'} = \frac{d_{ij}}{max_j - min_j} \tag{2.40}$$

$$d_i = \sqrt{\sum_{j=1}^{M} {d_{ij}^{'}}^2} \tag{2.41}$$

where max_j and min_j are the maximum and minimum values of $x_j, j \in \{1, \ldots, M\}$. Each distance measure d_{ij} is normalised into the range $[0, 1]$, denoted by $d_{ij}^{'}$, to make the absolute distances compatible with each other over different domains. Note that if $max_j - min_j = 0$, then $max_j = min_j$. That is, A_j^* of O is identical with A_{ij} of $R_i, j \in \{1, \ldots, M\}$. In this case, $d_{ij}^{'} = 0$.

Intermediate Rule Construction

Suppose $N (N \geq 2)$ closest rules have been chosen from the observation. Such rules are represented as $R_i, i \in \{1, \ldots, N\}$, each has M antecedents $A_{ij}, j \in \{1, \ldots, M\}$. Let $w_{A_{ij}}$ denote the weight to which the j-th antecedent of the i-th rule contributes to the intermediate rule. The normalised weight $w_{A_{ij}}^{'}$ can be defined as:

$$w_{A_{ij}} = \frac{1}{d_{ij}} \tag{2.42}$$

$$w_{A_{ij}}^{'} = \frac{w_{A_{ij}}}{\sum_{i=1}^{N} w_{A_{ij}}} \tag{2.43}$$

Note that if $d_{ij} = 0$, then $Rep(A_{ij}) = Rep(A_j^*)$. In this case, the antecedent of the observation is considered to be identical to the corresponding antecedent of the rule R_i, in terms of the currently applied definition of *Rep*. Thus, $w_{A_{ij}} = 1$ for the identical ones, while $w_{A_{ij}} = 0$ for the remainder.

The antecedent of the so-called intermediate fuzzy term $A_j^{''}$ is constructed from the antecedents of these closest rules. Another process $shift$ is then introduced to modify $A_j^{''}$ to the antecedent of the intermediate rule $A_j^{'}$ so that it will have the same *Rep* as A_j^*:

$$A_j^{''} = \sum_{i=1}^{N} w_{A_{ij}^{'}} A_{ij} \tag{2.44}$$

$$A_j^{'} = A_j^{''} + \delta_{A_j}(max_j - min_j) \tag{2.45}$$

where δ_{A_j} is a constant defined by

$$\delta_{A_j} = \frac{Rep(A_j^*) - Rep(A_j^{''})}{max_j - min_j} \tag{2.46}$$

Note that if $max_j - min_j = 0$, then $max_j = min_j$. That is, A_j^* is identical with $A_j^{''}, j \in \{1, \ldots, M\}$. In this case, $\delta_{A_j} = 1$. Regarding the consequence of the intermediate rule $B^{'}$, it can be calculated by analogy to the computation of the antecedent, such that

$$B^{''} = \sum_{i=1}^{N} w_{B_i}^{'} B_i \tag{2.47}$$

$$B^{'} = B^{''} + \delta_B(max - min) \tag{2.48}$$

where $B^{''}$ is the consequence of the intermediate fuzzy term, max and min are the maximum and minimum values of consequent variable, $w_{B_i}^{'}$ and δ_B are the means of $w_{A_{ij}}^{'}$ and $\delta_{A_j}, i \in \{1, \ldots, N\}, j \in \{1, \ldots, M\}$, respectively, which are defined as:

$$w_{B_i}^{'} = \frac{1}{M} \sum_{j=1}^{M} w_{A_{ij}^{'}} \tag{2.49}$$

$$\delta_B = \frac{1}{M} \sum_{j=1}^{M} \delta_{A_j} \tag{2.50}$$

Then, the intermediate rule is constructed as

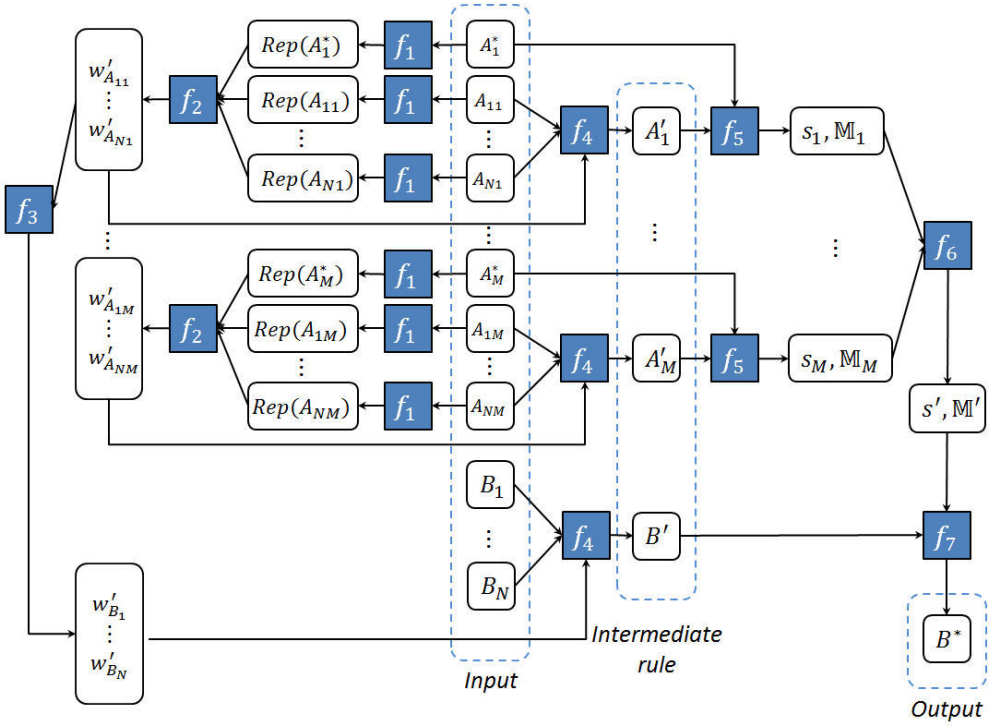


Figure 2.16: T-FRI with mutilple multi-antecedent rules

$$If\ x_1\ is\ A_1^{'},\ \cdots,\ x_j\ is\ A_j^{'},\ \cdots,\ x_M\ is\ A_M^{'},\ then\ y\ is\ B^{'}$$

Having generated the required intermediate rule, the rest of the interpolation involves firing this rule by the given observation, which is the same as that of interpolation with two rules described previously. The process of the T-FRI with multiple multi-antecedent rules is illustrated in Figure 2.16. In addition, extrapolation is a special case of interpolation when all the N closest rules lie on one side of the given observation. However, the processes of choosing the closest rules and constructing the intermediate rule are carried out in exactly the same way as the procedures for interpolation.

2.5.3 Other Approaches

In addition to the aforementioned approaches, a number of other existing approaches have also been reported in the literature [92, 103, 115, 116, 117, 118], several of them are reviewed in the following sections. For details of other implementations, refer to the corresponding references given.

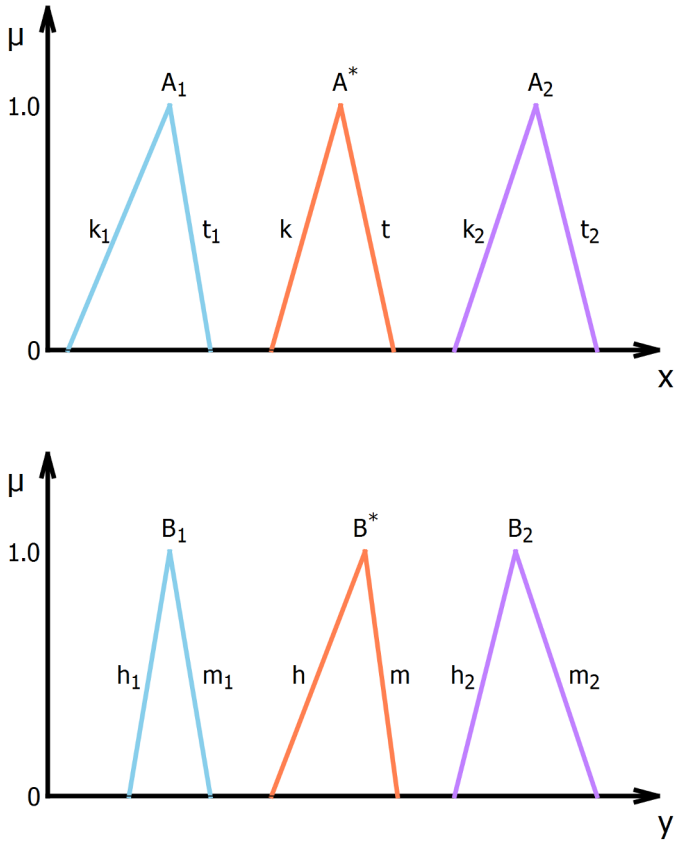


Figure 2.17: HCL interpolation

2.5.3.1 HCL Interpolation

The HCL (Hsiao, Chen, and Lee) approach [92] eliminates the abnormal problem by fixing the core of the consequence generated by the KH approach and shifting its support along with the consequent variable axis. It represents both slopes of each fuzzy set as a linear function. The slopes of the consequent fuzzy set are also linear functions whose parameters are interpolated from those of the observation and the fuzzy sets involved in the rule bases. A ratio between the left slope and the right slope of the consequence is then calculated and utilised to shift the support of the generated consequence by the KH approach in reference to the normal point of the consequence. Unfortunately, this approach is only applicable to triangular fuzzy sets.

The typical interpolation problem is shown in Figure 2.17, where k_1, t_1, k, t, k_2, t_2, h_1, m_1, h, m, h_2, and m_2 represent the slopes of the corresponding fuzzy sets. The HCL approach calculates the support of B^* in the same way as the KH approach but

the top point is calculated in a different way. The process to determine the top point of B^* is described below.

The slopes h and m of B^* are calculated first. Let:

$$\begin{cases} k = k_1 x + k_2 y \\ t = t_1 x + t_2 y \end{cases} \tag{2.51}$$

where x and y are real numbers. If $\frac{k_1}{t_1} \neq \frac{k_2}{t_2}$, then x and y are computed by Equation 2.51. Let:

$$\begin{cases} h = c|h_1 x + h_2 y| \\ m = -c|m_1 x + m_2 y| \end{cases} \tag{2.52}$$

where c is a constant. Otherwise, let:

$$\begin{cases} h = ck \\ m = ct \end{cases} \tag{2.53}$$

where c is a constant.

The position of the top point of B^* is then decided by

$$\frac{CP(B^*) - sup(B^*)}{CP(B^*) - inf(B^*)} = \frac{h}{m} \tag{2.54}$$

where $CP(B^*)$ denotes the centre point of core of B^*. Equation 2.54 can be recalculated as

$$CP(B^*) = \frac{sup(B^*)m - inf(B^*)h}{m - h} \tag{2.55}$$

Not that if $m = h$, then $sup(B^*) = inf(B^*)$ can be derived from Equation 2.54. In this case, $CP(B^*) = sup(B^*) = inf(B^*)$.

2.5.3.2 CCL Interpolation

The CCL (Chang, Chen, and Liau) approach [26] can be seen as an improvement of the HCL approach. This approach first determines the core of the consequence by using the KH approach, which is calculated as follows:

$$b^* = b_1 + \frac{(a^* - a_1)(b_2 - b_1)}{a_2 - a_1} \tag{2.56}$$

where a_1, a_2, a^*, b_1, b_2, and b^* are the normal points of the involved triangular fuzzy sets A_1,A_2,A^*,B_1,B_2, and B^*, respectively.

The areas of the two sides of the core are then calculated from the corresponding areas of the given observation and all the fuzzy sets involved in the rules used for interpolation in a manner of linear interpolation.

$$S_K(B^*) = \begin{cases} S_K(A^*) \sum_{i=1}^{2} W_i \frac{S_K(B_i)}{S_K(A_i)} & if \ \exists i \ \ S_K(A_i) > 0 \\ S_K(A^*) & if \ \forall i \ \ S_K(A_i) = 0 \end{cases} \tag{2.57}$$

where $K \in L, R, S_L(B^*)$ and $S_R(B^*)$ denote the left and right area of B^*, respectively, and

$$\begin{cases} W_1 = 1 - \frac{a^* - a_1}{a_2 - a_1} \\ W_2 = 1 - W_1 \end{cases} \tag{2.58}$$

The interpolated result B^* is therefore derived by

$$B^* = (b^* - 2S_L(B^*), b^*, b^* + 2S_R(B^*)) \tag{2.59}$$

Unlike the HCL approach, this approach is able to deal with interpolation and extrapolation with multiple multi-antecedent rules, with each rule involving any shape of fuzzy sets.

2.5.3.3 QMY Interpolation

The QMY (Qiao, Mizumoto, and Yan) approach [118] employs the same mechanism for generating intermediate rules as the T-FRI approach, but the *Rep* is restricted to being the centre point of core. The similarity degree between the observation A^* and the antecedent $A^{'}$ of the intermediate rule is captured using the so-called parameters *lower similarity* and *upper similarity*, which are defined by

$$\begin{cases} S_{L(A^*,A')}(\alpha) = \frac{d(inf(A^*_\alpha), CP(A^*))}{d(inf(A'_\alpha), CP(A^*))} \\ S_{U(A^*,A')}(\alpha) = \frac{d(sup(A^*_\alpha), CP(A^*))}{d(sup(A'_\alpha), CP(A^*))} \end{cases} \tag{2.60}$$

where $\alpha \in (0, 1]$.

With reference to the centre point of the core, a convex and normal fuzzy set can be divided into two parts, namely, the lower part and the upper part. The *lower*

similarity measures the difference of the lower parts of two fuzzy sets, by comparing the lengths of a certain level cut, and the *upper similarity* does that of the upper parts.

In so doing, the consequence B^* is derived from the following equations:

$$\begin{cases} CP(B^*) = CP(B^{'}) \\ S_{L(B^*,B')}(\alpha) = S_{L(A^*,A')}(\alpha) \\ S_{U(B^*,B')}(\alpha) = S_{U(A^*,A')}(\alpha) \end{cases} \tag{2.61}$$

Combining Equations 2.60 and 2.61 gives

$$\begin{cases} inf(B^*_\alpha) = S_{L(A^*,A')} d(inf(B^{'}_\alpha), CP(B^{'})) + CP(B^{'}) \\ sup(B^*_\alpha) = S_{U(A^*,A')} d(sup(B^{'}_\alpha), CP(B^{'})) + CP(B^{'}) \end{cases} \tag{2.62}$$

Thus B^* can be calculated with the representation principle of fuzzy sets.

2.5.3.4 CK Interpolation

The CK (Chen and Ko) approach [119] ensures that the core of each fuzzy set of a created intermediate rule is equal to that of the corresponding fuzzy set of the resultant interpolated rule.

First, the *Reps* of all the involved fuzzy sets are obtained by the T-FRI approach, resulting in the parameter λ. The values of $la^{'}_{0,1}$ and $la^{'}_{1,2}$ are then calculated:

$$\begin{cases} la^{'}_{0,1} = (1-\lambda) la_{1_{0,1}} + \lambda la_{2_{0,1}} \\ la^{'}_{1,2} = (1-\lambda) la_{1_{1,2}} + \lambda la_{2_{1,2}} \end{cases} \tag{2.63}$$

where $la^{'}_{0,1}$ and $la^{'}_{1,2}$ denote the left and the right support length of the antecedent of the intermediate rule. The values of $lb^{'}_{0,1}$ and $lb^{'}_{1,2}$ can be calculated in the same way.

Next, the antecedent of the intermediate rule is constructed:

$$\begin{cases} a^{'}_0 = a^{'}_1 - la^{'}_{0,1} \\ a^{'}_1 = a_1 \\ a^{'}_2 = a^{'}_1 + la^{'}_{1,2} \end{cases} \tag{2.64}$$

Similarly, the consequence of the intermediate rule can be constructed by means of the previously obtained $lb'_{0,1}$ and $lb'_{1,2}$

$$\begin{cases} b'_0 = b'_1 - lb'_{0,1} \\ b'_1 = b_1 \\ b'_2 = b'_1 + lb'_{1,2} \end{cases} \tag{2.65}$$

where b_1 is the core of the estimated interpolated conclusion, which is determined as follows:

$$b_1 = (1 - \lambda_{a_1}) Rep(B_1) + \lambda_{a_1} Rep(B_2) \tag{2.66}$$

where

$$\begin{aligned} \lambda_{a_1} &= \frac{d(A_1, a_1)}{d(A_1, A_2)} \\ &= \frac{a_1 - Rep(A_1)}{Rep(A_2) - Rep(A_1)} \end{aligned} \tag{2.67}$$

Note that $Rep(A_1) \neq Rep(A_2)$ because $A_1 \prec A_2$ or $A_2 \prec A_1$

In order to measure the similarity degree between two fuzzy sets with the same core, only their left slopes and right slopes need to be compared. Two transformations, i.e., increment transformation and ratio transformation are then utilised for this purpose, with one aiming to increase the length of a certain level cut of a slope during the transformation, and the other to decrease the length. From this, $B^* = (b_0^*, b_1^*, b_2^*)$ can be derived, where b_0 and b_2 are calculated as

$$b_0 = \begin{cases} b_1 - la_{0,1} \dfrac{a_{20} - a_{12}}{b_{20} - b_{12}} + la'_{0,1} \dfrac{a_{20} - a_{12}}{b_{20} - b_{12}} - lb'_{0,1} & if\ la_{0,1} \geq la'_{0.1} \\ b_1 - \dfrac{la_{0,1} lb'_{0,1}}{la'_{0,1}} & otherwise \end{cases} \tag{2.68}$$

$$b_2 = \begin{cases} b_1 - la_{1,2} \dfrac{a_{20} - a_{12}}{b_{20} - b_{12}} + la'_{1,2} \dfrac{a_{20} - a_{12}}{b_{20} - b_{12}} - lb'_{1,2} & if\ la_{1,2} \geq la'_{1,2} \\ b_1 - \dfrac{la_{1,2} lb'_{1,2}}{la'_{1,2}} & otherwise \end{cases} \tag{2.69}$$

2.5.4 Summary of Comparison of Typical Interpolation Methods

For all the outlined typical interpolation methods, comparisons are summarised based on the identified criteria in Table 2.2. This comparison shows that all the methods satisfy the following key criteria: preservation of neighbouring quality (PNQ), mapping similarity (MS), multiple rules for support (MRS), rule-base preservation/ modus ponens validity (RBP), approximation capability (AC), and fuzziness of inferred result (FIR). Two criteria: avoidance of abnormal conclusion (AAC), and preservation of convexity and normality (CNF) are fulfilled by : HCL, QMY, T-FRI and CK. Multiple antecedent variables for support (MAVS) is not supported by HCL The KH method has the lowest computational complexity.

Table 2.2: Summary of Typical Interpolation Methods with regards to Evaluation Criteria

	PNQ	MS	MRS	MAVS	RBP	AC	FIR	AAC	CNF
KH	Yes	Yes	Yes	Yes	Yes	Yes	Yes	NO	No
HCL	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
T-FRI	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
QMY	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CK	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

2.6 Summary

This chapter has presented a systematic review of typical deep machine learning architectures and also lists their real world applications as on-going research. This chapter has also introduced basic concepts of and recent developments in fuzzy inference systems. Generally the implementations of fuzzy inference techniques can be categorised into two groups: CRI and FRI. The former works on a dense rule base and a number of typical methods have been reviewed. The latter can tackle the problem that CRI can not be used when a sparse rule base is presented. FRI can be categorised into two groups: one interpolating the consequence directly from a given observation, and the other following a two step approach. The latter approach first generates an intermediate rule such that its antecedent part is as close to the given observation as possible, and then this intermediate rule is fired by the given observation through similarity-based fuzzy reasoning. The original KH approach

and the T-FRI approach have been taken as representatives of the two groups in this chapter. The implementations of both approaches have been discussed, including the basic case, multiple antecedents case, and multiple rules case. Further, a review has been provided for the typical techniques that were developed in order to modify and improve the KH approach that may arrive a result which is not an fuzzy set.

Chapter 3

Interpolating DeSTIN Features for Image Classification

One critical step to successfully build an image classifier is to extract and use informative features from given images [43, 44]. However, generating more features increases computational complexity, which will cause practical difficulties, especially in the need of performing real-time classification tasks. It is desirable to develop a method that can work with limited generation of necessary features without causing a problem in conducting accurate classification.

A possible way to do this is to take a biologically-inspired approach, based on the argument that the neo-cortex does not explicitly pre-process sensory signals, but rather allows them to propagate through a hierarchy of modules which incrementally learn to represent observed regularities they exhibit [39, 40, 50, 51, 58]. Under this assumption, it would be possible to train such a hierarchical network on a large set of observations and later extract features from the network to be fed to a classification system for the purpose of robust pattern recognition. Deep Spatio-Temporal Inference Network (DeSTIN) [5, 120, 121] is a good example of this approach. Note that robustness here refers to the ability to exhibit invariance to a diverse range of transformations and distortions, including noise, scale, rotation, and lighting conditions.

The extracted features minimise the computational complexity to a certain extent. Unfortunately, the underlying technique itself introduces significant computation,

which may well offset the potential benefit on the efficiency in performing the entire feature extraction process by the use of DeSTIN. An alternative approach is to employ interpolation to produce a more informative feature set, which improves the representation of the underlying images to be classified, thereby increasing the classification accuracy without sacrificing the efficiency.

This chapter presents a novel discriminative deep learning architecture that combines DeSTIN with interpolation, in an attempt to implement the above idea. This architecture, which is referred to DESTINI hereafter, leads to a highly scalable modelling system which is capable of extracting image features effectively and efficiently. Simulation results demonstrate that the framework is highly promising.

The rest of this chapter is organised as follows. Section 3.1 outlines the background of DeSTIN. Section 3.2 details the DESTINI approach, including an outline of the resulting computational algorithm for feature extraction. Section 3.3 gives a brief overview of Support Vector Machines (SVMs) that will be used to implement the image classifiers for experimental evaluation. Section 3.4 shows the experimental results, supported by comparative studies. This chapter is summarised in Section 3.5.

3.1 Deep Spatio-Temporal Inference Network

Deep Spatio-Temporal Inference Network (DeSTIN) [5] provides a scalable modelling system which is capable of effectively dealing with high-dimensional signals. Figure 3.1 shows the generic architecture of such a network. A DeSTIN comprises multiple instantiations of an identical cortical circuit (or node) which populate all layers of the network. Each node is tasked with characterising the sequences of patterns that are presented to it. In the very lowest layer of the hierarchy, the nodes receive as input the raw data (e.g., pixel values of the image), which is a high-dimensional signal

$$M_{n\times m} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix} \tag{3.1}$$

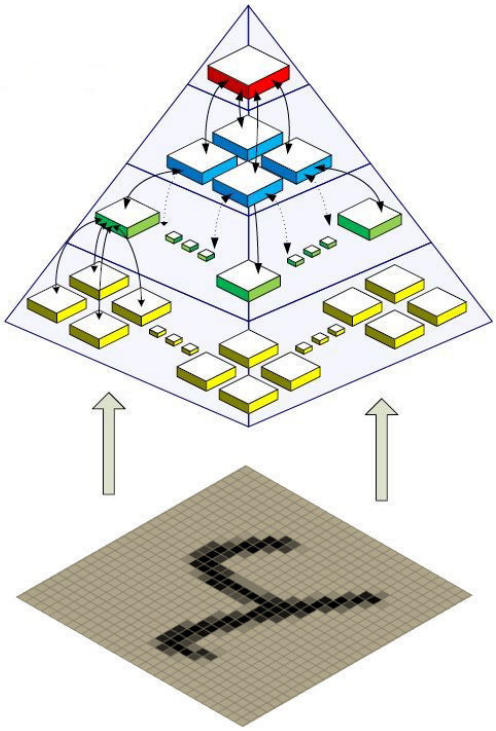


Figure 3.1: Topological architecture of DeSTIN

Each node continuously constructs a belief state that attempts to characterise the sequence of patterns received. The second layer, and each of those above it, receives the belief states of those nodes in the layer right below, and attempts to construct belief states that capture regularities in their inputs. The spatio-temporal dependencies that exist within different observations are modelled inherently in an unsupervised manner. Each node of the network represents sequences it observes by means of clustering, over the distribution of the sequences using Bayesian inference. For completeness the theoretical foundation of DeSTIN is outlined below; further details can be found in [5, 120, 121]. As a discriminative machine learning model, each node in DeSTIN aims at constructing an efficient belief state for the given stimuli it is presented with. The basic belief update rule, which governs the learning process and is identical for every node in the architecture, is discussed next. The belief state is a probability mass function over the sequences of stimuli that the nodes learns to represent. Consequently, each node is allocated a predefined number of state variables each denoting a dynamic pattern, or sequence, that is autonomously learned. The role that parental advice plays in the processing by writing the fundamental

belief update rule of DeSTIN as

$$b^{'}(s^{'}|a) = \frac{P_r(o|s^{'})\sum_{s\epsilon S} P_r(s^{'}|s,a)b(s)}{\sum_{s^{''}\epsilon S} P_r(o|s^{''})\sum_{s\epsilon S} P_r(s^{''}|s,c)b(s)} \tag{3.2}$$

which maps the current observation o, the belief b (with argument the system state s) and advice from a higher-layer node a, to a new (updated) belief and state $b^{'}(s^{'})$ at the next time step, with a normalization factor in the denominator. One interpretation of this equation is that the (static) pattern similarity metric, $Pr(o|s^{'})$, is modulated by a construct that reflects the system dynamics, $\sum_{s\epsilon S} P_r(s^{'}|s,a)b(s)$. (For shorthand, the latter is denoted as PSSA.) These two constructs are the main items which must be learned from the data. The dependence of the belief on the advice from the parent, which is important for the "multiple observer" model, is also considered. For the static pattern similarity metric, $Pr(o|s^{'})$, online clustering and mixture model formulations are applied. An online clustering algorithm based on the winner-take-all competitive learning approach is employed at the heart of each node's static pattern learning process. The algorithm includes constructs for improving performance and modulating the learning rate, but in this work a semi-constant learning rate is employed across a layer of nodes, with the rate set to learn fast for one layer and slower for subsequent (higher) layers until some stopping criteria is reached, at which point the rate is set to 0 for the "stopped" layer and adjusted to a fast rate for the next layer up.

For the system dynamics or PSSA, an advice generation methodology using tabular methods is applied. In the tabular method the transitions from s to $s^{'}$ given the advice a is simply counter. In past implementations, the advice or belief of the parent node, a, was chosen using the selection rule of $a = argmax b(s)$, but this was not robust to evolving belief conditions in the online learning process. Instead an online advice generation rule where each parental node examines the temporal sequence of input beliefs and performs unsupervised online clustering is used. The resulting label is then passed to the child nodes who use the advice to train the dynamic patterns. Furthermore, the clustering mechanisms described earlier provide good generalization but do not lend themselves well to a consistent labelling from movement to movement. The system dynamics can compensate for this problem to some degree, but some level of consistent labelling is necessary which is unfortunately not guaranteed with normal online clustering. Here the advice in a multiple-observer model to formulate a set of "belief in advice states" which serves as

a cumulative estimate of the belief in the advice state of a node is used. This allows each node to incorporate learning from the parent node over time to generalize the longer temporal and spatial scales, but retain a level of local knowledge and capture sufficient variation to make good supervised learning classifiers. In this mechanism, the system dynamics vary from movement to movement and each child node retains a brief history of its recent observations and beliefs within a single subject presentation. Then when the parental advice is available, the PSSA is learned across the past observations and movements. After training, each possible advice state is interpreted as a different observer, with multiple attempts at observation as well (one for each movement), and thus the belief in that advice state is computed as a cumulative prediction error $B(a) = \sum_m \sum_s b(s_m|a) - b(s_m)$, where s_m is the state at movement m. When there are a total of A advice states, this produces a vector of dimension A for the output of each node after the complete temporal sequence is observed. The advice component is passive, meaning the value for $B(a)$ and test for each advice state using the model residing in each node for each different advice state learned during the training process is simply computed. This approach is more robust in the sense that "good" advice is not required in real-time from the parent, and also occasional cases of near-zero probability do not cause the entire evolution to grind to a halt.

Let n be the cardinality of the centroid set defined for the top layer, then the output of DeSTIN can be represented by a vector of features, say, $V_{original} = (O_1, \cdots, O_n)^T$.

3.2 DESTINI

After propagating through a DeSTIN, the original image is converted to a vector of features, which can be represented as

$$V_{original} = (O_1, \cdots, O_n)^T \tag{3.3}$$

To perform effective image classification such a vector is expected to have a significant dimensionality. However, this means that the number of layers required is significant and hence will incur considerable computation. Thus, a trade off between subsequent classification effectiveness and computational effort in covering the original image into the feature vector is needed. This observation leads to the present development by applying simple interpolation methods to create additional elements

in an artificially expanded feature vector. That is, interpolation can be applied to $V_{original}$ in order to generate additional vector elements:

$$V_{additional} = (l_1, \cdots, l_m)^T \tag{3.4}$$

resulting in a final vector of features that jointly represent the original image:

$$V_D = (O_1, \cdots, O_n, l_1, \cdots, l_m)^T \tag{3.5}$$

3.2.1 Interpolation Methods

3.2.1.1 Linear Interpolation

The simplest possible interpolation is to linearly set $l_k = i_k$ $(1 \leq k \leq m)$ with i_k being local sums $O_j + O_{j+1}, 1 \leq j \leq n-1$, which is represented as $O_{j+(j+1)}$. On top of these local sums, the global sum $sum = \sum_{i=1}^{n} O_i$ can also be obtained in a straightforward manner. In so doing, the dimensionality of the additional feature vector can be increased to a number that is up to n. Obviously, such linear addition-based interpolation involves little computation.

Consider that if $m = 2$, which means that additional vectors of each containing two elements, $V_{additional} = (i_1, i_2)^T$ can be generated from the original vector produced by DeSTIN. In particular, i_1 may be set to be one element of the local sum set $\{O_{1+2}, O_{2+3}, \cdots, O_{(n-1)+n}\}$ or the global sum $sum = \sum_{i=1}^{n} O_i$. So there are n options, after choosing the value of i_1, the size of the optional set is reduced to $n-1$. For i_2, there are $n-1$ options. So if $m = 2$, there are C_n^2 different types of possible combination of the elements contained within the optional set to construct an additional feature vector.

In general, if $m = j,\ j \in \{2, 3, ..., n\}$, then j-dimensional vectors can be generated, with $i_k, (1 \leq k \leq m)$ representing the global sum $sum = \sum_{i=1}^{n} O_i$ or the local sum $O_{1+2}, O_{2+3}, \cdots, O_{(j-1)+j}, \cdots, O_{(n-1)+n}$. There are C_n^j different options of possible combination to form such additional vectors. Thus, if necessary, a total of $C_n^2 + C_n^3 + \cdots + C_n^{n-1} + C_n^n$ possible combinations can be generated when $V_{original}$ is an n-dimensional vector.

3.2.1.2 Newton Interpolation

In addition to linear interpolation (that is computationally least complex), another possible means to create artificial features is through Newton interpolation. The

basic idea is that given the values of $V_{original} = (O_1, \cdots, O_n)^T$, a set of $k+1$ data points $P = \{(x_0, y_0), (x_1, y_1), \ldots, (x_k, y_k)\}, (k \geq n)$ are created, such that the values of $\{x_0, x_1, \ldots, x_k\}$ are set to consecutively integer values, and the elements of $V_{original}$ are assigned as the values to a subset of $\{y_0, y_1, \ldots, y_k\}$, with the other $(k+1-n)$ missing values of this set to be generated via interpolation. To distinguish linearly interpolated features from those produced by the Newton method, the latter are denoted by I_j $(1 \leq j \leq k+1-n = m)$ below.

Formally, the Newton interpolation polynomial is of the form:

$$N(x) = \sum_{j=0}^{k} a_j n_j(x) \tag{3.6}$$

with the so-called Newton basis polynomials defined by

$$n_j(x) = \prod_{i=0}^{j-1} (x - x_i) \tag{3.7}$$

and the coefficients by

$$a_j = [y_0, \ldots, y_j] \tag{3.8}$$

where $[y_0, \ldots, y_j]$ denotes divided differences.

Given $k+1$ data points $(x_0, y_0), (x_1, y_1), \ldots, (x_k, y_k)$, the divided differences are defined as:

$$[y_v] = y_v, v \in \{0, \ldots, k\} \tag{3.9}$$

$$[y_v, \ldots, y_{v+j}] = \frac{[y_{v+1}, \ldots, y_{v+j}] - [y_v, \ldots, y_{v+j-1}]}{x_{v+j} - x_v} \tag{3.10}$$

where $v \in \{0, \ldots, k-j\}, j \in \{1, \ldots, k\}$.

Now, consider the specific case of $m = 1$, where additional 1-dimensional vectors $V_{additional} = (I_1)^T$ can be generated from the original vectors produced by DeSTIN. In this case, the dimensionality of data points is $n+1$. First, $\{x_0, x_1, \ldots, x_k\}$ are set to integer values consecutively; without losing generality, these can be represented as $\{x_0 = 1, x_1 = 2, \ldots, x_{k-1} = n, x_k = n+1\}$. Then, O_1 is set to becoming the value of $y_i, i \in \{0, 1, 2, ..., k\}$. So, there are $n+1$ options for such an assignment. After choosing the value of y_i, the size of the optional set is reduced to n. For O_2, there are n options, and so on. Thus, if $m = 1$, there are C_{n+1}^1 different kinds of possible combination of artificial features to form an additional feature vector.

In general, similar to the linear interpolation case, if $m = j,\ j \in \{2, 3, ..., n+1\}$, then j-dimensional vectors can be generated, with I_k $(1 \leq k \leq m)$ representing the Newton interpolated feature value. There are C_{n+1}^{j} different options of possible combination to construct the additional vector. Together, if required, a total of $C_{n+1}^{1} + C_{n+1}^{2} + \cdots + C_{n+1}^{n} + C_{n+1}^{n+1}$ possible combinations can be generated when $V_{original}$ is an n-dimensional vector.

3.2.2 The DESTINI Algorithm

A given image for classification is first processed by a DeSTIN of a low complexity. The resulting low-dimensional feature vector of the (relatively simple) DeSTIN is then translated into a feature set of a higher dimensionality and hence of potentially more discriminating power, through interpolation that leads to Equation 3.5. Reflecting the above-introduced straightforward interpolation mechanisms, l_j $(j \in \{1, 2, ..., m\})$ in V_D stands for i_j in linear interpolation and for I_j in Newton interpolation. The feature vectors so produced that consist of the original DeSTIN outputs and interpolated additional artificially created features are regarded as the returns of the DESTINI system, thereby computationally integrating DeSTIN and feature interpolation. From this, any of the generated V_D can be fed to an SVM (which acts as the image classifier, see below) for the purpose of robust image classification. This integrated use of DeSTIN and interpolation significantly increases the feature vector dimensionality without incurring much additional computation.

The working of DESTINI can be summarised as shown in Algorithm 3.2.1. Given a set of training data, the algorithm starts by assuming that the (initial) brief states b are set to default values (line 1). As the main body of the algorithm, it then runs an iteration loop (lines 2-6), in which an image is assigned to a matrix $M_{n\times m}$ (line 2), and then for each layer, an update is carried out to refine the brief state (line 3). Having updated all the layers, set the original features $V_{original}$ to be the output of DeSTIN (line 4), and interpolate it to build V_D (line 5). Repeat the loop until no image remaining to be processed (line 6). What is returned by this algorithm is the V_D constructed from a set of original features $M_{n\times m}$ (line 7).

The time complexity of DESTINI is mainly determined by two aspects: the time complexity of generating $V_{original} = (O_1, \cdots, O_n)^T$ and that of computing $V_{additional} = (l_1, \cdots, l_m)^T$. Consider that the core calculation for $V_{original}$ is the application of the

Algorithm 3.2.1: The DESTINI Algorithm

1: Initialise b, with all unclassified images to buffer
2: Load an image to $M_{n\times m}$
3: For each layer of DeSTIN, update b to $b^{'}$
4: Set $V_{original}$ to the output of DeSTIN $(O_1,\cdots,O_n)^T$
5: Interpolate $V_{original}$ to derive $V_D=(O_1,\cdots,O_n,l_1,\cdots,l_m)^T$
6: If every image has been processed go to the next, else go to Step 2
7: Return $V_D=(O_1,\cdots,O_n,l_1,\cdots,l_m)^T$

Euclidean distance metric, so the time complexity of computing $V_{original}$ is $O(n^2)$. If $V_{additional}$ is generated by linear interpolation, the time complexity for $V_{additional}$ is $O(m)$. Together the time complexity of producing $V_D=(O_1,\cdots,O_n,l_1,\cdots,l_m)^T$, is $O(n^2)+O(m)$. If however, $V_{additional}$ is generated by Newton interpolation, the time complexity for obtaining $V_{additional}$ is $O(m^2)$. Together the time complexity of producing V_D is $O(n^2)+O(m^2)$. Yet, to generate an original feature vector of the same dimensionality, which is $(m+n)$, the time complexity of DeSTIN is $O((m+n)^2)$. Given the same dimensionality, it is clearly more efficient for DESTINI to generate V_D.

3.2.3 Generic Worked Examples

To illustrate the basic idea of DESTINI, consider the case where the output of DeSTIN is a 3-dimensional vector:

$$V_{original}=(O_1,O_2,O_3)^T \tag{3.11}$$

Thus, the following additional interpolated vectors

$$V_{additional}=(i_1,\cdots,i_m)^T, 2\le m\le 3 \tag{3.12}$$

can be generated by linear interpolation, plus the global feature $sum=\sum_{i=1}^{3}O_i$.

If $m=2$, which means additional vectors of 2 dimensions $V_{additional}=(i_1,i_2)^T$ can be generated from the original vector. So $i_k, k\in\{1,2\}$, represents O_{1+2}, O_{2+3} or $sum=\sum_{i=1}^{3}O_i$, and there are C_3^2 =3 different kinds of combination:

$$(i_1=O_{1+2}, i_2=O_{2+3})^T \tag{3.13}$$

$$(i_1=O_{1+2}, i_2=sum)^T \tag{3.14}$$

$$(i_1 = O_{2+3}, i_2 = sum)^T \tag{3.15}$$

Consequently, the combined V_D may be either of the following three:

$$(O_1, O_2, O_3, O_{1+2}, O_{2+3})^T \tag{3.16}$$

$$(O_1, O_2, O_3, O_{1+2}, sum)^T \tag{3.17}$$

$$(O_1, O_2, O_3, O_{2+3}, sum)^T \tag{3.18}$$

If $m = 3$, then a 3-dimensional additional feature vector consisting of the following can be produced: $i_1 = O_{1+2}$, $i_2 = O_{2+3}$, $i_3 = sum$. Thus, V_D represents

$$V_D = (O_1, O_2, O_3, O_{1+2}, O_{2+3}, sum)^T \tag{3.19}$$

Together, out of the original 3-dimensional feature vector, $C_3^2 + C_3^3 = 4$ combinations can be created to act as the artificially generated additional features.

Now, consider an example using Newton interpolation with the same 3-dimensional original feature vector. In this case, the following additional interpolated vectors can be generated:

$$V_{additional} = (I_1, \cdots, I_m)^T, 1 \leq m \leq 4 \tag{3.20}$$

If $m = 1$, which means that additional vectors of 1 dimension $V_{additional} = (I_1)^T$ are required to be generated.

Thus, the set of data points is $P = \{(1, y_0), (2, y_1), (3, y_2), (4, y_3)\}$, which may be either of:

$$\{(1, I_1), (2, O_1), (3, O_2), (4, O_3)\} \tag{3.21}$$

$$\{(1, O_1), (2, I_1), (3, O_2), (4, O_3)\} \tag{3.22}$$

$$\{(1, O_1), (2, O_2), (3, I_1), (4, O_3)\} \tag{3.23}$$

$$\{(1, O_1), (2, O_2), (3, O_3), (4, I_1)\} \tag{3.24}$$

There are therefore $C_4^1 = 4$ different kinds of combination.

If $m = 2$, additional vectors of 2 dimensions $V_{additional} = (I_1, I_2)^T$ can be generated.

The set of artificially created data points is

$P = \{(1, y_0), (2, y_1), (3, y_2), (4, y_3), (5, y_4)\}$, which may be either of:

$$\{(1, I_1), (2, O_1), (3, I_2), (4, O_2), (5, O_3)\} \tag{3.25}$$

$$\{(1, I_1), (2, O_1), (3, O_2), (4, I_2), (5, O_3)\} \tag{3.26}$$

$$\{(1, I_1), (2, O_1), (3, O_2), (4, O_3), (5, I_2)\} \tag{3.27}$$

$$\{(1, O_1), (2, I_1), (3, O_2), (4, I_2), (5, O_3)\} \tag{3.28}$$

$$\{(1, O_1), (2, I_1), (3, O_2), (4, O_3), (5, I_2)\} \tag{3.29}$$

$$\{(1, O_1), (2, O_2), (3, I_1), (4, O_3), (5, I_2)\} \tag{3.30}$$

forming the $C_4^2 = 6$ different kinds of combination.

If $m = 3$, additional feature vectors of 3 dimensions $V_{additional} = (I_1, I_2, I_3)^T$ can be created. The set of data points is $P = \{(1, y_0), (2, y_1), (3, y_2), (4, y_3), (5, y_4), (6, y_5)\}$, which may be either of the C_4^3 =4 different kinds of combination:

$$\{(1, I_1), (2, O_1), (3, I_2), (4, O_2), (5, I_3), (6, O_3)\} \tag{3.31}$$

$$\{(1, I_1), (2, O_1), (3, I_2), (4, O_2), (5, O_3), (6, I_3)\} \tag{3.32}$$

$$\{(1, I_1), (2, O_1), (3, O_2), (4, I_2), (5, O_3), (6, I_3)\} \tag{3.33}$$

$$\{(1, O_1), (2, I_1), (3, O_2), (4, I_2), (5, O_3), (6, I_3)\} \tag{3.34}$$

If $m = 4$, which means that a 4-dimensional additional vector

$V_{additional} = (I_1, I_2, I_3, I_4)^T$ can be generated, with the set of artificial data points being $P = \{(1, y_0), (2, y_1), (3, y_2), (4, y_3), (5, y_4), (6, y_5), (7, y_6)\}$:

$$\{(1, I_1), (2, O_1), (3, I_2), (4, O_2), (5, I_3), (6, O_3), (7, I_4)\} \tag{3.35}$$

Altogether, $C_4^1 + C_4^2 + C_4^3 + C_4^4 = 15$ combinations can be produced from the original 3-dimensional feature vector.

3.3 Support Vector Machines

After gaining V_D, Support Vector Machines (SVMs) [122] are herein used to implement the task of image classification. SVMs work by mapping input feature vectors onto the underlying image class labels. Such a classifier seeks to find the optimal separating hyperplane amongst different classes, by focusing on those training points (named support vectors) which are placed at the edge of the underlying feature vectors and whose removal will change the solution to be found.

More formally, SVMs construct a hyperplane in a space of a dimensionality higher than that of the original, which is then used for classification (or for other tasks such as regression and prediction). The underlying intuition is that by mapping the original data space onto a much higher-dimensional space, the class separation between data points will become easier in that space. SVMs use a specific mapping such that the cross products of data points in the larger space are defined in terms of a kernel function [123] which is selected to suit the given problem. In so doing, the cross products may be computed in terms of the variables in the original space, thereby minimising computational effort. In particular, a hyperplane in the higher dimensional space is defined as the set of points whose inner product with any vector in that space is constant. A good hyperplane is learned over a training process such that the resulting hyperplane has the largest distance to the nearest training data points of any given class. This is in order to increase the discriminating power of the trained classifier.

In the following, the Radial Basis Function (RBF) kernel is adopted to implement the SVM-based classifiers, and the sequential minimal optimisation algorithm of [124] is used to train the SVMs. Note that in implementation, as the default settings of SVMs are taken from the LIBSVM [125], no hyper parameters of individual SVMs are further tuned in order to give equal footings in results comparison. Detailed SVM learning mechanism is beyond the scope of this report, but can be found in the literature (e.g., [122, 126]).

3.4 Experimental Results

Although the approach taken in this research is of a generic nature, the present work concentrates on the classification of the MNIST dataset of handwritten digits [32].

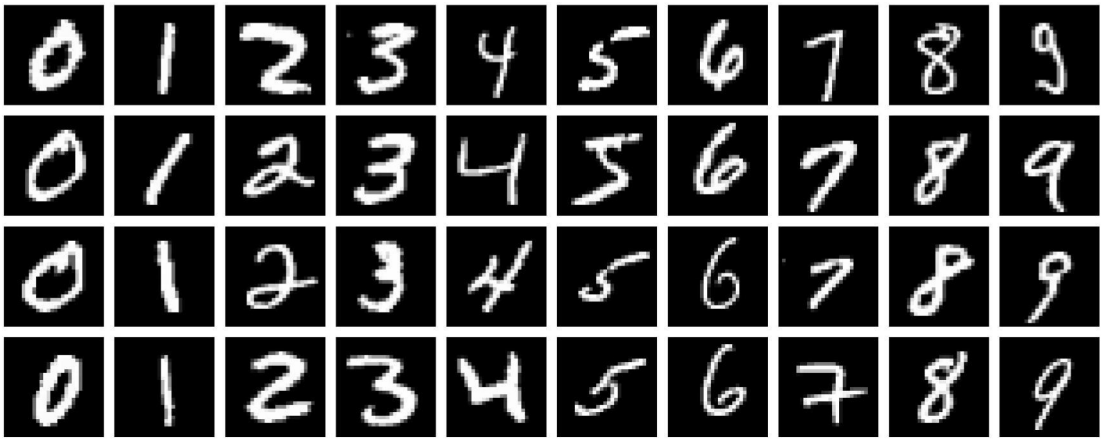


Figure 3.2: Example images in MNIST database

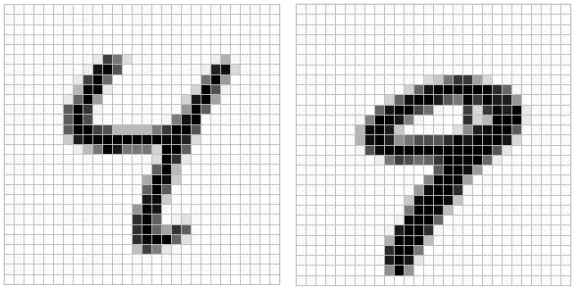


Figure 3.3: Example images for digit 4 and 9

The dataset consists of 60,000 training images and 10,000 test images. Each hand-written digit was originally extracted from a larger set made available by NIST [32], before being size-normalised and centred in a fixed-size image (28×28 pixels). Each image is labeled by one of 10 classes corresponding to the numbers 0-9. Figure 3.2 shows a part of the MNIST database, and Figure 3.3 presents example images for digit 4 and digit 9. The application problem for this research is to develop an image classifier that can detect and recognise different digital numbers from such a given hand-written figure.

The topology of the underlying DeSTIN chosen to perform this experimental investigation consists of 4 layers, with the first layer hosting 8×8 nodes, each receiving a non-overlapping 4×4 patch of a given image that is vectorised into a 16 element input. At each subsequent layer, there are one quarter of the number of the nodes within the preceding layer, such that layer two hosts 4×4 nodes, layer

three 2×2 nodes, and layer four just one node as depicted in Figure 3.1. The dimensionalities of the belief states for layers one, two and three are set to 25, 16 and 12, respectively. In order to give the top layer sufficient representational capacity, the dimensionality of its centroid set is ranged from 3 to 6. Thus, the output of the DeSTIN can be represented as $V_{original} = (O_1, \cdots, O_n)^T, 3 \leq n \leq 6$. Note that as the layer index increases, the information compression rate increases, as reflected by the corresponding reduced dimensionality of the belief space.

3.4.1 Use of DESTINI Features with Linear Interpolation

Table 3.1: Accuracy using additional interpolated features based on a 3-dimensional original vector

Feature list	Accuracy
O_1, O_2, O_3	86.73%
$O_1, O_2, O_3, O_{1+2}, O_{2+3}$	87.96%
$O_1, O_2, O_3, O_{1+2}, sum$	88.61%
$O_1, O_2, O_3, O_{2+3}, sum$	88.80%
$O_1, O_2, O_3, O_{1+2}, O_{2+3}, sum$	88.92%

This experimentation is to investigate the effect of utilising linear interpolation to enrich the representation of features extracted by the given DeSTIN. It first focuses on the use of 3-dimensional DeSTIN outputs in order to minimise the underlying computational complexity required: $V_{original} = (O_1, O_2, O_3)^T$. Additional vectors of 2 dimensions $V_{additional} = (i_1, i_2)^T$ are generated from the original vectors, and there are 3 different kinds of possible combination: $V_{additional} = (i_1 = O_{1+2}, i_2 = O_{2+3})^T$, $V_{additional} = (i_1 = O_{1+2}, i_2 = sum)^T$, and $V_{additional} = (i_1 = O_{2+3}, i_2 = sum)^T$, where $sum = \sum_{i=1}^{3} O_i$. A further interpolated vector of 3 dimensions $V_{additional} = (i_1 = O_{1+2}, i_2 = O_{2+3}, i_3 = sum)^T$ can also be generated. Table 3.1 lists the correct classification rates produced by the resulting SVM classifiers, respectively using different vectors composed by the union of the original features and certain interpolated features. Clearly, the classification accuracy of using the DESTINI features is greater than 86.73%, the accuracy obtained using the original features alone.

Experimentation has also been carried out for cases where more than three original DeSTIN features are used. Table 3.2 lists the correct classification rates based

Table 3.2: Accuracy using linear interpolation based on different dimensional original vectors

Feature list	Combination	Exception	Best	Worst	Average	Original
$O_1, O_2, O_3, i_1, \ldots, i_m (m = 2, 3)$	4	0	88.92%	87.96%	88.57%	86.73%
$O_1, O_2, O_3, O_4, i_1, \ldots, i_m (m = 2, 3, 4)$	11	0	97.56%	97.03%	97.29%	96.52%
$O_1, O_2, O_3, O_4, O_5, i_1, \ldots, i_m (m = 2, 3, 4, 5)$	26	1	98.56%	98.20%	98.41%	98.22%
$O_1, O_2, O_3, O_4, O_5, O_6, i_1, \ldots, i_m (m = 2, 3, 4, 5, 6)$	57	1	98.73%	98.22%	98.50%	98.25%

Table 3.3: Accuracy using Newton interpolation based on different dimensional original vectors

Feature list	Combination	Exception	Best	Worst	Average	Original
$O_1, O_2, O_3, I_1, \ldots, I_m (m = 1, 2, 3, 4)$	15	3	95.62%	85.10%	90.96%	86.73%
$O_1, O_2, O_3, O_4, I_1, \ldots, I_m (m = 1, 2, 3, 4, 5)$	31	1	98.65%	96.43%	98.03%	96.52%
$O_1, O_2, O_3, O_4, O_5, I_1, \ldots, I_m (m = 1, 2, 3, 4, 5, 6)$	63	2	98.98%	98.02%	98.68%	98.22%
$O_1, O_2, O_3, O_4, O_5, O_6, I_1, \ldots, I_m (m = 1, 2, 3, 4, 5, 6, 7)$	127	1	98.92%	98.14%	98.65%	98.25%

Table 3.4: Accuracy using Newton interpolated features based on a 4-dimensional original vector

Feature list	Accuracy	Feature list	Accuracy
I_1, O_1, O_2, O_3, O_4	98.39%	$I_1, O_1, I_2, O_2, I_3, O_3, O_4$	98.10%
O_1, I_1, O_2, O_3, O_4	97.18%	$I_1, O_1, I_2, O_2, O_3, I_3, O_4$	98.28%
O_1, O_2, I_1, O_3, O_4	96.65%	$I_1, O_1, I_2, O_2, O_3, O_4, I_3$	98.58%
O_1, O_2, O_3, I_1, O_4	97.34%	$I_1, O_1, O_2, I_2, O_3, I_3, O_4$	98.06%
O_1, O_2, O_3, O_4, I_1	98.37%	$I_1, O_1, O_2, I_2, O_3, O_4, I_3$	98.58%
$I_1, O_1, I_2, O_2, O_3, O_4$	98.62%	$I_1, O_1, O_2, O_3, I_2, O_4, I_3$	98.60%
$I_1, O_1, O_2, I_2, O_3, O_4$	98.16%	$O_1, I_1, O_2, I_2, O_3, I_3, O_4$	96.43%
$I_1, O_1, O_2, O_3, I_2, O_4$	98.21%	$O_1, I_1, O_2, I_2, O_3, O_4, I_3$	98.01%
$I_1, O_1, O_2, O_3, O_4, I_2$	98.45%	$O_1, I_1, O_2, O_3, I_2, O_4, I_3$	98.59%
$O_1, I_1, O_2, I_2, O_3, O_4$	96.86%	$O_1, O_2, I_1, O_3, I_2, O_4, I_3$	98.18%
$O_1, I_1, O_2, O_3, I_2, O_4$	97.24%	$I_1, O_1, I_2, O_2, I_3, O_3, I_4, O_4$	97.92%
$O_1, I_1, O_2, O_3, O_4, I_2$	98.59%	$I_1, O_1, I_2, O_2, I_3, O_3, O_4, I_4$	98.65%
$O_1, O_2, I_1, O_3, I_2, O_4$	96.93%	$I_1, O_1, I_2, O_2, O_3, I_3, O_4, I_4$	98.65%
$O_1, O_2, I_1, O_3, O_4, I_2$	98.28%	$I_1, O_1, O_2, I_2, O_3, I_3, O_4, I_4$	98.45%
$O_1, O_2, O_3, I_1, O_4, I_2$	98.49%	$O_1, I_1, O_2, I_2, O_3, I_3, O_4, I_4$	97.68%
$I_1, O_1, I_2, O_2, I_3, O_3, I_4, O_4, I_5$	98.43%	O_1, O_2, O_3, O_4	96.52%

on the DeSTIN outputs of a different dimensionality. It presents six performance indicators to show the accuracy. When the output of the underlying DeSTIN is a 4-dimensional vector, the best classification accuracy of using the DESTINI features is 97.56%, while the worst is 97.03%, with the average accuracy being 97.29%, which is significantly higher than that of using four original features (96.52%). Also, the classification accuracy of using the DESTINI features is generally higher than that (98.22%) of using 5-dimensional original DeSTIN features, with only one exception, where the $V_{additional} = (i_1 = O_{3+4}, i_2 = O_{4+5})^T$. Even on this occasion, the accuracy is 98.20%, a mere 0.02% worse off. The classification accuracy of using the DESTINI features is obviously better than that (98.25%) of using the original 6 dimensional features, again with only one exception, where the $V_{additional} = (i_1 = O_{4+5}, i_2 = O_{5+6})^T$ involving a tiny difference in value (0.03%).

Overall, it can be seen from the above results that the classification rates using the DESTINI features are higher than those achievable using the original DeSTIN

features alone. This shows that the classification accuracy is improved with only very minor extra computation overheads, without the need of directly generating a larger number of DeSTIN features which would otherwise incur substantially more computation.

3.4.2 Use of DESTINI Features with Newton Interpolation

The second experimentation reported here deals with the use of applying Newton interpolation to the DeSTIN outputs. The first subset of experiments involves a 4-dimensional original feature vector $V_{original} = (O_1, O_2, O_3, O_4)^T$. Additional vectors of one dimension $V_{additional} = (I_1)^T$ are artificially created using the original feature vector. There are 5 different kinds of such artificial vector. Additional vectors of 2 dimensions $V_{additional} = (I_1, I_2)^T$ are also generated and in this case, there are $C^n_{n+2} = C^4_6$ possible combinations with the original. Of course, many further interpolated vectors can be generated which are shown in Table 3.4. The best classification accuracy of using the DESTINI features is 98.65%, and the average accuracy is 98.03%. There is only one exception, where the use of $V_D = (O_1, I_1, O_2, I_2, O_3, I_3, O_4)^T$ fails to beat the use of just the original DeSTIN features, though the difference between 96.43% and 96.52% is rather small.

The second subset of experiments investigates the correct classification rates produced by the SVM classifiers through the use of different dimensional DeSTIN outputs. The results are listed in Table 3.3. When the output of the given DeSTIN is a 3-dimensional vector, the best classification accuracy using the DESTINI features is 95.62%, while the average reaches 90.96%. There are 3 exceptions (out of 15 combinations), which are $V_D = (O_1, I_1, O_2, O_3)^T$, $V_D = (O_1, O_2, I_1, O_3)^T$, and $V_D = (O_1, I_1, O_2, I_2, O_3)^T$ where the use of the DESTINI features leads to a lower accuracy than that (86.73%) obtainable by the use of the original DeSTIN features (with the corresponding accuracy rates being 86.47%, 85.97% and 85.1%).

The performance of using the DESTINI features can be further improved when the dimensionality of the original DeSTIN feature vectors is slightly increased. For example, with 5-dimensional DeSTIN feature vectors, the accuracy rate is generally higher than that (98.22%) of using just the original, with only two exceptions over 63 cases (where $V_D = (O_1, O_2, I_1, O_3, I_2, O_4, O_5)^T$ and $V_D = (O_1, O_2, I_1, O_3, I_2, O_4, I_3, O_5)^T$) that lead to slightly lower rates (98.09% and 98.02%). However, the average accuracy is 98.68%. Also, the classification accuracy of using the DESTINI features is obviously

Table 3.5: Confusion matrix using interpolation aided feature sets (worse performance)

	0	1	2	3	4	5	6	7	8	9
0	1093	0	6	4	0	3	0	12	4	0
1	0	956	0	0	5	4	0	6	0	3
2	0	4	1001	0	10	0	13	0	3	0
3	4	6	0	915	9	5	11	9	22	4
4	4	0	2	0	865	0	5	5	10	9
5	9	3	0	0	0	972	0	0	0	3
6	0	4	0	0	2	0	1007	8	25	0
7	11	0	2	3	0	17	0	961	4	0
8	0	0	4	14	4	0	32	13	867	0
9	0	10	0	0	1	6	0	0	0	1006

better than that (98.25%) when using 6-dimensional feature vectors, with only one exception (of $V_D = (O_1, O_2, O_3, I_1, O_4, O_5, O_6)^T$). Even on this exceptional occasion, the accuracy is 98.14%, a mere degradation in performance (0.11%). Overall, it is clear that the classification rates attainable by the use of the DESTINI features are significantly higher than those of using the original features alone.

To further investigate the above results, 3 confusion matrix have been listed in Table 3.5,3.6,3.7,respectively. Table 3.7 shows the confusion matrix using the original DeSTIN returned features. Table 3.6 shows the confusion matrix using $V_D = I_1, O_1, O_2, O_3, O_4$, the classification accuracy is 98.39%. 9 out of 10 classes are better classified using V_D.The only exception is digit 1, in this case, 955 instances are correctly classified, while 956 instances are recognized as digit 1 using DeSTIN returned features. Table 3.5 shows the situation that applying interpolation leads to worse performance. As shown in the table, only digit 1, digit 3 and digit 5 are better recognized than that using original feature sets.

3.4.3 Comparison between Linear and Newton Interpolations

Clearly, both interpolation methods work well with DeSTIN. The employment of linear interpolation helps improve the accuracy without causing much computation, while Newton interpolation performs even better with a little extra computation. To support

Table 3.6: Confusion matrix using interpolation aided feature sets (better performance)

	0	1	2	3	4	5	6	7	8	9
0	1115	0	0	0	0	3	4	0	0	0
1	0	955	14	0	0	0	0	0	0	5
2	0	0	1015	0	0	0	16	0	0	0
3	0	0	6	941	0	14	5	0	15	4
4	0	0	2	5	886	0	5	2	0	0
5	0	0	3	0	0	984	0	0	0	0
6	0	0	6	0	0	0	1035	0	5	0
7	6	0	0	0	5	0	3	984	0	0
8	0	0	6	0	0	0	8	5	915	0
9	1	0	10	0	0	3	0	0	0	1009

Table 3.7: Confusion matrix using DeSTIN returned features

	0	1	2	3	4	5	6	7	8	9
0	1089	0	12	4	0	5	0	12	0	0
1	0	956	6	0	5	4	0	0	0	3
2	0	4	1001	0	13	0	13	0	0	0
3	0	6	0	910	9	5	11	13	27	4
4	4	0	7	0	866	0	5	4	10	4
5	9	3	0	0	0	972	0	0	0	3
6	0	4	0	0	0	4	1013	10	15	0
7	11	0	2	0	0	11	7	963	4	0
8	0	0	7	7	4	0	28	14	874	0
9	0	10	0	0	1	4	0	0	0	1008

a more direct comparison between the two versions of DESTINI, Figures 3.4, 3.5, 3.6 and 3.7 summarise the accuracy rates gained using Newton interpolation and those using linear interpolation, for cases where the original feature set utilised is of a dimensionality of 3, 4, 5 and 6, respectively.

The general trends across all cases are rather similar. Although the worst result obtained using Newton interpolation is worse than that of using linear interpolation, it is known from the previous discussions that only for few situations, using either linear or Newton interpolation, does DESTINI show such under-performed behaviour. For a great majority of cases, Newton interpolation-based DESTINI systems outperform linear interpolation-based ones. In particular, consider the case of having a

3-dimensional original feature vector as an example. The best performance achieved by the use of linear interpolation is 88.92% whilst its counterpart using Newton interpolation is 95.62%. Also, the average accuracy by using Newton interpolation is 90.96% which is also higher than 88.57% that is achievable using linear interpolation. Of course, as stated earlier, Newton interpolation does incur slightly more computation. Nevertheless, the cost is still substantially less than that required to generate and use vectors of original features of a dimensionality which is equal to that of the artificially expanded feature sets.

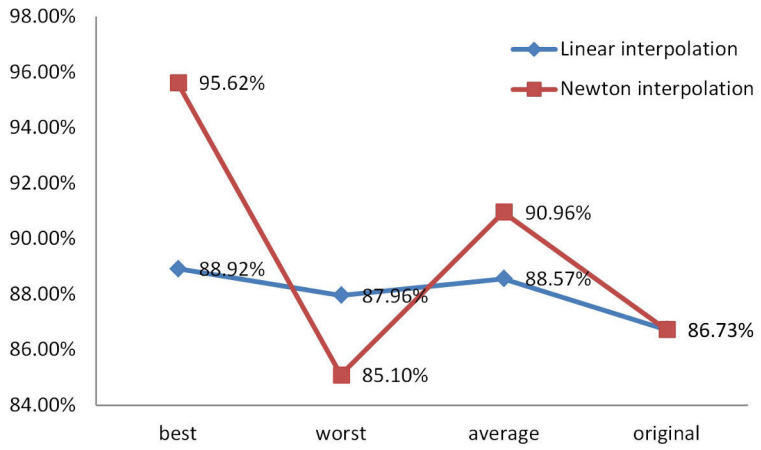


Figure 3.4: Accuracy based on a 3-dimensional original vector

3.5 Summary

This chapter has presented a novel deep learning architecture which draws upon the fundamentals of DeSTIN, supported by feature interpolation. The resulting feature extraction mechanism is well-suited for image classification which is implemented using popular SVMs. This work has been tested using the MNIST dataset. Systematic experimental results demonstrate that the approach developed in this research is capable of efficiently extracting features suitable for input to SVM-based classifiers, generally delivering significantly improved performance in terms of classification accuracy.

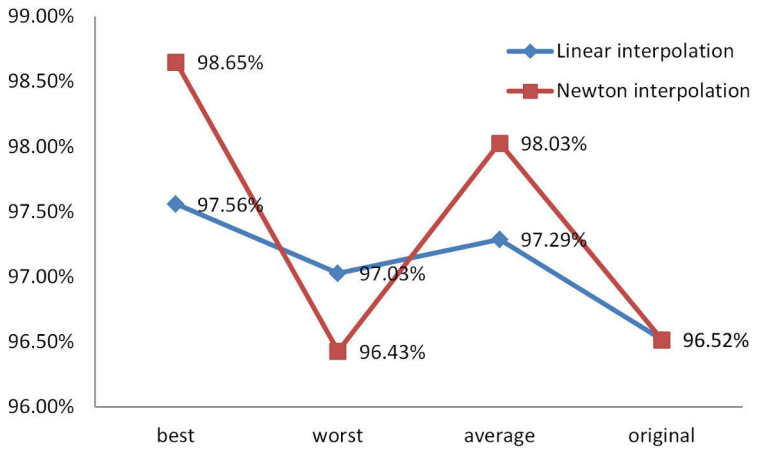


Figure 3.5: Accuracy based on a 4-dimensional original vector

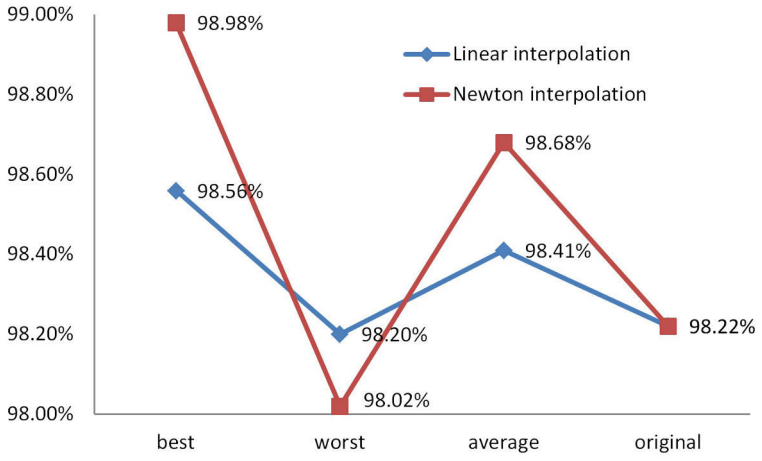


Figure 3.6: Accuracy based on a 5-dimensional original vector

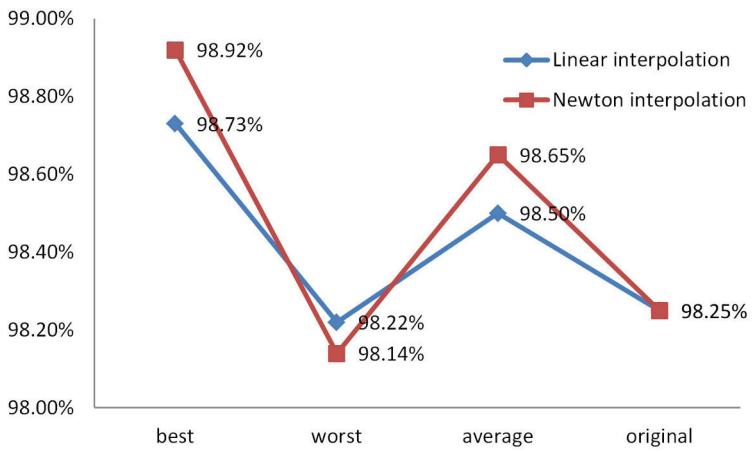


Figure 3.7: Accuracy based on a 6-dimensional original vector

Chapter 4

Clustering Supported Learning Network with Application to Handwritten Digit Recognition

Recent neuroscience findings suggest that the mammalian neocortex does not explicitly preprocess sensory signals, but rather allows them to propagate through a complex hierarchy of processors that learn to represent observations [53, 54]. This provides the biological inspiration for the development of deep machine learning techniques. A number of influential and successful deep learning models have been proposed, including Deep Belief Networks (DBNs) [2], Stacked Autoencoders (SAEs) [3], Convolutional Neural Networks (CNNs) [4], and Deep Spatio-Temporal Inference Network (DeSTIN) [5]. Unfortunately, the process of running this type of network for feature extraction typically involves information processing and passing through a good number of layers. This may introduce considerable overheads on computation and therefore, may offset the potential benefit on the efficiency gained by the entire feature extraction process. An alternative approach is desirable. Driven by this observation, a substantially simplified 2-layer machine learning network is introduced here, exploiting unsupervised learning for pattern representation to support extracting effective features efficiently. Within such a network, each node models patterns it observes by means of clustering. As an initial study, the development reported in this work is focussed on application to handwritten digit recognition, facilitating comparisons with the existing techniques.

The rest of this chapter is organised as follows. Section 4.1 describes the basic concepts and general design of the proposed clustering supported learning network (CSLN). Section 4.2 presents an outline of different clustering techniques that may be used to enable the learning process. Section 4.3 shows the experimental results, supported by comparative studies, demonstrating that the proposed approach is highly promising. The chapter is summarised in Section 4.4.

4.1 Clustering Supported Learning Network (CSLN)

This section presents the structure of CSLN and describes the generic learning mechanisms that may be employed to support the specification of such a network for image feature extraction.

4.1.1 CSLN Structure

Figure 4.1 presents the generic architecture of the proposed CSLN, consisting of a simple hierarchical network of just two layers of nodes. Conceptually, the bottom layer (Layer 1) contains multiple instantiations of an identical node (or cortical circuit), receiving and processing the input data to the network. The input data may be temporally varying, such as image pixel values over a sequence of frames. The top layer (Layer 2) receives as input the outputs of the nodes within the bottom layer. All nodes are each tasked with observing and generalising the data that are presented to it. The resulting outputs of the top layer can be fed to a classifier, such as a support vector machine as extracted features for subsequent processing (e.g., classification).

More specifically, an input to a CSLN at a given time is a digital image, represented as a high-dimensional signal:

$$H_{n\times m} = \begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,k-1} & \cdots & a_{0,m-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,k-1} & \cdots & a_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i-1,0} & a_{i-1,1} & \cdots & a_{i-1,k-1} & \cdots & a_{i-1,m-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,k-1} & \cdots & a_{n-1,m-1} \end{pmatrix} \tag{4.1}$$

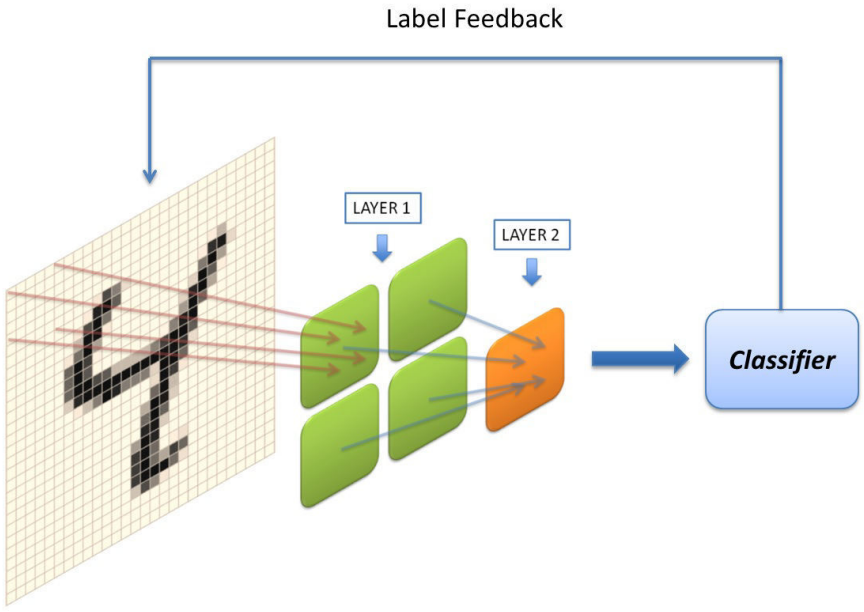


Figure 4.1: Topological architecture of a CSLN

To perform pattern recognition, the number of nodes within the bottom layer of such a CSLN is determined by the complexity of the images it is expected to receive. In general, the more complex the images are, the more nodes it may require, as the nodes are intended to efficiently and effectively process and extract information from a given image patch without losing essential contents. Given $H_{n\times m}$ and the number of nodes (N) in Layer 1, the patch each node processes can be represented by

$$I_{j\times k} = \begin{pmatrix} a_{x,y} & a_{x,y+1} & \cdots & a_{x,y+k-1} \\ a_{x+1,y} & a_{x+1,y+1} & \cdots & a_{x+1,y+k-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{x+j-1,y} & a_{x+j-1,y+1} & \cdots & a_{x+j-1,y+k-1} \end{pmatrix} \tag{4.2}$$

where (x, y) is the displacement shift in $H_{n\times m}$, which is determined by the position of the corresponding specific node. Technically, each node encodes the possible patterns over the patches that it learns to represent. Suppose that the i-th possible pattern is represented by

$$P_i = \begin{pmatrix} p^{(i)}_{0,0} & p^{(i)}_{0,1} & \cdots & p^{(i)}_{0,k-1} \\ p^{(i)}_{1,0} & p^{(i)}_{1,1} & \cdots & p^{(i)}_{1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ p^{(i)}_{j-1,0} & p^{(i)}_{j-1,1} & \cdots & p^{(i)}_{j-1,k-1} \end{pmatrix} \tag{4.3}$$

$i = 1, 2, ..., M$, where M is the number of possible patterns that may be maximally depicted by a node. Each of such patterns is further denoted by a centroid point, thereby signalling that a cluster is created. This leads to a possible pattern space consisting of $\{P_1, P_2, \ldots, P_i, \ldots, P_M\}$. After such a clustering process is complete across all nodes, the internal structure of each individual node in Layer 1 is determined, empowering each node to characterise the observed data (image patch).

4.1.2 Pattern Space Construction and Updating for Network Nodes

The question is how to construct and update the aforementioned pattern spaces for individual nodes within Layer 1. Fortunately, many existing clustering methods can be adopted to perform the required pattern space updating task. Broadly, these can be divided into two categories: off-line clustering and on-line clustering.

A typical off-line clustering algorithm repeatedly sweeps through a set of data samples in an attempt to capture their underlying structure. A stop criterion is checked against every iteration to determine whether the process should continue or not. Algorithm 4.1.1 shows a generalised algorithm for off-line clustering. Given a training dataset $\mathscr{D}$, the algorithm assigns each data point in $\mathscr{D}$ to a certain cluster whose centroid is closest to the point. After each training data has been assigned a cluster, the clusters are updated to determine their new centroids, based on their current contents. The loop continues until a predefined stopping criterion has been reached, producing the learned pattern space.

Different (slightly) from off-line clustering, the category of on-line clustering methods assumes that at every time-step t new data samples $\mathscr{D}(t) = \{X^{(t)}_1, \ldots, X^{(t)}_E\}$ are received that either form a new sequence or are a continuation of a certain previously observed sequence. Such a clustering algorithm is expected to gradually

Algorithm 4.1.1: Generalised off-line clustering algorithm

$\mathscr{D} = \{X_1, \ldots, X_E\}$, the training data;
$\mathscr{P} = \{P_1, \ldots, P_M\}$, the possible pattern space;
M, the number of patterns

1: **while** stop criterion is not satisfied **do**
2: **for all** $X \in \mathscr{D}$ **do**
3: Assign X to closest centroid (pattern)
4: **end for**
5: Update $\mathscr{P} = \{P_1, \ldots, P_M\}$
6: **end while**
7: Output $\mathscr{P} = \{P_1, \ldots, P_M\}$

improve its centroid constructs over time. A generalised on-line clustering algorithm is shown in Algorithm 4.1.2. At each time-step t, a new data subset $\mathscr{D}(t)$ is obtained, then each data $X_i^{(t)}$ in $\mathscr{D}(t)$ is assigned to the closest cluster, which is itself updated at each time step. If there is no more dataset to be learned, the algorithm outputs the last updated pattern space.

Algorithm 4.1.2: Generalised on-line clustering algorithm

$\mathscr{D}(t) = \{X_1^{(t)}, \ldots, X_E^{(t)}\}$, the training data;
$\mathscr{P} = \{P_1^{(t)}, \ldots, P_M^{(t)}\}$, the possible pattern space;
M, the number of patterns;
t, the time step

1: **for all** $t = 1, \ldots, \infty$ **do**
2: Obtain new sequences $\mathscr{D}(t) = \{X_1^{(t)}, \ldots, X_E^{(t)}\}$
3: **for all** $X \in \mathscr{D}(t)$ **do**
4: Assign X to closest centroid
5: Update centroids $\mathscr{P} = \{P_1^{(t)}, \ldots, P_M^{(t)}\}$
6: **end for**
7: Output $\mathscr{P} = \{P_1^{(t)}, \ldots, P_M^{(t)}\}$
8: **end for**

Note that a measurement function $\arg\min s_i, 1 \leq i \leq M$ is used to assign the data $I_{j\times k}$(X_i or $X_i^{(t)}$) to the nearest cluster, the similarity s_i between P_i and $I_{j\times k}$ can be obtained through the use of a distance metric, such as the Euclidean distance or the Cosine distance. Without losing generality, suppose that the Euclidean distance is

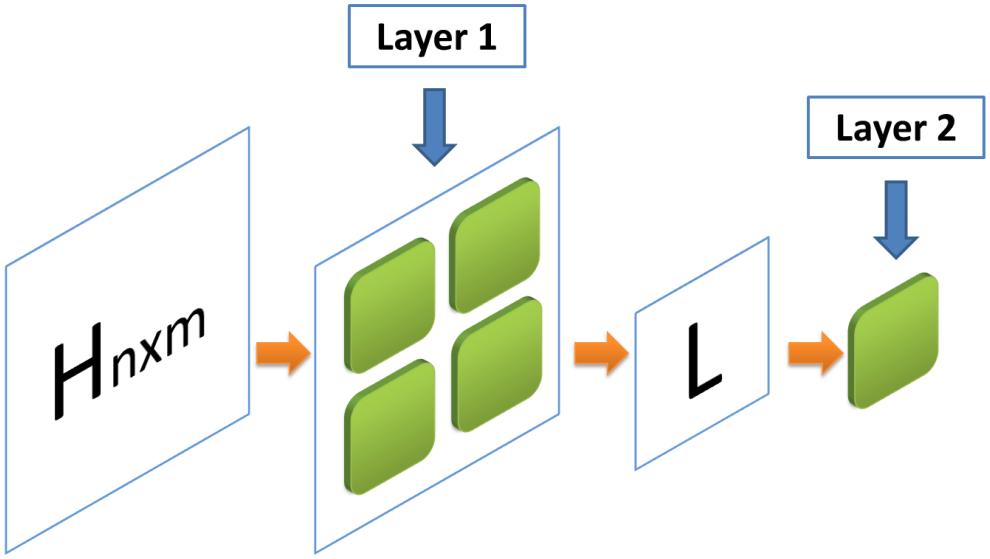


Figure 4.2: Transformation of a CSLN

employed:

$$\begin{aligned} s_i &= d(P_i, I_{j\times k}) \\ &= \sqrt{||p_{0,0} - a_{x,y}||^2 + ||p_{0,1} - a_{x,y+1}||^2 + \cdots + ||p_{j-1,k-1} - a_{x+j-1,y+k-1}||^2} \end{aligned} \tag{4.4}$$

where $1 \leq i \leq M$, M is the number of possible patterns (of the corresponding node), and (x, y) is the displacement shift in the given image $H_{n\times m}$.

As with the nodes in Layer 1, the (unique) node in Layer 2 tries to capture the possible patterns from the inputs it receives. Thus, the clustering algorithm employed in Layer 2 is similar to the algorithm employed by each of the nodes in Layer 1 except that the outputs of Layer 1 are used as the input to Layer 2, in place of the raw image that has been fed to Layer 1 as input.

Figure 4.2 summarises the learning process: Passing an image through a CSLN, the image is processed by two layers of clustering nodes. Layer 1 computes the similarities between image patches to reveal possible patterns and views such similarities as the features extracted, or transformed, from the given image. Computationally, each

patch image is translated into a similarity matrix. For example, the output of the j-th node in Layer 1 can be obtained such that

$$S_j = (s_{j,1}, s_{j,2}, \ldots, s_{j,i}, \ldots, s_{j,M})^T, 1 \leq i \leq M \tag{4.5}$$

where $1 \leq j \leq N$, N is the number of the nodes in Layer 1. After obtaining all S_j, the output of Layer 1 can be represented as

$$L = (S_1, S_2, \ldots, S_j, \ldots, S_N)^T = \begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,M} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ s_{j,1} & s_{j,2} & \cdots & s_{j,M} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N,1} & s_{N,2} & \cdots & s_{N,M} \end{pmatrix} \tag{4.6}$$

This is fed to Layer 2. The only node in Layer 2 comprises a number of centroids, each representing a possible pattern further computed from the composed similarity matrix (as represented in Equation 4.6). As such, any centroid $C_h, 1 \leq h \leq Q$, where Q is the number of the centroids in (the only node of) Layer 2, has the same dimensionality as that of the input L to this layer:

$$C_h = \begin{pmatrix} E_{h,1} \\ E_{h,2} \\ \vdots \\ E_{h,N} \end{pmatrix} = \begin{pmatrix} e_{h,1,1} & e_{h,1,2} & \cdots & e_{h,1,M} \\ e_{h,2,1} & e_{h,2,2} & \cdots & e_{h,2,M} \\ \vdots & \vdots & \ddots & \vdots \\ e_{h,N,1} & e_{h,N,2} & \cdots & e_{h,N,M} \end{pmatrix} \tag{4.7}$$

The similarity O_h between L and C_h can be measured by

$$O_h = d(L, C_h) = \sum_{j=1}^{N} ||S_j - E_{h,j}||^2 = \sum_{j=1}^{N} \sum_{k=1}^{M} ||s_{j,k} - e_{h,j,k}||^2 \tag{4.8}$$

From this, the centroid which L belongs to is determined by $\arg\min O_h$.

The possible pattern space of the node in Layer 2 is updated iteratively. The best possible patterns that characterise the observed data (i.e., the similarities between

patch images and possible patterns in Layer 1) are obtained when the clustering process is terminated. After both Layer 1 and Layer 2 are trained, CSLN is fixed, with each node in this architecture bears the best possible pattern space it has learned from the input data. As such, after propagating through the CSLN, a given original image is converted into a vector of similarity measures, which can be represented by

$$V_{original} = (O_1, \cdots, O_Q)^T \tag{4.9}$$

This is then, fed to a subsequent classifier for pattern recognition. The implementation of the core part of this work is therefore, the use of a clustering algorithm to learn the centroids in each node.

4.2 Clustering Algorithms

The following presents an overview of popular clustering algorithms that may each be employed for updating centroids, or for contributing towards performing the required learning process.

4.2.1 K-Means Clustering

K-means is one of the simplest unsupervised learning algorithms for clustering, which is outlined in Algorithm 4.2.1. The procedure groups data into a presumed number (K) of clusters [127]. The main idea is to define and learn K centroids, one for each cluster. Here, K is preferably larger than necessary to provide each node with sufficient representational capacity in modelling the observed information. The centroids should be placed in a cunning way since different locations lead to different results. The choice is to place them from each other as far away as possible. It works by taking each point belonging to a given data set and associating it to the nearest centroid. When no point is pending, an initial grouping is done. At this stage K new centroids of the clusters need to be re-calculated. From these (K) new centroids, a new binding is carried out between the same data points and their respective nearest new centroids. This process forms a loop, following which the K centroids change their locations iteratively until no more changes are possible as per given data.

Formally, this algorithm works by minimising an objective function, defined as the squared error function below:

$$J = \sum_{j=1}^{K} \sum_{i=1}^{F} ||X_i^{(j)} - C_j||^2 \tag{4.10}$$

Algorithm 4.2.1: K-means algorithm

$\mathscr{D} = \{X_1, \ldots, X_E\}$, the training data;
$\mathscr{C} = \{C_1, \ldots, C_K\}$, the means;
K, the number of means;
t, the number of iterations

1: **loop**
2: **for all** $X \in \mathscr{D}$ **do**
3: $C_i^{(t)} = \{X : ||X - C_i^{(t)}||^2 \leq ||X - C_j^{(t)}||^2 \ \forall j, 1 \leq j \leq K\}$
4: **end for**
5: $C_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{X \in C_i^{(t)}} X$
6: **end loop**

where $||X_i^{(j)} - C_j||$ is a chosen distance measure between a data point $X_i^{(j)}$ and the cluster centre C_j, which is defined as the mean of those data points currently belonging to the cluster, with F denoting the number of data points in the j-th cluster. Although the procedure will always terminate, it does not necessarily find the most optimal cluster configuration (which corresponds to the global objective function minimum) [128]. The algorithm is sensitive to the initial randomly selected cluster centres [129]. However, the K-means algorithm can be run multiple times to reduce this effect.

4.2.2 Fuzzy C-Means

Fuzzy c-means (FCM) is a method of clustering which allows one piece of data to partially belong to two or more clusters [130], as shown in Algorithm 4.2.2. This method is based on minimisation of the following objective function:

$$J_m = \sum_{i=1}^{E} \sum_{j=1}^{C} u_{ij}^m ||X_i - C_j||^2, \tag{4.11}$$

where m is any real number greater than or equal to 1, u_{ij} is the degree of membership of X_i in the cluster j that is of the centroid C_j, and $||X_i - C_j||$ is any norm expressing the dissimilarity between X_i and C_j.

Fuzzy partitioning is carried out through an iterative optimisation process in accordance to the above objective function, updating the membership u_{ij} and the

Algorithm 4.2.2: Fuzzy c-means algorithm

$\mathscr{D} = \{X_1, \ldots, X_E\}$, the training data;
$\mathscr{C} = \{C_1, \ldots, C_K\}$, the centroids;
K, the number of centroids;
t, the number of iterations

1: Initialize $U = \left[u_{ij}\right]$ matrix, $U^{(0)}$
2: **while** $||U^{(t+1)} - U^{(t)}|| > \epsilon$ **do**
3: Calculate $\mathscr{C}^{(t)} = \left[C_j\right]$ with $U^{(t)}$

$$C_j = \frac{\sum_{i=1}^{E} u_{ij}^m X_i}{\sum_{i=1}^{E} u_{ij}^m}$$

4: Update $U^{(t)}$, $U^{(t+1)}$

$$u_{ij} = \frac{1}{\sum_{k=1}^{K} \left(\frac{||X_i - C_j||}{||X_i - C_k||}\right)^{\frac{2}{m-1}}}$$

5: **end while**

cluster centroid C_j such that

$$u_{ij} = \frac{1}{\sum_{k=1}^{K} \left(\frac{||X_i - C_j||}{||X_i - C_k||}\right)^{\frac{2}{m-1}}} \tag{4.12}$$

$$C_j = \frac{\sum_{i=1}^{E} u_{ij}^m X_i}{\sum_{i=1}^{E} u_{ij}^m} \tag{4.13}$$

This iteration will stop when $\max_{ij} \left\{ \left| u_{ij}^{(t+1)} - u_{ij}^{(t)} \right| \right\} < \epsilon$, where ϵ is the termination threshold between 0 and 1, and t is the number of iterations.

4.2.3 Incremental Clustering

The basic idea, as presented in Algorithm 4.2.3, is that K centroids are initialised by the first K data points available. Each centroid is associated with a counter $n_i, i = 1, \ldots, K$, which is used to store the number of data points that belong to the cluster of the centroid is C_i. The Euclidean distances E_i between a new data point X and each centroid are calculated. If C_i is the closest centroid to X and $E_i < \epsilon$, which means the closest distance is less than a preset threshold ϵ, then the counter

Algorithm 4.2.3: Incremental clustering algorithm

$\mathscr{D} = \{X_1, \ldots, X_E\}$, the training data;
$\mathscr{C} = \{C_1, \ldots, C_K\}$, the centroids;
$N = (n_1, \ldots, n_K)$, the counter set

1: Initialize $\mathscr{C} = (C_1, \ldots, C_K)$ to the first K data points
2: Set $n_i = 1, i = 1, \ldots, K$
3: **for all** $X \in \mathscr{D}$ **do**
4: Calculate closest centroid i
5: **if** $\arg\min E_i < \epsilon$ **then**
6: $n_i = n_i + 1$
7: Update C_i, $C_i \leftarrow C_i + (1/n_i)(X - C_i)$
8: **else**
9: $K = K + 1$
10: Add a new centroid $C_{new} = C_K$
11: Add a new counter $n_{new} = 1$ for C_{new}
12: **end if**
13: **end for**

n_i associated with the centroid C_i that is nearest to X is increased by 1, with the others remaining unchanged. Update the centroid C_i by:

$$C_i \leftarrow C_i + (1/n_i)(X - C_i) \tag{4.14}$$

If however, $\arg\min E_i > \epsilon$, which means even the closest distance is larger than the threshold, a new centroid C_{new} is added. This implies that a new pattern P_{new} is added and therefore, a new counter n_{new} is created for the new cluster of the centroid C_{new}. Consequently, the distances of the data point from each centroid are re-calculated. Incremental clustering method is sensitive to the presentation order of data.

4.2.4 Complexity Analysis

The time complexity of each node in a CSLN is determined by the clustering methods it implements. Suppose that there are P data points and the maximum number of centroids $|C|_{max}$, the maximum dimension of an input is $|D|_{max}$, and the maximum number of iterations is $|I|_{max}$. The time complexity of K-means clustering is $O(P \times |D|_{max} \times |C|_{max} \times |I|_{max})$. For fuzzy c-means clustering, the time complexity is $O(P \times |D|_{max} \times |C|^2_{max} \times |I|_{max})$. When incremental clustering is employed,

the time complexity is $O(P \times |D|_{max} \times |I|_{max})$. In the CSLN, which contains 2 layers, the estimates for the time complexity of the three implementations are therefore: $O(M \times P \times |D|_{max} \times |C|_{max} \times |I|_{max})$ or $O(M \times P \times |D|_{max} \times |C|^2_{max} \times |I|_{max})$ or $O(M \times P \times |D|_{max} \times |I|_{max})$, respectively, where M is the total number of the nodes in the CSLN. In terms of space requirement, the complexity of a CSLN is $O(M \times |C|_{max})$.

4.3 Experimentation

In this section, the results of a number of comparative experimental studies carried out are reported to demonstrate the potential of the proposed approach.

4.3.1 Experimental Setup

4.3.1.1 Dataset and Classifier

In this work, CSLN is evaluated on a popular problem, namely, the MNIST data set of handwritten digits, which is widely used for various machine learning and pattern recognition algorithms. The underlying objective of this application is to develop an image pattern classifier that can detect and recognise different digital numbers from a given hand-written figure under noisy environments. For this reason, CSLN is compared against DeSTIN and three other popular deep learning networks when noise is present. In particular, two types of common noise are added to the data: Gaussian noise and Salt-and-Pepper noise. For the former, the mean is 0 and the variance is set to 0.01, 0.06 and 0.1, respectively. For the latter, the noise density is set to 0.01, 0.1 and 0.5, respectively. Figure 4.3 shows a number of example images with Gaussian or Salt-and-Pepper noise added.

As indicated previously, Support Vector Machines (SVMs) [122] are herein used to implement the task of image classification. To have a common basis upon which to perform comparative studies, the popular Radial Basis Function (RBF) kernel is adopted to implement all SVM-based classifiers, and the sequential minimal optimisation algorithm of [131] is used to train the SVMs. The default settings of SVMs are directly taken from LIBSVM [132], no hyper parameters of individual SVMs are further tuned.

Figure 4.3: The sample of noise dataset

4.3.1.2 Deep Learning Networks Compared

In order to compare the performance of CSLN, four popular deep learning networks are chosen to test on the same dataset.

- Deep Spatio-Temporal Inference Network (DeSTIN) [5] is proposed to deal with complex high-dimensional data. Such a network contains a hierarchy of layers whereby each layer consists of multiple instantiations of an identical node. Each node is tasked with observing and learning the sequences of data that are presented to it. The lowest layer of the hierarchy processes the input data (that may be temporally varying) to the network, such as image pixels, and over time continuously constructs a belief state that attempts to characterise the data sequences received. The second layer, and each of all those above it

receives as input the belief states of the nodes at their immediate lower layer, attempting to construct belief states that capture any interesting regularities in the original data. The resulting outputs can be fed to a classifier, such as an SVM as extracted features for subsequent processing (e.g., pattern classification).

- Convolutional Neural Networks (CNNs) [4] are a family of multi-layer neural networks designed for minimal data preprocessing, popular for application in performing image processing tasks. When in action, small portions of an image (dubbed local receptive fields) are treated as inputs to the lowest layer of the hierarchical structure. Information generally propagates through the different layers of the network where the input image is convolved with a set of digital filters whose coefficients are either trained or pre-determined with respect to certain criteria. In particular, the first (or lowest) layer of the network consists of feature maps resulting from the convolution processes, with an additive bias and possibly, a compression or normalisation of the features. This initial stage is followed by a subsampling (typically a small-sized averaging operation) that further reduces the dimensionality. The subsampled feature map then receives a weighting and trainable bias and is propagated through an activation function. The outputs form a new feature map that is then passed through another iteration of convolution, sub-sampling and activation. Such a process can be repeated an arbitrary number of times. The activation outcomes at the termination of this iteration process are forwarded to a conventional feedforward neural network that produces the final output of the system. The intimate relationship between the layers and spatial information in CNNs makes them suitable for image processing and understanding, generally performing well at autonomously extracting salient features from images (but at the cost of significant computation).

- Deep Belief Networks (DBNs) [2] are a type of probabilistic generative model, composed of multiple layers of hidden units (namely latent variables), with connections between the layers but not between units within each layer. When trained on a set of examples in an unsupervised way, a DBN learns to probabilistically reconstruct its inputs. As such, the layers may be seen to act as feature detectors on inputs. After this learning step, a DBN can be further trained in a supervised way to perform classification. In implementation, a DBN can be seen as a composition of a set of simple Restricted Boltsmann Machines (RBMs) [72], where nodes are categorised into two groups (named

visible and hidden nodes respectively) such that a pair of nodes from each of the two groups may have a symmetric link between them whilst no links existing between the nodes within the same group.

- Stacked Autoencoders (SAEs) [3] are neural networks that are formed with multiple layers of sparse autoencoders in which the outputs of each layer are linked to the inputs of the successive layer. An autoencoder (or autoassociator) is a network itself consisting of 3 layers: input, hidden and output. It is devised for the hidden layer to learn a certain representation of a given input such that the input can be reconstructed in the output layer. That is, the target output is the input itself. With stochastic gradient descent the network parameters are learned by minimising the reconstruction error. Interestingly, if the hidden layer is linear and the mean squared error is used as the reconstruction criterion, then the resultant autoencoder can perform principal component transformation. Alternative strategies may be proposed to make autoencoders nonlinear which are appropriate to build deep learning networks in order to extract meaningful representations of data. After such unsupervised training, an SAE can be further trained in a supervised way to perform classification, e.g., through the use of back-propagation.

4.3.1.3 Evaluation Criteria

Owing to the popularity of using DeSTIN to performing the task of handwritten digit recognition, the experimental investigations will first compare the proposed work against the use of DeSTIN. The results are analysed using: both overall recognition accuracy and the precision and recall rates of the recognition. Here, accuracy is defined by the ratio of the number of correctly recognised images to that of the total number of given images; precision is the ratio of the number of correctly recognised images to that of all recognised per digit; and recall is the ratio of the number of correctly recognised images to that of all images that represent the right digits as those correctly recognised. For further comparisons with the other three types of deep learning methods, to avoid redundancy, the performance is assessed using accuracy only.

Note that as a commonly adopted test bed for many learning algorithms, MNIST has already been divided into fixed training and testing subsets in the literature. Thus, in the following analysis of experimental results, conventional means of cross validation and t-test are no longer required.

4.3.2 Experimental Results

4.3.2.1 Comparison with DeSTIN

This first experimental investigation compares the recognition accuracy of using the present approach and that of employing the features extracted by the classical multi-layer deep learning inference networks. A DeSTIN of 4 layers is employed (which is the typical implementation for the same application problem in the literature), each layer uses a different number of centroids, empirically set to 25, 16, 12 and 10, respectively. Note that the topology of the CSLN is of 2 layers with the first layer containing 4 nodes and the top layer containing just one node. The outputs of either of the two types of network are used as the input to an SVM-based classifier.

Table 4.1 presents the recognition accuracy with the use of CSLN-returned features and that of DeSTIN features. No matter which clustering method is employed, the accuracy using the CSLN-returned features is substantially higher than that achievable using the DeSTIN-returned features [133], with much narrower error bound. For each line in Table 4.1, the average accuracy is calculated base on all the results achieved under 6 different noise situations. The error margin is determined by the biggest value between the individual accuracy and the mean (average accuracy). To further examine the relative performance, Table 4.2 and Table 4.3 list the precision and recall results. Again, these results clearly demonstrate the effectiveness of the proposed approach, independent of which clustering method is employed to train the CSLN (noting that the use of incremental clustering consistently outperforms the rest). The error margin is calculated similarly as that achieved in Table 4.1, except that accuracy is replaced by precision or recall.

Table 4.1: Accuracy with CSLN or DeSTIN returned features on datasets with added noise

	Gauss			Salt-Pepper			Average	Error
	0.01	0.06	0.1	0.01	0.1	0.5	accuracy	margin
DeSTIN	0.907	0.888	0.871	0.863	0.828	0.626	0.831	0.205
CSLN (K-means)	0.942	0.923	0.895	0.932	0.903	0.698	0.882	0.184
CSLN (Fuzzy C-means)	0.962	0.950	0.938	0.962	0.933	0.815	0.927	0.112
CSLN (Incremental)	0.978	0.973	0.964	0.979	0.971	0.920	0.964	0.044

Table 4.2: Precision with CSLN or DeSTIN returned features on datasets with added noise

	Gauss			Salt-Pepper			Average	Error
	0.01	0.06	0.1	0.01	0.1	0.5	precision	margin
DeSTIN	0.906	0.888	0.871	0.864	0.827	0.623	0.83	0.207
CSLN (K-means)	0.942	0.923	0.895	0.932	0.903	0.697	0.882	0.185
CSLN (Fuzzy C-means)	0.962	0.950	0.938	0.962	0.933	0.815	0.927	0.112
CSLN (Incremental)	0.978	0.973	0.964	0.979	0.971	0.915	0.963	0.048

Table 4.3: Recall with CSLN or DeSTIN returned features on datasets with added noise

	Gauss			Salt-Pepper			Average	Error
	0.01	0.06	0.1	0.01	0.1	0.5	recall	margin
DeSTIN	0.907	0.888	0.871	0.863	0.828	0.626	0.831	0.205
CSLN (K-means)	0.943	0.923	0.895	0.932	0.903	0.698	0.882	0.184
CSLN (Fuzzy C-means)	0.962	0.950	0.938	0.962	0.933	0.815	0.927	0.112
CSLN (Incremental)	0.978	0.973	0.964	0.979	0.971	0.915	0.963	0.048

The above results are obtained with substantially simplified a network structure as compared to the topology of the DeSTIN. This helps ensure that the learning time is significantly saved. Table 4.4 shows the time consumed to train a CSLN or a DeSTIN, when the experiments are sequentially carried out on the same hardware (Inter(R) Core(TM)2 Duo 3.00 GHz×2 CPU, 4 GB RAM) supported with the same operation system (Windows 7, 64-bit).

Table 4.4: Running time to train CSLN and DeSTIN

DeSTIN	CSLN		
	K-means	Fuzzy C-means	Incremental
169201.1(s)	1063.1(s)	2941.09(s)	496.067(s)

More formally, given a k-layer DeSTIN, which contains $N = 2^0 + 2^2 + 2^4 + \cdots + 2^{2\times(k-1)}$ nodes, the time complexity is $O(N \times P \times D \times C \times I) + O(N \times P \times C \times I) + O(N \times P \times D \times I)$, where P is the number of data points, D is the dimensionality of the input, C is the number of the centroids and I is the number of iterations [5]. For the problem

of digit recognition, the DeSTIN consists of 4 layers. Thus, $N = 2^0 + 2^2 + 2^4 + 2^6 = 85$, and the time complexity is $85 \times (O(P \times D \times C \times I) + O(P \times C \times I) + O(P \times D \times I))$. However, a CSLN only contains 5 nodes, the time complexity is $5 \times O(P \times D \times C \times I)$, $5 \times O(P \times D \times C^2 \times I)$ or $5 \times O(P \times D \times I)$, depending on which clustering method is employed. Clearly CSLN is more efficient in terms of time complexity. When comparing the space requirement, a DeSTIN uses $O(N \times C)$ storage space, while a CSLN only requires $O(M \times C)$ space. Since $M \ll N$, the DeSTIN definitely consumes more storage.

4.3.2.2 Comparison with Other Deep Networks

Further to the performance comparison between CSLN and DeSTIN, this section presents the results of comparing the use of CSLN with that of three other deep learning networks: Deep Belief Networks (DBNs), Convolutional Neural Networks (CNNs) and Stacked Autoencoders (SAEs). In this set of experiments, the incremental clustering method is chosen for the implementation of CSLN. Again, the output of each of these networks is utilised as input features for the SVM-based pattern recogniser.

Table 4.5 lists the results, which demonstrate that overall, CSLN leads to the highest recognition accuracy on the MNIST handwritten digit recognition problem. When the noise level is high, CSLN and CNN perform similarly, significantly outperforming the other two. However, it is worth noting that the computational complexity of CNN is significantly higher than that of CSLN involving convolution amongst layers of filters. SAEs and DBN are underperformed in this case. SAEs is comprised of autoencoders, random small initial weights cause the training to settle into a bad local minima or saddle point. A DBN model is considered as a number of restricted Boltzmann machines (RBMs) stacked together, raining DBN can be simplified to train the multiple layers of RBMs. After training RBMs, the network is fine-tuned by back propagation (BP) algorithm, this leads to the model achieving convergence to local optimal point.

4.4 Summary

This chapter has presented a novel approach to developing a learning network that is of simple topological structure for pattern recognition, through the exploitation

Table 4.5: Accuracy using different deep networks on noise dataset

	Gauss			Salt-Pepper			Average	Error
	0.01	0.06	0.1	0.01	0.1	0.5	accuracy	margin
DBN	0.808	0.635	0.575	0.884	0.727	0.239	0.645	0.406
CNN	0.972	0.963	0.962	0.979	0.971	0.923	0.962	0.039
SAE	0.878	0.846	0.833	0.904	0.856	0.778	0.849	0.071
CSLN	0.978	0.973	0.964	0.979	0.971	0.920	0.964	0.044

of a standard data clustering mechanism. This work has been tested using the popular MNIST dataset, in comparison with four different deep learning techniques. Systematic experimental results demonstrate that the proposed approach is capable of efficiently extracting features suitable for subsequent image pattern recognition tasks, ensuring high accuracy.

Chapter 5

Enriching Data-Driven Fuzzy Rule-Based Classification with Fuzzy Rule Interpolation

Inducing explicit knowledge from data for the development of intelligent systems has received significant attention in the last few decades. In particular, learning classification rules from a given collection of experienced data instances forms a major contribution made by the data mining and machine learning communities. This type of learning, so-called rule induction or concept learning, has led to the successful induction of different descriptions of explicit, machine-usable knowledge, including propositional rules [134] and decision trees [135]. Although the classification rules generated by such techniques are useful in building intelligent classifiers, they often suffer from inadequately expressing and handling the vagueness and ambiguity associated with human thinking and perception. To overcome these shortcomings, a number of approaches have been proposed, ranging from direct modification of non-fuzzy learning methods with fuzzy representation [136, 137, 138, 139, 140] to automatic generation of fuzzy rule-based models using neural networks [141], genetic algorithms [142, 143] or other computational intelligent techniques [14, 94, 95]. The resulting learning mechanisms allow for the creation and maintenance of a concurrent, real time rule base for inference under imprecise conditions and have since found many practical applications [144, 145, 146, 147].

The fundamental assumption underlying aforementioned rule induction approaches is that classification rules learned reflect the quality of the given data. However, the generalisation of the data through induction can only lead to knowledge description of the experienced data. For situations where past data does not cover the full problem domain, learned rules can only work for areas where the given data covers. In other words, systems developed using classification rules induced by such approaches are not always directly applicable to the cases where the learned fuzzy rule base contains gaps due to missing areas of experienced data. A system implemented with such an incomplete rule base is hereafter referred to as a sparse rule based system. Obviously, classical fuzzy reasoning methods can no longer be applied in certain cases due to the fact that a traditional rule-based inference mechanism (e.g., compositional rule of inference [14, 94, 95]) will fail when no fuzzy rule may be found to match the given observation. This becomes a major drawback from the viewpoint of using fuzzy classification systems in solving many real-world problems where typically past data and knowledge learned do contain significant gaps.

Fuzzy rule interpolation (FRI) [27, 28, 29, 31] is of particular significance to support reasoning in the presence of insufficient knowledge. Given a sparse rule base, if an observation has no overlap with the antecedent of any rules, no rule can be invoked in classical fuzzy inference and therefore, no consequence can be derived. Fortunately, the use of FRI techniques enables inference to be performed in such cases. Inspired by this observation, this chapter presents a direct utilisation of FRI in enhancing the reasoning robustness of a fuzzy rule-based classifier which utilises a sparse rule base, namely whose explicit knowledge has been induced from limited experienced data that does not cover the entire problem domain. For completeness, the chapter will first briefly introduce the basic concepts of data-driven fuzzy rule induction, including a simple and effective technique implementing such concepts in Section 5.1. Then, it will describe an integrated application of compositional rule of inference (CRI) and fuzzy rule interpolation in building classification systems that work based on a given sparse rule base, including key computational operations involved, as presented in Section 5.2. This is followed by comparative experimental results of Section 5.3, demonstrating the efficacy of the proposed approach. A successful application to mammographic risk analysis is presented in Section 5.4. Finally, Section 5.5 summaries the work.

5.1 A Simple Approach to Fuzzy Rule Induction

The key task of learning a fuzzy rule-based classifier is to find a finite set of fuzzy production or IF-THEN rules capable of reproducing the input-output behaviour of a given system. Without losing generality, the system to be modelled is herein assumed to be of multiple inputs and a single output (or class variable). That is, the input-output behaviour of interest is depicted by k inputs and one output which may involve q different pattern classes. A fuzzy IF-THEN rule $R_i, i = 1, 2, \ldots, N$, for such a system is represented as follows:

$$\textit{If } x_1 \textit{ is } A_{i1} \textit{ , } x_2 \textit{ is } A_{i2} \textit{ , } \ldots \textit{ , } x_k \textit{ is } A_{ik} \textit{ , then } y \textit{ is } B_i \textit{ with } w_i$$

where N denotes the number of fuzzy rules; $x_1, x_2, \ldots, x_k$ are the k underlying linguistic antecedent variables, jointly defining a k-dimensional pattern space; $A_{ij}, j = 1, 2, \ldots, k$, are the fuzzy values taken by the corresponding antecedents x_j, respectively; $B_i, i = 1, 2, \ldots, N$, are the consequent class labels; and w_i is the rule weight of fuzzy rule R_i, indicating the strength that any input pattern within the fuzzy subspace delimited by the given antecedent values is deemed to belong to the consequent class B_i.

In order to generate an initial set of fuzzy IF-THEN rules, the pattern space is divided into a predefined number of sub-regions. Practically speaking, this is typically done by the domain experts. In this work, a given problem space is assumed to be divided into K^k ($K \geq 2$) fuzzy regions by using Fuzzy C-Means (FCM) clustering [130].

Now consider one of the linguistic antecedent variables, x_i, in order to build the corresponding fuzzy value space, FCM is applied to partition the domain of x_i into K subspaces (K clusters), each of the resulting clusters represents a possible fuzzy value A_{ij}. So the original k-dimensional pattern space is divided into K^k fuzzy regions. For simplicity triangular fuzzy sets are considered in the present work. With the resulting K clusters in each dimension, the triangular membership function of each cluster along a certain dimension of the pattern space can be generated by three points $U(p_{i0}, \alpha), V(p_{i2}, \alpha), W(p_{i1}, 1), i = 1, 2, \ldots, K$, as shown in Figure 5.1, where p_{i0} and p_{i2} represent the minimum and maximum value of a partition i (P_i), p_{i1} represents the value of the centre of a cluster i (C_i) or a partition i (P_i), α is used to

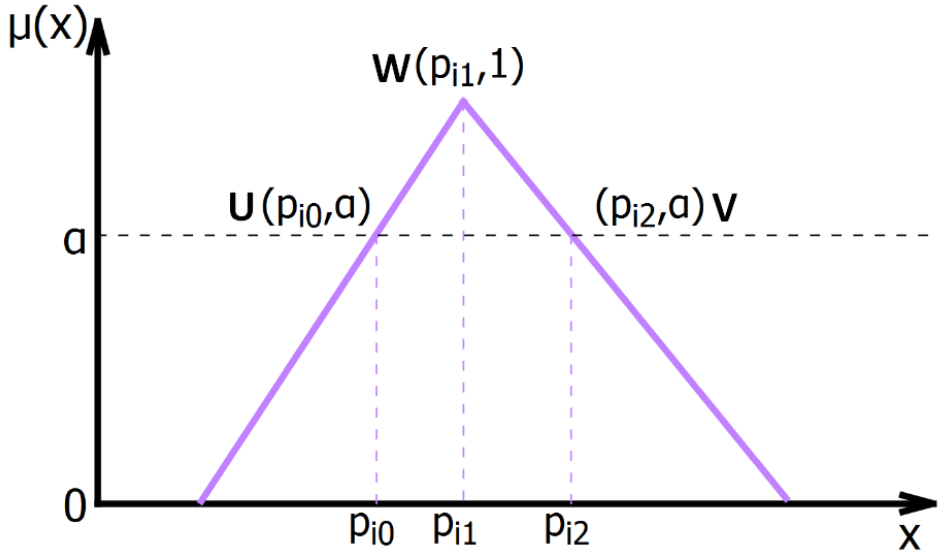


Figure 5.1: Membership function

control the shape of the triangular membership function, the smaller the α is, the narrower the shape of triangular is. Intuitively, the larger the distance from a point x to the centre of the cluster p_{i1}, the less membership value of x should take. To reflect this, given $U(p_{i0}, \alpha)$ and $W(p_{i1}, 1)$, the membership function of the left slope in the triangular is:

$$y = \frac{1-\alpha}{p_{i1}-p_{i0}}x + \frac{p_{i1}\alpha - p_{i0}}{p_{i1}-p_{i0}} \tag{5.1}$$

Similarly, the membership function of the right slope in the triangular is generated based on $W(p_{i1}, 1)$ and $V(p_{i2}, \alpha)$:

$$y = \frac{1-\alpha}{p_{i1}-p_{i2}}x + \frac{p_{i1}\alpha - p_{i2}}{p_{i1}-p_{i2}} \tag{5.2}$$

So given a new observation data x_i, the membership $\mu(x_i)$ can be obtained by:

$$\mu(x_i) = \begin{cases} \dfrac{1-\alpha}{p_{i1}-p_{i0}}x_i + \dfrac{p_{i1}\alpha - p_{i0}}{p_{i1}-p_{i0}}, \ if \ \dfrac{p_{i0}-p_{i1}\alpha}{1-\alpha} \leq x_i \leq p_{i1} \\ \dfrac{1-\alpha}{p_{i1}-p_{i2}}x_i + \dfrac{p_{i1}\alpha - p_{i2}}{p_{i1}-p_{i2}}, \ if \ p_{i1} \leq x_i \leq \dfrac{p_{i2}-p_{i1}\alpha}{1-\alpha} \end{cases} \tag{5.3}$$

If $\mu(x_i) \geq \alpha$ in a fuzzy cluster, then x_i is classified as the term this fuzzy partition represents. In so doing, the initial input pattern is equivalently translated into a fuzzy IF-THEN rule. Therefore, a fuzzy rule base is generated. However, in many

real-world problems, data provided to conduct such learning may not completely cover the entire problem space. As such, the resultant rule base may have gaps scattered in the pattern space, leading to the situation where rule-firing by matching a given observation with the learned clusters becomes void. In this case, as argued previously, fuzzy rule interpolation offers a promising means to perform approximate inference.

Note that FCM is used here as an example method for data-driven fuzzy rule induction owing to its popularity and simplicity. However, many other techniques (e.g., Fuzzy QSBA [148] and Fuzzy Decision Trees [136]) may be employed as alternatives. Which rule induction mechanism is used in implementing the present approach may affect the fuzzy rule-based reasoning accuracy, but has no implication to the approach itself. All such a method provides is an initial fuzzy rule base. It is due to this observation that the simple FCM is adopted here, for illustration purpose. Also, to reduce the overall computation cost for both building the fuzzy rule base and performing inference with the learned rule base, if a given problem involves a large number of domain attributes, it would be helpful to pre-process the data with a certain attribute selection technique before carrying out rule induction [149, 150]. Given the robustness of the present approach, the possible side-effect of using reduced datasets in producing rule bases of significant gaps can be alleviated.

5.2 Integrating Fuzzy Rule Interpolation and Compositional Rule-Based Inference

Depending on the nature of the rule base either fuzzy inference (CRI) or interpolation (FRI) may be employed to draw conclusions from given observations. CRI works by employing the conventional compositional rule of inference (and hence, the abbreviation), relying on a dense rule base in which any observation can find at least a complete or partial matching rule. In many real-world problems, obtaining such a complete rule base is costly or even impractical. Interpolation is more robust when working on sparse rule bases. However, the resulting interpolated conclusions may be not as accurate as their inferred counterparts if partial matching between a given observation and the rule-base can be established. To compensate for the drawbacks of these two techniques, an integrated rule-based classification system structure is proposed, where both inference and interpolation methods can work together to produce the conclusion for an observation, given a sparse rule base.

5.2.1 Architecture of the Integrated System

The overall operation of the proposed integrated system is depicted by the flow-chart of Figure 5.2.

For efficiency, the system starts by matching the α-cut sets of the rule antecedents with a given observation, where α is a predefined confidence level. If certain rules match the observation with at least, the confidence level then the rule whose antecedent overlaps the most is determined. From that, CRI is performed using the highest overlapped rule in general. However, for the specific application to classification problems as the current work is concerned with, no CRI is required, but the class can be directly derived from the rule of the highest overlap between its antecedent and the obervation. This saves a significant amount of computation. Nevertheless, for other reasoning problems such as prediction and regression, CRI will be required.

If no match above the confidence level is found between the observation and any rule in the rule base, then it employs a simple distance metric (e.g., that measuring the distance between the relevant centres of gravity), to identify rules that are the closest to the observation in order to perform interpolation, or extrapolation. In particular, if these closest rules are on the same side of the hyperplane of the observation, then extrapolation is used, otherwise interpolation is used. In this research, to infer the conclusion, the scale and move transformation interpolation (T-FRI) method [30] is utilised due to its popularity and availability (although any other fuzzy interpolation mechanism may act as an alternative).

The following subsections present the key components that jointly implement the proposed fuzzy rule-based reasoning system.

5.2.2 Alpha-Cut Overlapping

Given a confidence level α, an α-cut converts a fuzzy set into a crisp set. Formally, let A be a fuzzy set in the universe of discourse X, $\mu(x)$ is the membership function of A and $\alpha \in [0,1]$. Then α-cut of A is [28]:

$$A_\alpha = \{x \in X |\ \mu_A(x) \geq \alpha\} \tag{5.4}$$

In general, each fuzzy rule in the rule base is assumed to consist of multiple antecedent variables. Given: a certain rule R_i, an α-cut threshold α, the fuzzy set $A = (a_0, a_1, a_2)$

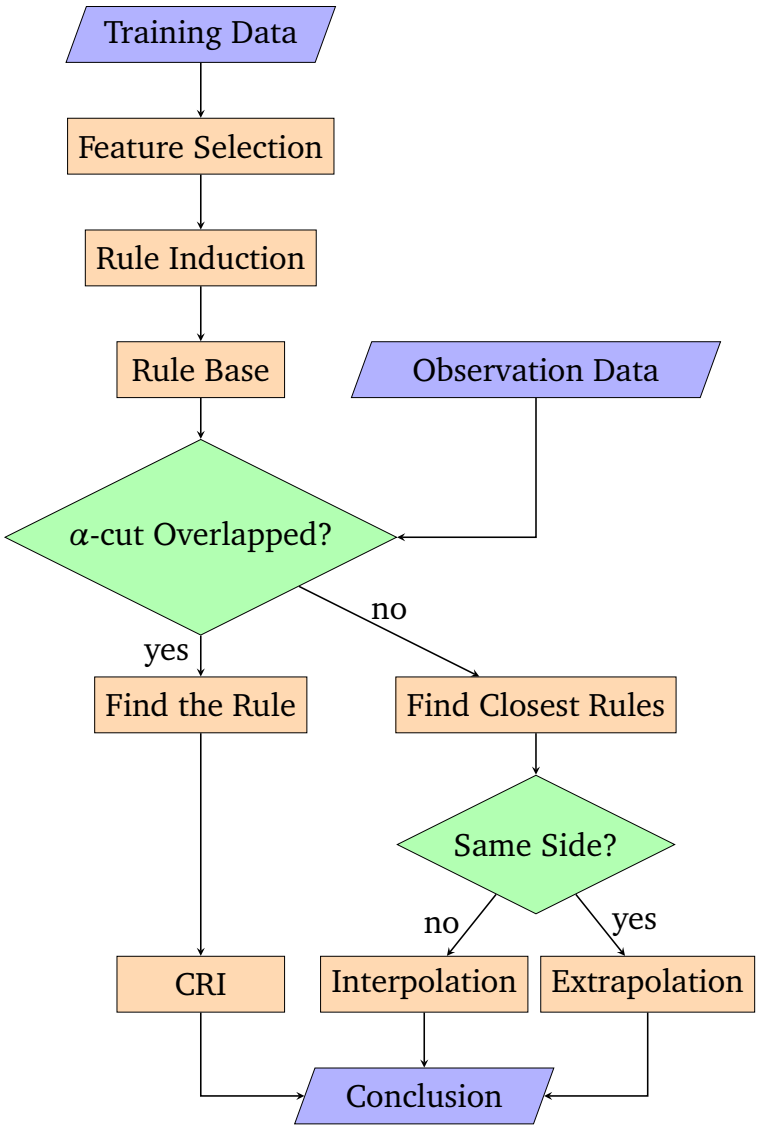


Figure 5.2: Fuzzy rule interpolation-aided reasoning system

of an antecedent variable x_j and the corresponding observation $O = (o_0, o_1, o_2)$, the check for α-cut overlapping is to efficiently decide whether there is a (partial) match between these two values. The procedure for such checking is illustrated in Figure 5.3. For the crisp input O, $o_0 = o_1 = o_2$ is obtained, in this case, O is a singleton.

Two general situations exist in performing checks for α-cut overlapping: one is for the antecedent fuzzy set lying on the left hand side of the observation and the other is for the other way round. Figure 5.3 shows the former situation, where *min* and *max* stand for the *minimum* and *maximum* operator, respectively. First of all,

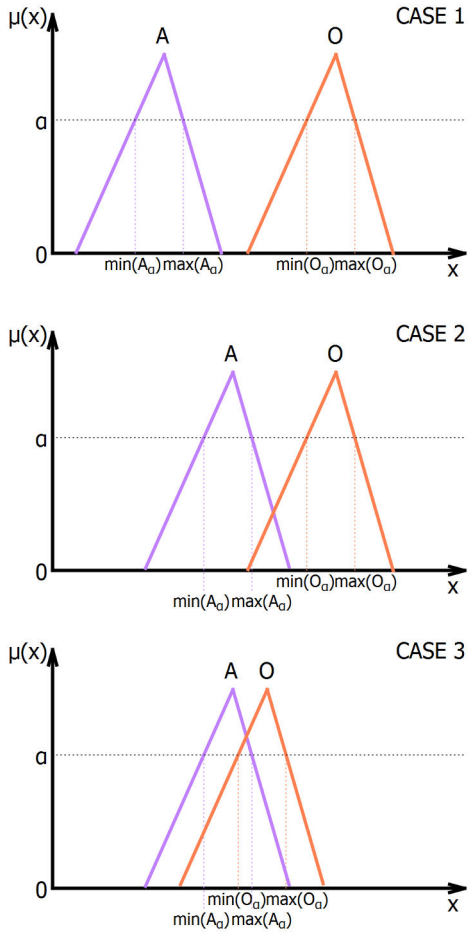


Figure 5.3: α-cut overlapping

an observation is compared with the possible antecedent values for α-cut matching based on the given α level. In this situation, there may exist 3 different cases. In case 1, $o_0 \geq a_2$, which means the left extreme point of O is larger than the right extreme point of A, there is no matching. In case 2, $(o_0 \leq a_2) \wedge (min(O_\alpha) \geq max(A_\alpha))$, therefore there is no matching between A and O above the confidence level α. In case 3, $min(A_\alpha) \leq min(O_\alpha) \leq max(A_\alpha)$, the matching is above the α level. The other situation can be checked similarly, resulting in three other mirrored cases.

Since the detection of possible overlap rules is only necessary to be run above the α level, much calculation for rules which overlap with the observation below the α-level is saved. If all antecedents of only one rule are matched with the corresponding elements of the observation above the α level then the conclusion is directly inferred on the basis of the matched rule. However, if more than one rule are matched with the observation above the threshold then the rule that is of the highest matching

degree is selected. The matching degree is computed by finding the sum of all areas which are delimited by the overlap of two partially matched fuzzy sets of the rule and the observation. The rule with the highest matching value is subsequently used to derive the conclusion via CRI.

In summary, the α-cut overlapping process finds the best matching rule above the α threshold for the given observation from the existing rule base. If such a (partial or exact) matching rule is indeed found then there is no need for computationally more expensive FRI; conventional fuzzy inference, which is well-developed in the literature, can be performed directly to the best matched rule. Although in general multiple matched rules may also be used instead of choosing just one, such multiple rules based CRI unnecessarily increases the overall complexity of the system and therefore, is not adopted here.

Algorithm 5.2.1: α-cut overlapping algorithm

$\mathscr{R} = \{R_1, \ldots, R_N\}$, the fuzzy rule base

```
1: maxArea=0, maxIndex=-1;
2: for all R_i in R do
3:    if min(A_α) ≤ min{O_α} ≤ max(A_α) or min(A_α) ≤ max{O_α} ≤ max(A_α) then
4:       overlap=overlapping area of O and A above α;
5:       if overlap ≥ maxArea then
6:          maxArea=overlap;
7:          maxIndex=i;
8:       end if
9:    end if
10: end for
11: if maxIndex == −1 then
12:    return NULL;
13: else
14:    return R_maxIndex;
15: end if
```

5.2.3 Closest Rule Selection

If none of the rules in the rule base overlaps with the observation above the α level, the closest rules for the given observation are selected for interpolation or extrapolation. A proper distance metric should be defined in order to measure the

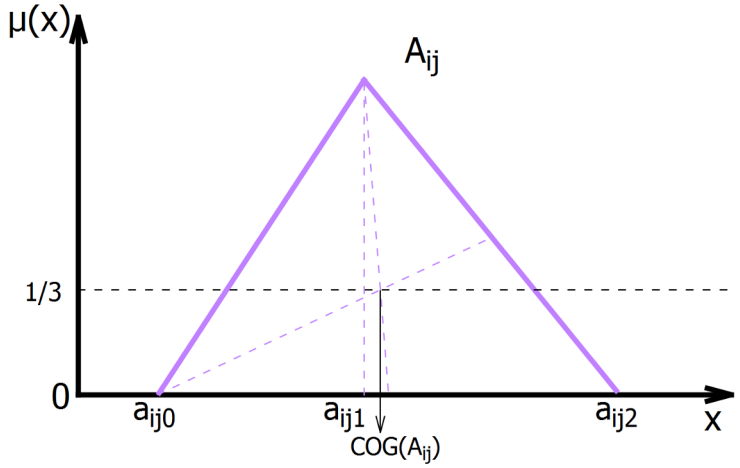


Figure 5.4: Centre of gravity for triangular membership functions

closeness between a rule and the observation. The most commonly used metric for such purposes is the centre of gravity (COG). The concept of COG represents the algebraic average of the masses factored by their distances from a reference point. It is popularly used since it reflects both the location and the shape of a fuzzy set. For a triangular fuzzy set $A_{ij} = (a_{ij0}, a_{ij1}, a_{ij2})$ as shown in Figure 5.4, the COG is calculated as follows and depicted in Figure 5.4 [30]:

$$COG(A_{ij}) = \frac{a_{ij0} + a_{ij1} + a_{ij2}}{3} \tag{5.5}$$

The COG distance between two fuzzy sets O_j (say, the j-th element of the observation) and A_{ij} (the j-th antecedent) is calculated by

$$d(O_j, A_{ij}) = d(COG(O_j), COG(A_{ij})) \tag{5.6}$$

where $d(O_j, A_{ij})$ is any conventional distance metric, depicted in Figure 5.5.

For a k-dimensional observation O and the i-th rule R_i in the rule base, the COG distance between them is defined by extension as follows:

$$COG(O, R_i) = \sum_{j=1}^{k} \frac{d(COG(O_j), COG(A_{ij}))}{range_{x_j}} \tag{5.7}$$

where $COG(O_j)$ and $COG(A_{ij})$ are the COGs of O_j and A_{ij} respectively, and $range_{x_j} = max(x_j) - min(x_j)$ (over the domain of the variable x_j).

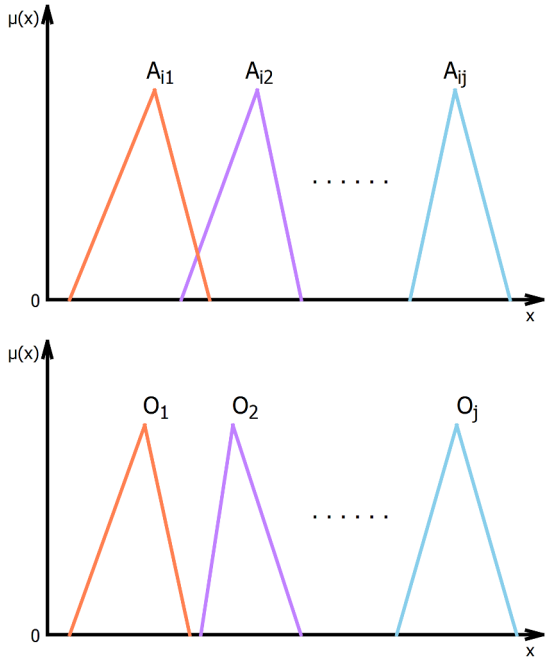


Figure 5.5: Distance between A_i and O

Note that if a certain antecedent of a given rule is missing, the distance between the observation and the rule regarding this variable is treated as 0, reflecting the intuition that any data value is very close to the *null* variable value. This allows for measuring the distance between a given observation and a given rule which may not have fuzzy sets associated with certain attributes.

With the above-defined COG metric, the distances between a given observation and all rules in the rule base can be calculated. Those n rules which have the minimal distances are chosen as the closest rules to the observation. It is worth noting that the n closest rules do not necessarily flank the observation. In the extreme case, all the chosen rules may lie on one side of the hyperplane of the observation, resulting in the need for extrapolation rather than interpolation. However, mathematically, as proven in [31] rule extrapolation has the identical form as rule interpolation. Thus, the following description will focus on rule interpolation unless otherwise stated.

5.2.4 Intermediate Rule Construction

This section proposes how to construct the intermediate rule after $n(n \geq 2)$ closest rules have been chosen. Without losing generality these closest rules are assumed to be:

Algorithm 5.2.2: Find n closest rules based on COG

N, the number of rules;
$\mathscr{R} = \{R_1, \ldots, R_N\}$, the fuzzy rule base;
O, the observation;
$\mathscr{D} = \{D_1, \ldots, D_N\}$, the distance between O and R_i

```
1: for all R_i in R do
2:    D_i = COG(O, R_i);
3: end for
4: for all i = 1 to N do
5:    for all j = 1 to N − 1 do
6:       if D_j ≥ D_{j+1} then
7:          swap( D_j, D_{j+1} )
8:       end if
9:    end for
10: end for
11: for all i = 1 to n do
12:    for all j = 1 to N do
13:       if D_j = D_i then
14:          save the i-th nearest rule R_j
15:       end if
16:    end for
17: end for
```

$$
\begin{aligned}
R_1 &: \mathit{If}\ x_1\ \mathit{is}\ A_{11}\ ,\ \ldots\ ,\ x_k\ \mathit{is}\ A_{1k}\ ,\ \mathit{then}\ y\ \mathit{is}\ B_1 \\
R_2 &: \mathit{If}\ x_1\ \mathit{is}\ A_{21}\ ,\ \ldots\ ,\ x_k\ \mathit{is}\ A_{2k}\ ,\ \mathit{then}\ y\ \mathit{is}\ B_2 \\
&\vdots \\
R_n &: \mathit{If}\ x_1\ \mathit{is}\ A_{n1}\ ,\ \ldots\ ,\ x_k\ \mathit{is}\ A_{nk}\ ,\ \mathit{then}\ y\ \mathit{is}\ B_n
\end{aligned}
$$

Given the observation being represented as:

$$O : \mathit{If}\ x_1\ \mathit{is}\ A_1^*\ ,\ \ldots\ ,\ x_k\ \mathit{is}\ A_k^*\ ,\ \mathit{then}\ y\ \mathit{is}\ B^*$$

the intermediate rule derived from R_1 to R_n can be represented by

$$I : \mathit{If}\ x_1\ \mathit{is}\ A_1^{'}\ ,\ \ldots\ ,\ x_k\ \mathit{is}\ A_k^{'}\ ,\ \mathit{then}\ y\ \mathit{is}\ B^{'}$$

Let $W_{ij}, i = 1, \ldots, n, j = 1, \ldots, k$, denote the weight to which the j-th term of the i-th fuzzy rule contributes to constructing the j-th intermediate fuzzy term $A_j^{'}$. Intuitively,

the larger the distance from A_{ij} to A_j^* , the less value W_{ij} should take. To reflect this, the inversion of the distance is used:

$$W_{ij} = \frac{1}{d(A_{ij}, A_j^*) + 1} \tag{5.8}$$

where $d(A_{ij}, A_j^*)$ is defined in Equation 5.6. Of course, alternative non-increasing functions such as $W_{ij} = exp^{-d(A_{ij}, A_j^*)}$ may also be adopted to assign weights if preferred.

For each variable j, the weights $W_{ij}, i = 1, \ldots, n$ are used to compute the intermediate fuzzy term $A_j^{'}$. Prior to that, they are normalised as follows:

$$W_{ij}^{'} = \frac{W_{ij}}{\sum_{i=1}^{n} W_{ij}} \tag{5.9}$$

so that their sum equals to 1. The intermediate fuzzy term $A_j^{''}, j = 1, 2, \ldots, k$, are computed as:

$$A_j^{''} = \sum_{i=1}^{n} W_{ij}^{'} A_{ij} \tag{5.10}$$

The $A_j^{''}$ calculated via Equation 5.10 does not have the same COG as the observation A_j^*. This is generally true when more than one antecedent are involved (that is why the symbol $A_j^{''}$, rather than $A_j^{'}$, is used here). As such, it fails to satisfy the general requirement of having the same COG value, as imposed by the scale and move transformations. To address this issue, one possible way is to modify $A_j^{''}$ by $zooming$ the outcome of Equation 5.10, so that it becomes a new fuzzy intermediate term $A_j^{'}$ which has the same COG as A_j^*, as follows:

$$A_j^{'} = \gamma_j A_j^{''} \tag{5.11}$$

where γ_j is a constant defined as

$$\gamma_j = \frac{COG(A_j^*)}{COG(A_j^{''})} \tag{5.12}$$

In so doing, the following holds

$$COG(A_j^{'}) = COG(A_j^*) \tag{5.13}$$

Similarly, regarding the consequent, the intermediate fuzzy output $B^{''}$ can be computed by

$$B^{''} = \sum_{i=1}^{n} W_{ai}^{'} B_i \tag{5.14}$$

where $W_{ai}^{'}$ is the mean of $W_{ij}^{'}$:

$$W_{ai}^{'} = \frac{1}{k} \sum_{j=1}^{k} W_{ij}^{'} \tag{5.15}$$

$B^{''}$ is then zoomed to $B^{'}$ as follows:

$$B^{'} = \gamma_a B^{''} \tag{5.16}$$

where γ_a is the mean of the zooming parameters γ_j,

$$\gamma_a = \frac{1}{k} \sum_{j=1}^{k} \gamma_j \tag{5.17}$$

From the above an intermediate rule I can be derived from n rules, R_1 to R_n. It is then, feasible to perform fuzzy reasoning with this new rule without further reference to its originals. Note that interpolation with just two closest rules is the simplest case of the generalised multi-rule interpolation and is the most commonly adopted strategy in implementing FRI in the literature (mainly due to its computational simplicity).

As a certain degree of similarity between $A_j^{'}$ and A_j^{*} has been established, it is intuitive to require that the consequent parts $B^{'}$ and B^{*} attain the same similarity degree. The question is now how to obtain an operator which can represent the similarity degree between fuzzy sets $A_j^{'}$ and A_j^{*}, and to allow transforming $B^{'}$ to B^{*} with the desired degree of similarity. For this purpose, two transformations are proposed as follows.

5.2.5 Scale Transformation

Given a scale rate $s_j (s_j \geq 0), j = 1, 2, \ldots, k$, in order to transform the current support $(a_{j2}^{'} - a_{j0}^{'})$, of fuzzy set $A_j^{'} = (a_{j0}^{'}, a_{j1}^{'}, a_{j2}^{'})$, into a new support $(s_j * (a_{j2}^{'} - a_{j0}^{'}))$ while keeping the same centre of gravity and ratio of left-support $(t_{j1} - t_{j0})$ to right-support

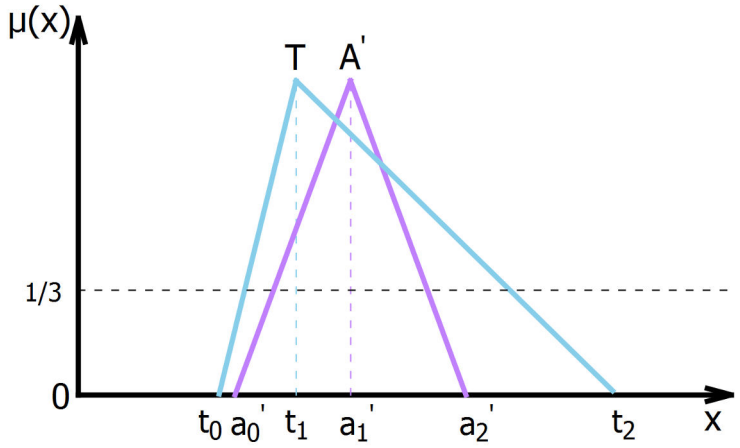


Figure 5.6: Scale transformation

$(t_{j2}-t_{j1})$ of the transformed fuzzy set, $T_j=(t_{j0},t_{j1},t_{j2})$, as those of its original, that is, $COG(A^{'})=COG(T)$ and $\frac{a^{'}_{j1}-a^{'}_{j0}}{a^{'}_{j2}-a^{'}_{j1}}=\frac{t_{j1}-t_{j0}}{t_{j2}-t_{j1}}$, the new t_{j0}, t_{j1}, and t_{j2} must satisfy;

$$t_{j0}=\frac{a^{'}_{j0}(1+2s_j)+a^{'}_{j1}(1-s_j)+a^{'}_{j2}(1-s_j)}{3} \tag{5.18}$$

$$t_{j1}=\frac{a^{'}_{j0}(1-s_j)+a^{'}_{j1}(1+2s_j)+a^{'}_{j2}(1-s_j)}{3} \tag{5.19}$$

$$t_{j2}=\frac{a^{'}_{j0}(1-s_j)+a^{'}_{j1}(1-s_j)+a^{'}_{j2}(1+2s_j)}{3} \tag{5.20}$$

In fact, to satisfy the conditions imposed over the transformation, the linear equations below must hold simultaneously:

$$\begin{cases} \dfrac{t_{j0}+t_{j1}+t_{j2}}{3}=\dfrac{a^{'}_{j0}+a^{'}_{j1}+a^{'}_{j2}}{3} \\ \dfrac{t_{j1}-t_{j0}}{t_{j2}-t_{j1}}=\dfrac{a^{'}_{j1}-a^{'}_{j0}}{a^{'}_{j2}-a^{'}_{j1}} \\ t_{j2}-t_{j0}=s_j(a^{'}_{j2}-a^{'}_{j0}) \end{cases} \tag{5.21}$$

Solving these equations leads to the solutions as given in Equation 5.18, 5.19, 5.20. Note that this scale transformation guarantees that the transformed fuzzy sets are valid as the following holds given $a^{'}_{j0}\leq a^{'}_{j1}\leq a^{'}_{j2}$ and $s_j\geq 0$:

$$t_{j1} - t_{j0} = s_j(a'_{j1} - a'_{j0}) \geq 0$$

$$t_{j2} - t_{j1} = s_j(a'_{j2} - a'_{j1}) \geq 0$$

The above shows how to obtain the resultant fuzzy set T_j when the original fuzzy set A'_j and a scale rate s_j are given. Conversely, in the case where two fuzzy sets $A^*_j = (a^*_{j0}, a^*_{j1}, a^*_{j2})$ and $A'_j = (a'_{j0}, a'_{j1}, a'_{j2})$ which have the same centre of gravity are given, the scale rate is calculated as follows:

$$s_j = \frac{a^*_{j2} - a^*_{j0}}{a'_{j2} - a'_{j0}} \geq 0 \tag{5.22}$$

This measure reflects the similarity degree between A^*_j and A'_j: the closer is s_j to 1, the more similar is A^*_j to A'_j. So $s = \frac{1}{k}\sum_{=1}^{k} s_j$ is therefore used to act as, or to contribute to, the desirable similarity degree in order to transform B' to B^*.

5.2.6 Move Transformation

Similar to the above scale transformation, given a moving distance $l_j, j = 1, 2, \ldots, k$, in order to transform the current fuzzy support $(t_{j2} - t_{j0})$ from the starting location t_{j0} to a new starting position $t_{j0} + l_j$ while keeping the same centre of gravity and length of support of the transformed fuzzy set as its original, i.e., $COG(T) = COG(A^*)$ and $t_{j2} - t_{j0} = a^*_{j2} - a^*_{j0}$, the new a^*_{j0}, a^*_{j1} and a^*_{j2} must be:

$$a^*_{j0} = t_{j0} + l_j, \tag{5.23}$$

$$a^*_{j1} = t_{j1} - 2l_j, \tag{5.24}$$

$$a^*_{j2} = t_{j2} + l_j, \tag{5.25}$$

These can be obtained by solving the equations which are imposed to the transformation:

$$\begin{cases} \dfrac{a^*_{j0} + a^*_{j1} + a^*_{j2}}{3} = \dfrac{t_{j0} + t_{j1} + t_{j2}}{3} \\ a^*_{j0} = t_{j0} + l_j \\ a^*_{j2} - a^*_{j0} = t_{j2} - t_{j0} \end{cases} \tag{5.26}$$

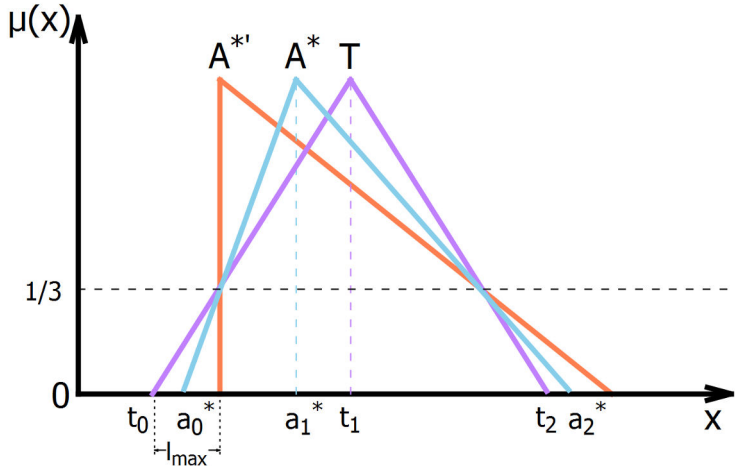


Figure 5.7: Move Transformation

To ensure A^* to be valid, the condition of $0 \leq l_j \leq l_{max} = (t_{j1} - t_{j0})/3$ must hold. If $l_j \geq l_{max}$, the transformation will generate invalid fuzzy sets. For instance, consider the extreme case in which T is transformed to $A^{*'}$, where the left slope of $A^{*'}$ becomes vertical (i.e. $a_0^* = a_1^*$) as shown in Figure 5.7. Here, $l_j = l_{max}$. Any further increase in l_j will lead to the resulting transformed fuzzy set being a non-normal valid fuzzy set. To avoid this, the move ratio m_j is introduced:

$$m_j = \frac{l_j}{(t_{j1} - t_{j0})/3} \tag{5.27}$$

The closer is m_j to 0, the less move (in terms of moving displacement l_j) is being made, and the closer is m_j to 1, the more move is being made. If move ratio $m_j \in [0, 1]$, then $l_j \leq l_{max}$ holds. This ensures that the transformed fuzzy set A_j^* to be normal and valid if T_j is itself a normal valid fuzzy set.

Note that the move transformation has two possible moving directions, the above discusses the right-direction case (from the viewpoint of t_{j0}) with $l_j \geq 0$, the left direction with $l_j \leq 0$ should hold by symmetry:

$$m_j = \frac{l_j}{(t_{j2} - t_{j1})/3} \in [-1, 0] \tag{5.28}$$

As with the description for scale transformation, the above describes how to calculate resultant fuzzy set A_j^* given the original fuzzy set T_j and a moving distance

l_j (or move ratio m_j). Now, consider the case where two valid triangular sets $T_j = (t_{j0}, t_{j1}, t_{j2})$ and $A_j^* = (a_{j0}^*, a_{j1}^*, a_{j2}^*)$ which have the same centre of gravity and have the same support lengths are given, the move ratio m_j can be calculated as follows:

$$m_j = \begin{cases} \dfrac{3(a_{j0}^* - t_{j0})}{t_{j1} - t_{j0}}, \ if \ a_{j0}^* \geq t_{j0} \\ \\ \dfrac{3(a_{j0}^* - t_{j0})}{t_{j2} - t_{j1}}, \ if \ a_{j0}^* \leq t_{j0} \end{cases} \tag{5.29}$$

This reflects the similarity degree between T_j and A_j^*: the closer is m_j to 0, the more similar is T_j to A_j^*. As T_j and A_j^* both are valid, $m_j \in [0,1]$ (when $a_{j0}^* \geq t_{j0}$) or $m_j \in [-1,0]$ (when $a_{j0}^* \leq t_{j0}$) must hold. So $m = \frac{1}{k}\sum_{=1}^{k} m_j$ is therefore used to act as, or to contribute to, the desirable similarity degree in order to transform $B^{'}$ to B^*.

5.3 Comparison with CRI

This section details the experiments conducted and the results obtained in verification of the proposed integrated approach, over 11 benchmark datasets taken from [151]. The system performance is evaluated by comparing the classification accuracy of using the present approach and the accuracy achieved using only CRI over the same rule base. These two methods are first applied on the the original learned rule base. Then, they are tested over a sparse rule base which is generated from the original rule base by randomly eliminating a subset of learned rules (see later for more details).

5.3.1 Experimental Setup

To perform a systematical experimental investigation, both two-fold cross validation (2-FCV) and ten-fold cross validation (10-FCV) are employed. In two-fold cross validation, for each run, a dataset is randomly split into a training set of 50% data and a testing set of the remaining 50%. In ten-fold cross validation, a given data set is partitioned into ten subsets. Of these ten subsets, nine subsets are used to perform training, where the algorithms to be tested are used to build the rule base. A single subset is retained as the test data. This process is then repeated ten times, one per subset or fold being used as the test data. The advantage of ten-fold cross validation over random subsampling is that all objects are used for both training and testing, and each object is used for testing only once per fold. The stratification

of the data prior to its division into different folds ensures that each class label (as far as possible) has equal representation in all folds, thereby helping to alleviate bias/variance problems [152].

In the following experimentation, for each dataset, in learning the rule base, two-fold cross validation is performed 100 times and ten-fold cross validation is performed 10 times in order to lessen the impact of random factors within the algorithms. Such sets of validation results are then averaged to produce the final experimental outcomes. Paired t-test is utilised to assess any statistical significance of the differences between the algorithms' results, with the parameter $p = 0.05$.

To reduce the overall computation cost for building the fuzzy rule bases and also, for performing the inference with the learned rule bases, the feature selection tool, including fuzzy rough feature selection (FRFS) [149], particle swarm optimization based feature selection (PSO-FS) [153], genetic algorithm based feature selection (GA-FS) [154], harmony search based feature selection (HS-FS) [155], is used here to pre-process the data reducing the dataset dimensionality. Then, FCM is run on the dimensionality-reduced training set to induce fuzzy classification rules and membership functions.

Having obtained a learned rule base per dataset, in order to reveal the potential of fuzzy rule interpolation, part of the rule base is randomly removed, resulting in a sparse rule base. The number of rules removed per dataset is listed in the relevant result tables. This rule deletion process is randomly repeated 10 times for each learned rule base to alleviate random factors within the selection process. A reasonable percentage of the originally learned rules are removed this way; particularly, one third for the iris dataset, and one tenth for the remaining datasets.

5.3.2 Systematic Comparison

Table 5.1, 5.2, 5.3, 5.4 list the averaged correct classification rates over the runs on the datasets using two-fold cross validation based on the features selected by FRFS, PSO-FS, GA-FS, HS-FS, respectively, where the better performance over the same rule base on each dataset is highlighted in boldface, and the sign "(v)" indicates that the corresponding result achieved using the proposed method (CRI+FRI) is statistically significantly ($p \leq 0.05$) better than that using CRI only. Importantly, there is no exception that the use of CRI+FRI significantly outperforms that of conventional

approach that works by pattern-matching given rules. It is interesting to note that running CRI+FRI over the sparse rule bases even substantially beats the performance of running CRI with the full learned rule bases across all datasets.

The above observation is completely mirrored in the results obtained from running the $10 \times 10 \times 10$-fold cross validation based on the selected features using different feature selection methods, as shown in Table 5.5, 5.6, 5.7, 5.8, respectively. These systematic comparative experiments clearly demonstrate the potential of the present approach.

Note that in the above results, no attempt has been made to optimise either the fuzzy rule learning mechanism or the fuzzy value definition of the domain variables. This explains why some of the collect recognition rates are rather low. However, no optimisation is intentionally made in order to have a fair ground for the comparative studies reported. The removal of part of the learned rules is randomly carried out without giving any bias to either of the approaches. Should these have been optimised it would be expected that the performance of both CRI and the proposed approach would be improved. However, the significant gap between using direct pattern matching as it is done by CRI and employing FRI is expected to still remain.

5.3.2.1 Closer Examination of the Efficacy of Proposed Approach

To further illustrate the working of the proposed classification system that integrates conventional fuzzy inference with fuzzy rule interpolation over a range of issues, the Iris Dataset is used (for practical simplicity). This dataset contains 150 instances, 50 for each of the three species of iris flowers to be distinguished: setosa, versicolor, and verginica. Each instance is described by four attributes: sepal width (SW), sepal length (SL), petal width (PW), and petal length (PL). The unit for all four of the attributes is centimetres, measured to the nearest millimetre.

Consider one loop of the $10 \times 10 \times 10$-fold cross validation based on FRFS selected features as an example. Table 5.9 shows the index labels of the test set randomly taken from the dataset, involving 5 of the given data for each label class. For this experiment the K for fuzzy c-means clustering is naturally set to 3, and α is set to 0.67. In the same order as that of the attributes appearing in the original dataset, the derived membership functions for the four attributes are shown in Figure 5.8 and the learned fuzzy inference rules are given in Table 5.10. Table 5.11 shows the sparse rule base generated by eliminating one third of the rules in Table 5.10.

Table 5.1: Accuracy (%) with 10×100×2-fold cross validation based on FRFS selected features: The better performance over the same rule base on each dataset is highlighted in boldface, with the sign "(v)" indicating that corresponding result achieved using proposed method is significantly ($p \leq 0.05$) better than that using CRI only.

Dataset	Instances	Features	Classes	Indices of Selected Features	Rules Deleted	Sparse Rule Base		Original Rule Base	
						CRI	CRI+FRI	CRI	CRI+FRI
iris	150	4	3	(1, 2, 3, 4)	6	53.85±11.87	**88.44±7.00(v)**	76.68±6.72	**92.21±4.19(v)**
glass	214	9	6	(1, 2, 3, 4, 5, 6, 7, 8, 9)	7	27.34±5.29	**43.75±5.80(v)**	29.87±5.21	**45.50±5.34(v)**
heart	270	13	2	(1, 3, 4, 5, 8, 10, 12)	8	48.79±6.63	**66.94±5.98(v)**	54.92±6.18	**69.52±4.68(v)**
cleveland	297	13	5	(1, 3, 4, 5, 7, 8, 10, 12)	13	13.28±4.17	**56.24±3.52(v)**	14.55±4.35	**56.43±3.33(v)**
vote	435	16	2	(1, 2, 3, 4, 9, 10, 11, 12, 13, 15, 16)	10	44.96±4.89	**85.34±5.60(v)**	47.50±4.50	**85.89±5.14(v)**
breast-cancer	286	9	2	(1, 2, 3, 5, 6, 7, 8, 9)	12	29.78±6.84	**65.83±3.48(v)**	32.20±6.92	**66.23±3.26(v)**
sonar	208	60	2	(5, 11, 15, 29, 44)	9	30.54±6.01	**66.87±5.09(v)**	34.74±5.85	**67.66±4.94(v)**
vehicle	846	18	4	(1, 2, 5, 6, 10, 12, 15, 16, 18)	30	29.31±2.63	**49.70±2.29(v)**	31.96±2.96	**50.92±2.85(v)**
ionosphere	230	34	2	(3, 4, 5, 16, 20, 21, 32, 33)	7	26.21±5.05	**59.46±5.00(v)**	28.72±4.08	**61.13±4.57(v)**
olitos	120	26	4	(4, 6, 7, 8, 18)	5	21.77±5.81	**53.31±6.48(v)**	24.08±5.88	**54.32±6.58(v)**
wine	178	14	3	(1, 7, 10, 11, 13)	5	43.20±6.73	**86.24±3.80(v)**	48.84±6.30	**87.17±3.50(v)**

Table 5.2: Accuracy (%) with 10×100×2-fold cross validation based on PSO selected features: The better performance over the same rule base on each dataset is highlighted in boldface, with the sign "(v)" indicating that corresponding result achieved using proposed method is significantly ($p \leq 0.05$) better than that using CRI only.

Dataset	Instances	Features	Classes	Indices of Selected Features	Rules Deleted	Sparse Rule Base		Original Rule Base	
						CRI	CRI+FRI	CRI	CRI+FRI
iris	150	4	3	(1, 2, 3, 4)	6	53.85±11.87	**88.44±7.00(v)**	76.68±6.72	**92.21±4.19(v)**
glass	214	9	6	(1, 2, 3, 4, 5, 6, 7, 9)	7	29.33±6.09	**44.88±6.24(v)**	32.17±6.26	**46.96±6.14(v)**
heart	270	13	2	(1, 3, 5, 7, 8, 10, 12, 13)	8	47.95±6.31	**72.35±4.99(v)**	52.89±5.93	**73.72±4.23(v)**
cleveland	297	13	5	(1, 3, 5, 7, 8, 10, 12, 13)	13	10.38±3.36	**52.58±3.49(v)**	11.37±3.37	**52.73±3.29(v)**
vote	435	16	2	(1, 2, 3, 4, 9, 10, 11, 12, 13, 15, 16)	10	44.96±4.89	**85.34±5.60(v)**	47.50±4.50	**85.89±5.14(v)**
breast-cancer	286	9	2	(1, 2, 3, 5, 6, 7, 8, 9)	12	29.78±6.84	**65.83±3.48(v)**	32.20±6.92	**66.23±3.26(v)**
sonar	208	60	2	(9, 15, 23, 41, 42, 49)	9	22.54±4.35	**70.08±4.88(v)**	25.51±4.31	**71.15±4.77(v)**
vehicle	846	18	4	(1, 5, 6, 10, 11, 15, 16, 18)	30	32.13±4.49	**50.45±2.63(v)**	35.33±4.63	**51.98±2.78(v)**
ionosphere	230	34	2	(5, 6, 7, 8, 14, 21, 34)	7	34.37±5.44	**61.34±5.72(v)**	37.95±4.14	**63.55±4.55(v)**
olitos	120	26	4	(5, 12, 13, 17, 20)	5	15.67±5.09	**54.74±6.43(v)**	17.30±5.21	**55.63±6.04(v)**
wine	178	14	3	(4, 5, 7, 12, 13)	5	44.25±6.28	**82.67±4.52(v)**	49.09±5.98	**83.93±4.21(v)**

Table 5.3: Accuracy (%) with 10×100×2-fold cross validation based on GA selected features: The better performance over the same rule base on each dataset is highlighted in boldface, with the sign "(v)" indicating that corresponding result achieved using proposed method is significantly ($p \leq 0.05$) better than that using CRI only.

Dataset	Instances	Features	Classes	Indices of Selected Features	Rules Deleted	Sparse Rule Base		Original Rule Base	
						CRI	CRI+FRI	CRI	CRI+FRI
iris	150	4	3	(1, 2, 3, 4)	6	53.85±11.87	**88.44±7.00(v)**	76.68±6.72	**92.21±4.19(v)**
glass	214	9	6	(1, 2, 3, 4, 5, 6, 7, 9)	7	29.33±6.09	**44.88±6.24(v)**	32.17±6.26	**46.96±6.14(v)**
heart	270	13	2	(1, 2, 3, 4, 5, 6, 7, 8)	8	44.41±6.33	**62.67±4.13(v)**	48.37±6.15	**64.15±3.11(v)**
cleveland	297	13	5	(1, 3, 4, 5, 7, 8, 10, 12)	13	13.28±4.17	**56.24±3.52(v)**	14.55±4.35	**56.43±3.33(v)**
vote	435	16	2	(1, 2, 3, 4, 9, 10, 11, 12, 13, 15, 16)	10	44.96±4.89	**85.34±5.60(v)**	47.50±4.50	**85.89±5.14(v)**
breast-cancer	286	9	2	(1, 2, 3, 5, 6, 7, 8, 9)	12	29.78±6.84	**65.83±3.48(v)**	32.20±6.92	**66.23±3.26(v)**
sonar	208	60	2	(13, 18, 26, 37, 41, 57, 59)	9	12.46±3.72	**62.87±4.70(v)**	13.86±3.96	**63.56±4.56(v)**
vehicle	846	18	4	(1, 3, 4, 5, 9, 10, 14, 15, 16)	30	28.36±3.69	**47.71±3.67(v)**	30.67±3.36	**49.06±3.43(v)**
ionosphere	230	34	2	(3, 4, 7, 18, 26, 31, 34)	7	29.46±5.59	**57.26±5.23(v)**	32.82±4.03	**59.32±4.49(v)**
olitos	120	26	4	(5, 8, 13, 14, 18, 23)	6	8.90±3.71	**42.85±6.34(v)**	9.98±3.91	**43.52±5.96(v)**
wine	178	14	3	(1, 2, 4, 9, 10)	5	36.58±6.29	**77.07±4.64(v)**	39.76±5.90	**77.99±4.45(v)**

Table 5.4: Accuracy (%) with 10×100×2-fold cross validation based on HS selected features: The better performance over the same rule base on each dataset is highlighted in boldface, with the sign "(v)" indicating that corresponding result achieved using proposed method is significantly ($p \leq 0.05$) better than that using CRI only.

Dataset	Instances	Features	Classes	Indices of Selected Features	Rules Deleted	Sparse Rule Base		Original Rule Base	
						CRI	CRI+FRI	CRI	CRI+FRI
iris	150	4	3	(1, 2, 3, 4)	6	53.85±11.87	**88.44±7.00(v)**	76.68±6.72	**92.21±4.19(v)**
glass	214	9	6	(1, 2, 4, 5, 6, 7, 8, 9)	7	29.27±5.58	**43.99±6.08(v)**	32.01±5.62	**45.77±5.92(v)**
heart	270	13	2	(1, 3, 4, 5, 8, 10, 12)	8	48.79±6.63	**66.94±5.98(v)**	54.92±6.18	**69.52±4.68(v)**
cleveland	297	13	5	(1, 3, 4, 5, 7, 8, 10, 12)	13	13.28±4.17	**56.24±3.52(v)**	14.55±4.35	**56.43±3.33(v)**
vote	435	16	2	(1, 2, 3, 4, 9, 10, 11, 12, 16)	10	53.47±5.60	**84.85±4.89(v)**	56.78±4.93	**85.51±4.32(v)**
breast-cancer	286	9	2	(1, 2, 3, 5, 6, 7, 8, 9)	12	29.78±6.84	**65.83±3.48(v)**	32.20±6.92	**66.23±3.26(v)**
sonar	208	60	2	(10, 12, 14, 36, 48, 56)	9	24.06±4.22	**69.25±4.33(v)**	26.94±3.81	**69.95±4.25(v)**
vehicle	846	18	4	(1, 4, 5, 7, 8, 14, 15, 16, 18)	30	30.19±4.08	**48.89±2.73(v)**	33.15±3.59	**50.83±2.01(v)**
ionosphere	230	34	2	(3, 5, 6, 9, 16, 25, 30, 34)	7	28.07±5.00	**59.92±4.39(v)**	30.65±3.64	**60.89±4.03(v)**
olitos	120	26	4	(6, 12, 14, 18, 22)	5	15.53±5.00	**54.19±5.98(v)**	17.22±5.02	**55.3±5.88(v)**
wine	178	14	3	(1, 4, 6, 7, 11)	5	42.21±5.67	**83.31±4.81(v)**	46.85±5.14	**84.55±4.56(v)**

Table 5.5: Accuracy (%) with 10×10×10-fold cross validation based on FRFS selected features: The better performance over the same rule base on each dataset is highlighted in boldface, with the sign "(v)" indicating that corresponding result achieved using proposed method is significantly ($p \leq 0.05$) better than that using CRI only.

Dataset	Instances	Features	Classes	Indices of Selected Features	Rules Deleted	Sparse Rule Base		Original Rule Base	
						CRI	CRI+FRI	CRI	CRI+FRI
iris	150	4	3	(1, 2, 3, 4)	8	55.78±15.94	**85.63±10.32(v)**	82.47±10.10	**91.00±7.14(v)**
glass	214	9	6	(1, 2, 3, 4, 5, 6, 7, 8, 9)	10	35.27±10.87	**47.63±11.10(v)**	38.33±10.66	**49.81±10.25(v)**
heart	270	13	2	(1, 3, 4, 5, 8, 10, 12)	20	23.41±9.07	**71.26±8.37(v)**	25.56±9.20	**71.15±8.65(v)**
cleveland	297	13	5	(1, 3, 4, 5, 7, 8, 10, 12)	23	15.91±6.98	**52.08±8.95(v)**	17.41±7.36	**52.24±9.09(v)**
vote	435	16	2	(1, 2, 3, 4, 9, 10, 11, 12, 13, 15, 16)	20	42.61±7.82	**86.83±6.66(v)**	45.23±7.36	**87.68±6.39(v)**
breast-cancer	286	9	2	(1, 2, 3, 5, 6, 7, 8, 9)	15	40.60±9.97	**66.69±9.02(v)**	43.29±9.76	**67.57±8.75(v)**
sonar	208	60	2	(5, 11, 15, 29, 44)	10	42.57±11.07	**66.18±10.04(v)**	47.10±10.76	**67.80±8.87(v)**
vehicle	846	18	4	(1, 2, 5, 6, 10, 12, 15, 16, 18)	50	36.08±5.38	**50.74±4.86(v)**	39.03±4.73	**52.06±4.18(v)**
ionosphere	230	34	2	(3, 4, 5, 16, 20, 21, 32, 33)	13	31.88±10.30	**61.82±10.41(v)**	35.22±9.68	**63.70±9.54(v)**
olitos	120	26	4	(4, 6, 7, 8, 18)	8	35.34±13.90	**57.74±14.30(v)**	39.00±14.48	**59.42±13.88(v)**
wine	178	14	3	(1, 7, 10, 11, 13)	7	56.58±11.90	**88.86±6.90(v)**	63.65±10.57	**89.94±6.67(v)**

Table 5.6: Accuracy (%) with 10×10×10-fold cross validation based on PSO selected features: The better performance over the same rule base on each dataset is highlighted in boldface, with the sign "(v)" indicating that corresponding result achieved using proposed method is significantly ($p \leq 0.05$) better than that using CRI only.

Dataset	Instances	Features	Classes	Indices of Selected Features	Rules Deleted	Sparse Rule Base		Original Rule Base	
						CRI	CRI+FRI	CRI	CRI+FRI
iris	150	4	3	(1, 2, 3, 4)	8	55.78±15.94	**85.63±10.32(v)**	82.47±10.10	**91.00±7.14(v)**
glass	214	9	6	(1, 2, 3, 4, 5, 6, 7, 9)	10	37.82±10.41	**49.01±11.84(v)**	40.52±9.56	**50.95±11.52(v)**
heart	270	13	2	(1, 3, 5, 7, 8, 10, 12, 13)	20	22.83±9.38	**74.66±7.86(v)**	25.00±9.69	**74.74±7.92(v)**
cleveland	297	13	5	(1, 3, 5, 7, 8, 10, 12, 13)	23	19.85±7.81	**54.60±9.35(v)**	21.76±8.17	**54.86±9.75(v)**
vote	435	16	2	(1, 2, 3, 4, 9, 10, 11, 12, 13, 15, 16)	20	42.61±7.82	**86.83±6.66(v)**	45.23±7.36	**87.68±6.39(v)**
breast-cancer	286	9	2	(1, 2, 3, 5, 6, 7, 8, 9)	15	40.60±9.97	**66.69±9.02(v)**	43.29±9.76	**67.57±8.75(v)**
sonar	208	60	2	(9, 15, 23, 41, 42, 49)	10	35.16±10.14	**71.37±10.18(v)**	37.85±10.38	**70.21±9.63(v)**
vehicle	846	18	4	(1, 5, 6, 10, 11, 15, 16, 18)	50	39.66±4.69	**52.28±4.10(v)**	43.63±5.29	**54.06±4.03(v)**
ionosphere	230	34	2	(5, 6, 7, 8, 14, 21, 34)	11	42.03±11.44	**63.67±11.15(v)**	47.57±11.17	**67.13±10.68(v)**
olitos	120	26	4	(5, 12, 13, 17, 20)	8	26.42±12.70	**59.45±15.10(v)**	29.08±12.89	**60.75±14.26(v)**
wine	178	14	3	(4, 5, 7, 12, 13)	8	55.67±12.60	**85.42±8.40(v)**	62.06±12.07	**86.76±7.55(v)**

Table 5.7: Accuracy (%) with 10×10×10-fold cross validation based on GA selected features: The better performance over the same rule base on each dataset is highlighted in boldface, with the sign "(v)" indicating that corresponding result achieved using proposed method is significantly ($p \leq 0.05$) better than that using CRI only.

Dataset	Instances	Features	Classes	Indices of Selected Features	Rules Deleted	Sparse Rule Base		Original Rule Base	
						CRI	CRI+FRI	CRI	CRI+FRI
iris	150	4	3	(1, 2, 3, 4)	8	55.78±15.94	**85.63±10.32(v)**	82.47±10.1	**91.00±7.14(v)**
glass	214	9	6	(1, 2, 3, 4, 5, 6, 7, 9)	10	37.82±10.41	**49.01±11.84(v)**	40.52±9.56	**50.95±11.52(v)**
heart	270	13	2	(1, 2, 3, 4, 5, 6, 7, 8)	20	24.00±8.98	**67.51±8.26(v)**	26.07±9.52	**67.78±8.13(v)**
cleveland	297	13	5	(1, 3, 4, 5, 7, 8, 10, 12)	23	15.91±6.98	**52.08±8.95(v)**	17.41±7.36	**52.24±9.09(v)**
vote	435	16	2	(1, 2, 3, 4, 9, 10, 11, 12, 13, 15, 16)	20	42.61±7.82	**86.83±6.66(v)**	45.23±7.36	**87.68±6.39(v)**
breast-cancer	286	9	2	(1, 2, 3, 5, 6, 7, 8, 9)	15	40.60±9.97	**66.69±9.02(v)**	43.29±9.76	**67.57±8.75(v)**
sonar	208	60	2	(13, 18, 26, 37, 41, 57, 59)	10	20.24±9.73	**63.92±10.40(v)**	21.75±9.81	**64.65±10.48(v)**
vehicle	846	18	4	(1, 3, 4, 5, 9, 10, 14, 15, 16)	50	35.62±6.52	**50.10±6.12(v)**	38.53±6.24	**51.23±6.12(v)**
ionosphere	230	34	2	(3, 4, 7, 18, 26, 31, 34)	13	34.65±11.07	**59.90±10.95(v)**	39.17±10.52	**62.91±10.47(v)**
olitos	120	26	4	(5, 8, 13, 14, 18, 23)	9	18.37±10.15	**45.78±12.89(v)**	20.33±10.70	**46.50±12.92(v)**
wine	178	14	3	(1, 2, 4, 9, 10)	8	47.41±12.54	**77.92±10.55(v)**	52.47±11.97	**79.41±10.44(v)**

Table 5.8: Accuracy (%) with 10×10×10-fold cross validation based on HS selected features: The better performance over the same rule base on each dataset is highlighted in boldface, with the sign "(v)" indicating that corresponding result achieved using proposed method is significantly ($p \leq 0.05$) better than that using CRI only.

Dataset	Instances	Features	Classes	Indices of Selected Features	Rules Deleted	Sparse Rule Base		Original Rule Base	
						CRI	CRI+FRI	CRI	CRI+FRI
iris	150	4	3	(1, 2, 3, 4)	8	55.78±15.94	**85.63±10.32(v)**	82.47±10.10	**91.00±7.14(v)**
glass	214	9	6	(1, 2, 4, 5, 6, 7, 8, 9)	10	38.24±9.9	**48.58±11.18(v)**	40.90±9.79	**50.19±11.34(v)**
heart	270	13	2	(1, 3, 4, 5, 8, 10, 12)	20	23.41±9.07	**71.26±8.37(v)**	25.56±9.20	**71.15±8.65(v)**
cleveland	297	13	5	(1, 3, 4, 5, 7, 8, 10, 12)	23	15.91±6.98	**52.08±8.95(v)**	17.41±7.36	**52.24±9.09(v)**
vote	435	16	2	(1, 2, 3, 4, 9, 10, 11, 12, 16)	20	57.50±8.07	**87.45±6.87(v)**	62.59±7.42	**88.55±6.49(v)**
breast-cancer	286	9	2	(1, 2, 3, 5, 6, 7, 8, 9)	15	40.60±9.97	**66.69±9.02(v)**	43.29±9.76	**67.57±8.75(v)**
sonar	208	60	2	(10, 12, 14, 36, 48, 56)	10	35.05±8.92	**69.81±10.14(v)**	38.05±8.72	**70.55±9.67(v)**
vehicle	846	18	4	(1, 4, 5, 7, 8, 14, 15, 16, 18)	50	36.86±5.92	**51.51±5.08(v)**	38.60±6.68	**51.66±5.24(v)**
ionosphere	230	34	2	(3, 5, 6, 9, 16, 25, 30, 34)	13	33.63±10.96	**61.74±11.08(v)**	37.30±10.55	**63.78±10.87(v)**
olitos	120	26	4	(6, 12, 14, 18, 22)	8	29.73±14.06	**60.28±14.21(v)**	32.50±14.56	**61.08±13.95(v)**
wine	178	14	3	(1, 4, 6, 7, 11)	8	55.67±12.60	**85.42±8.40(v)**	62.06±12.07	**86.76±7.55(v)**

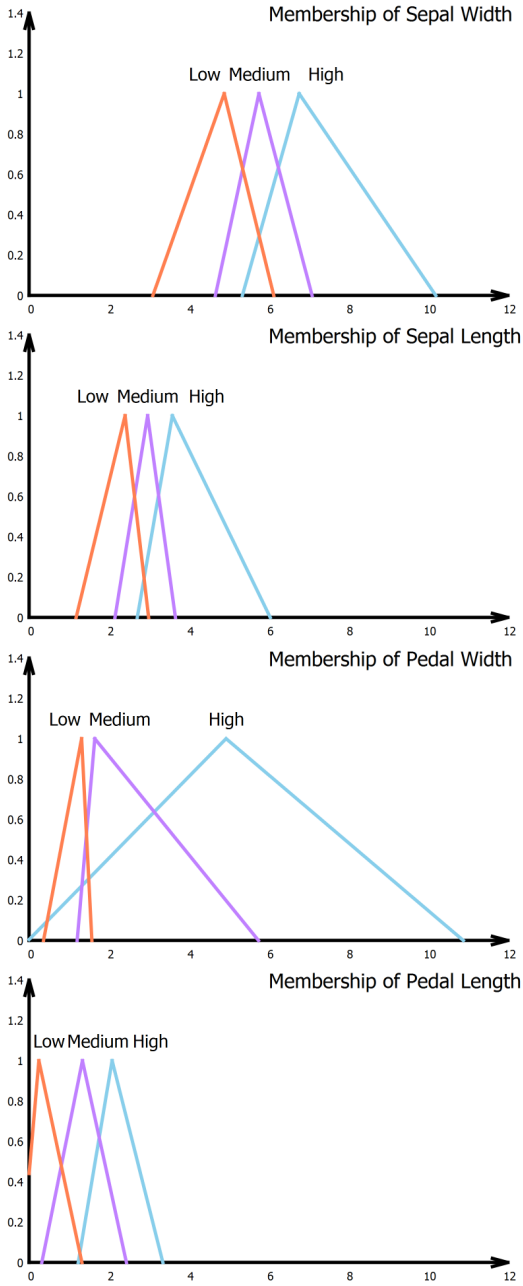


Figure 5.8: Membership function of each attribute

5.3.2.2 Use of CRI When Rules Matching the Observation

This first case shows the situation where a given observation matches with at least one rule within the sparse rule base, above the α threshold. In this case, there is no need for computing rule interpolation, but the compositional rule of inference (CRI) can be applied directly. The results are listed in Table 5.12 and Table 5.13 where the

Table 5.9: Labels for test data

Label list	42, 18, 35, 1, 20, 84, 70, 93, 95, 67, 127, 128, 144, 109, 115

Table 5.10: Original fuzzy rule base

Rule NO.	Sepal Width	Sepal Length	Petal Width	Petal Length	Class
Rule 1	Low	High	Medium	Low	Setosa
Rule 2	Low	High	Low	Low	Setosa
Rule 3	Low	Medium	Low	Low	Setosa
Rule 4	Medium	High	Low	Low	Setosa
Rule 5	Medium	High	Medium	Low	Setosa
Rule 6	Low	Medium	Medium	Low	Setosa
Rule 7	Medium	Low	High	Medium	Versicolor
Rule 8	Low	Low	Medium	Medium	Versicolor
Rule 9	High	Low	High	Medium	Versicolor
Rule 10	High	Medium	High	Medium	Versicolor
Rule 11	Medium	Medium	High	Medium	Versicolor
Rule 12	Low	Low	High	Medium	Versicolor
Rule 13	Medium	High	High	Medium	Versicolor
Rule 14	High	High	High	Medium	Versicolor
Rule 15	Low	Medium	High	Medium	Versicolor
Rule 16	Medium	Medium	High	High	Versicolor
Rule 17	High	High	High	High	Verginica
Rule 18	High	Medium	High	High	Verginica
Rule 19	High	Medium	High	Medium	Verginica
Rule 20	High	Low	High	High	Verginica
Rule 21	Medium	Medium	High	High	Verginica
Rule 22	Medium	Low	High	High	Verginica
Rule 23	Medium	Low	High	Medium	Verginica
Rule 24	Low	Low	High	Medium	Verginica
Rule 25	Medium	High	High	High	Verginica

Table 5.11: Sparse fuzzy rule base

Rule NO.	Sepal Width	Sepal Length	Petal Width	Petal Length	Class
Rule 1	Low	High	Medium	Low	Setosa
Rule 2	Low	High	Low	Low	Setosa
Rule 3	Low	Medium	Low	Low	Setosa
Rule 4	Medium	High	Medium	Low	Setosa
Rule 5	Medium	Low	High	Medium	Versicolor
Rule 6	Low	Low	Medium	Medium	Versicolor
Rule 7	Medium	Medium	High	Medium	Versicolor
Rule 8	Medium	High	High	Medium	Versicolor
Rule 9	High	High	High	Medium	Versicolor
Rule 10	Low	Medium	High	Medium	Versicolor
Rule 11	High	High	High	High	Verginica
Rule 12	High	Medium	High	High	Verginica
Rule 13	High	Low	High	High	Verginica
Rule 14	Medium	Medium	High	High	Verginica
Rule 15	Medium	Low	High	Medium	Verginica
Rule 16	Low	Low	High	Medium	Verginica
Rule 17	Medium	High	High	High	Verginica

Table 5.12: Illustrative example for the use of CRI (only 1 rule α-match)

INDEX	OBSERVATION				MATCH	RULE BASE				
	SW	SL	PW	PL	RULE	SW	SL	PW	PL	CLASS
18	5.1	3.5	1.4	0.3	Rule 2	Low	High	Low	Low	Setosa
1	5.1	3.5	1.4	0.2	Rule 2	Low	High	Low	Low	Setosa
20	5.1	3.8	1.5	0.3	Rule 1	Low	High	Medium	Low	Setosa
84	6	2.7	5.1	1.6	Rule 7	Medium	Medium	High	Medium	Versicolor
95	5.6	2.7	4.2	1.3	Rule 7	Medium	Medium	High	Medium	Versicolor
67	5.6	3	4.5	1.5	Rule 7	Medium	Medium	High	Medium	Versicolor
127	6.2	2.8	4.8	1.8	Rule 14	Medium	Medium	High	High	Verginica
128	6.1	3	4.9	1.8	Rule 14	Medium	Medium	High	High	Verginica
144	6.8	3.2	5.9	2.3	Rule 12	High	Medium	High	High	Verginica
109	6.7	2.5	5.8	1.8	Rule 13	High	Low	High	High	Verginica
115	5.8	2.8	5.1	2.4	Rule 14	Medium	Medium	High	High	Verginica

Table 5.13: Illustrative example for the use of CRI (more than 1 rule α-match)

INDEX	OBSERVATION				MATCH	FIRE RULE
	SW	SL	PW	PL		
70	5.6	2.5	3.9	1.1	Rule 5 Rule 15	Rule 5
93	5.8	2.6	4	1.2	Rule 5 Rule 15	Rule 5

first column lists the indices of the observations (in the data set), the second column gives the actual observations (along the four dimensions), and the third column presents the matched rules.

Table 5.12 shows all cases when only one rule is matched. Consider the first observation $O_1 = [5.1,\ 3.5,\ 1.4,\ 0.3]$ in Table 5.12 , through the check of α-cut overlapping it is known that O_1 overlaps with Rule 2 (and only Rule 2):

If Sepal Width is Low and Sepal Length is High and Petal Width is Low and Petal Length is Low then Class is Setosa.

Firing this rule leads to the class of this observation being identified to be Steosa.

There are cases when more than 1 rule are matched as shown in Table 5.13. Consider the first observation $O_1 = [5.6,\ 2.5,\ 3.9,\ 1.1]$, through the check of α-cut overlapping it is known that O_1 overlaps with Rule 5 and Rule 15. The matching degree is computed by finding the sum of all areas which are delimited by the overlap of two partially matched fuzzy sets of the rule and the observation. The rule with the highest matching value is Rule 5; firing this rule leads to the class of this observation being identified to be Versicolor.

5.3.2.3 Use of FRI When No Rules Matching the Observation

This case illustrates the effectiveness of utilising fuzzy rule interpolation (FRI). Table 5.14 shows the situations where there is no matching between observation and any rule in the given sparse rule base. The first column lists the indices of the observations in the data set, the second column gives the observation data (again regarding each of the four attributes), and the third column presents the two nearest rules determined by the use of the COG distance metric.

Table 5.14: Observations using fuzzy rule interpolation (FRI)

INDEX	OBSERVATION				NEAREST	RULE BASE				
	SW	SL	PW	PL	RULE	SW	SL	PW	PL	CLASS
42	4.5	2.3	1.3	0.3	Rule 3	Low	Medium	Low	Low	Setosa
					Rule 6	Low	Low	Medium	Medium	Versicolor
35	4.9	3.1	1.5	0.1	Rule 3	Low	Medium	Low	Low	Setosa
					Rule 2	Low	High	Low	Low	Setosa

Consider the first row in Table 5.14, given the observation $O = [4.5,\ 2.3,\ 1.3,\ 0.3]$, two rules, Rule 3 and Rule 6, are selected to be the nearest rules based on COG, in order to carry out fuzzy interpolation. For the first attribute Sepal Width (SW), the normalised weights of SW_{R3} and SW_{R6} are computed to be 0.5 and 0.5. According to Equation 5.10, the fuzzy term $SW^{''} = (4.3,\ 4.89,\ 5.3)$ is obtained. As $SW^{''}$ does not have the same COG as the input O_{SW}, the $zoom$ method is applied. Suppose that the $zoom$ method is used. According to Equation 5.12, $\gamma_{SW} = 0.93$ is obtained. The fuzzy term $SW^{''}$ is zoomed by γ_{SW} to $SW^{'} = (4,\ 4.56,\ 4.94)$. Similarly, SL has normalised weights 0.05 and 0.95 in constructing $SL^{''} = (2.04,\ 2.43,\ 2.63)$. With $\gamma_{SL} = 0.97$, $SL^{''}$ is zoomed to $SL^{'} = (1.98,\ 2.36,\ 2.56)$. PW has normalised weights 0.92 and 0.08 in constructing $PW^{''} = (1.04,\ 1.34,\ 1.52)$. With $\gamma_{PW} = 1$, $PW^{''}$ is zoomed to $PW^{'} = (1.04,\ 1.34,\ 1.52)$. PL has normalised weights 0.99 and 0.01 in constructing $PL^{''} = (0.11,\ 0.26,\ 0.62)$. With $\gamma_{PL} = 0.91$, $PL^{''}$ is zoomed to $PL^{'} = (0.1,\ 0.24,\ 0.56)$. The consequent fuzzy set $B^{''} = (1.38,\ 1.38,\ 1.38)$ can therefore be computed using the average weights of the attributes for two rules according to Equations 5.14 and 5.15. Thus, the intermediate class $B^{'} = (1.32,\ 1.32,\ 1.32)$ can be computed using the average of γ_{SW}, γ_{SL}, γ_{PW} and γ_{PL}, that is 0.95, with respect to Equations 5.16 and 5.17. From this, the scale and move transformations are applied, resulting in the required interpolated value, that is $B^{*} = (1.32,\ 1.32,\ 1.32)$. Through defuzzification, the result indicates that the given observation stands for the class Setosa (which correctly matches the underlying class of this observation). As a summary, Figure 5.9 shows this interpolation progress. Consider the second row in Table 5.14, given the observation $O = [4.9,\ 3.1,\ 1.5,\ 0.1]$, two rules, Rule 3 and Rule 2, are selected to be the nearest rules based on COG, in order to carry out fuzzy interpolation. For the first attribute Sepal Width (SW), the normalised weights of SW_{R3} and SW_{R2} are computed to be 0.5 and 0.5. According to Equation 5.10, the

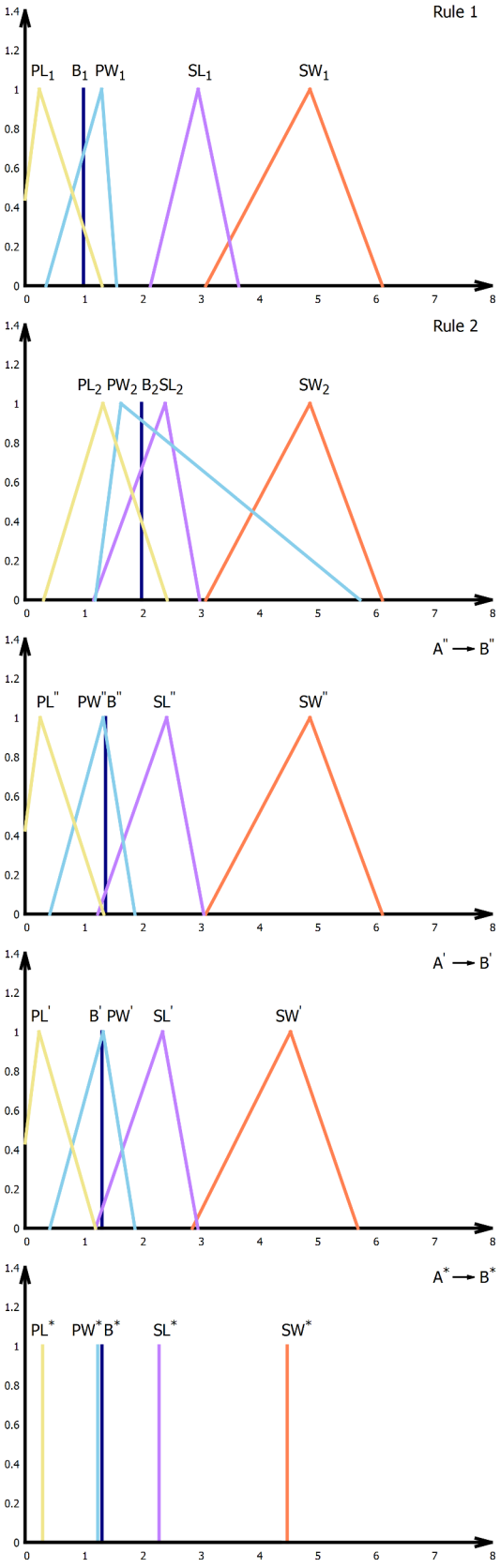


Figure 5.9: Case 1 transformation progress

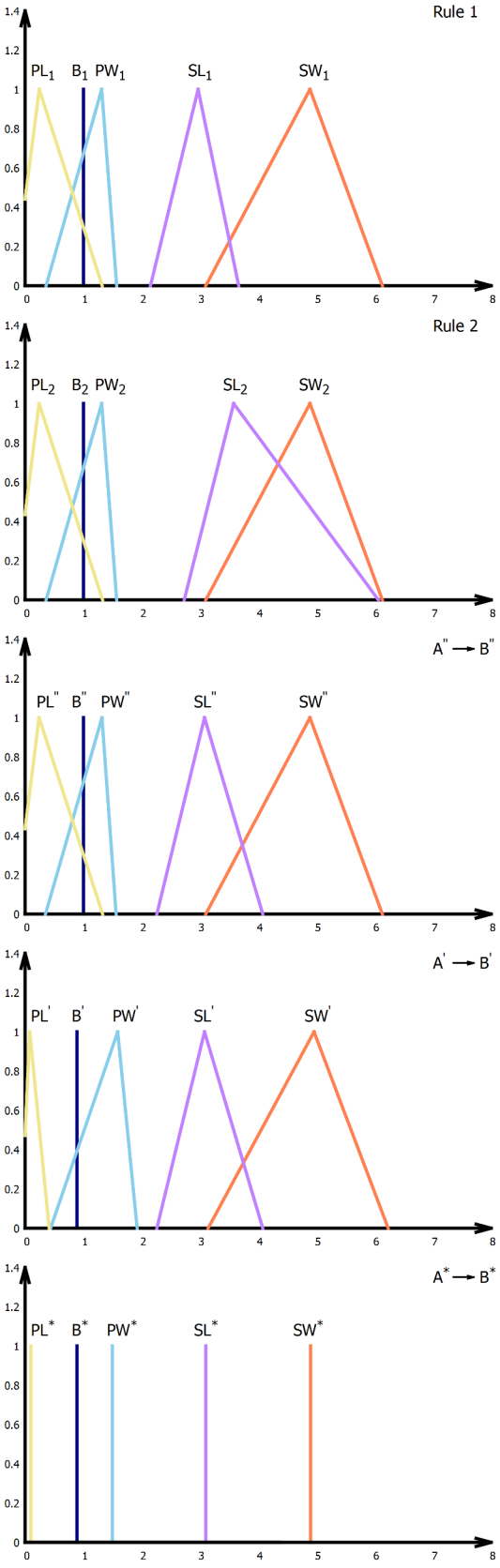


Figure 5.10: Case 2 transformation progress

fuzzy term $SW^{''} = (4.3,\ 4.89,\ 5.3)$ is obtained. As $SW^{''}$ does not have the same COG as the input O_{SW}, the $zoom$ method is applied. Suppose that the $zoom$ method is used. According to Equation 5.12, $\gamma_{SW} = 1.01$ is obtained. The fuzzy term $SW^{''}$ is zoomed by γ_{SW} to $SW^{'} = (4.36,\ 4.96,\ 5.38)$. Similarly, SL has normalised weights 0.82 and 0.18 in constructing $SL^{''} = (2.81,\ 3.08,\ 3.41)$. With $\gamma_{SL} = 1$, $SL^{''}$ is zoomed to $SL^{'} = (2.81,\ 3.08,\ 3.41)$. PW has normalised weights 0.5 and 0.5 in constructing $PW^{''} = (1,\ 1.32,\ 1.4)$. With $\gamma_{PW} = 1.21$, $PW^{''}$ is zoomed to $PW^{'} = (1.21,\ 1.59,\ 1.7)$. PL has normalised weights 0.5 and 0.5 in constructing $PL^{''} = (0.1,\ 0.24,\ 0.6)$. With $\gamma_{PL} = 0.32$, $PL^{''}$ is zoomed to $PL^{'} = (0.03,\ 0.08,\ 0.19)$. The consequent fuzzy set $B^{''} = (1,\ 1,\ 1)$ can therefore be computed using the average weights of the attributes for two rules according to Equations 5.14 and 5.15. Thus, the intermediate class $B^{'} = (0.89,\ 0.89,\ 0.89)$ can be computed using the average of γ_{SW}, γ_{SL}, γ_{PW} and γ_{PL}, that is 0.89, with respect to Equations 5.16 and 5.17. From this, the scale and move transformations are applied, resulting in the required interpolated value, that is $B^{*} = (0.89,\ 0.89,\ 0.89)$. Through defuzzification, the result indicates that the given observation stands for the class Setosa (which correctly matches the underlying class of this observation). As a summary, Figure 5.10 shows this interpolation progress.

5.4 Application to Mammographic Image Analysis

Breast cancer is a major health issue, and the most common amongst women in the EU. It is estimated that 8-13% of all women will develop breast cancer at some point during their lives [37, 38]. Furthermore, in the EU and US, breast cancer is recognised as the leading cause of death of women in their 40's [37, 38, 156]. Although increased incidence of breast cancer has been recorded, so too has the level of early detection through the screening of potential occurrence using mammographic imaging and expert opinion. However, even expert radiologists can sometimes fail to detect a significant proportion of mammographic abnormalities. In addition, a large number of detected abnormalities are usually discovered to be benign following medical investigation.

Existing mammographic Computer Aided Diagnosis (CAD) systems [157] concentrate on the detection and classification of mammographic abnormalities. As breast tissue density increases however, the effectiveness of such systems in detecting mammographic abnormalities is reduced significantly. Also, there is a strong correlation between mammographic breast tissue density and the risk of development of breast

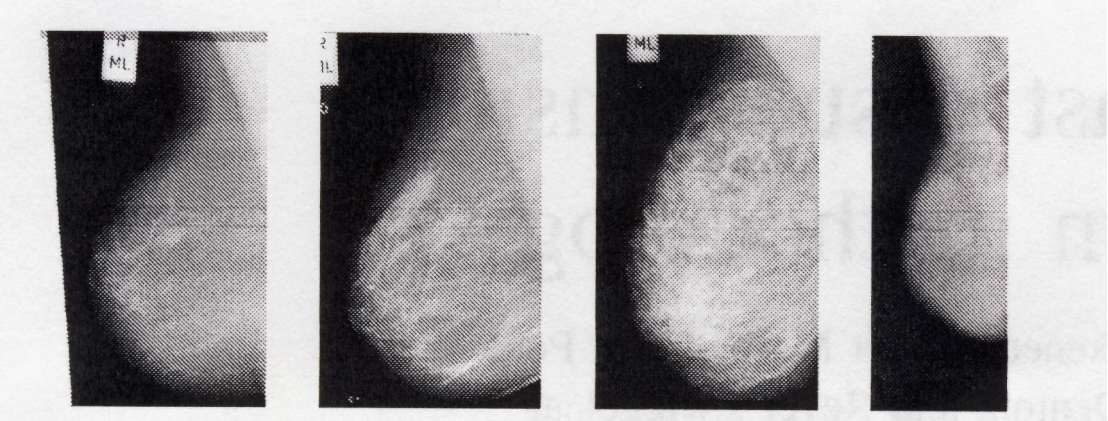


Figure 5.11: Example mammograms where breast tissue density increases from L-R corresponding to BIRADS class I(far left) to class IV (far right)

cancer [158, 159, 160]. Automatic classification which has the ability to consider tissue density when searching for mammographic abnormalities is therefore highly desirable. It must be stressed at this point that the problem under consideration here is mammographic risk analysis (MRA) rather than mammographic diagnosis from images, an area where many publications have been written in terms of the application of machine learning techniques [161, 162, 163, 164]. As such, MRA aims to classify image data objects into one of four BIRADS categories [165] shown in Figure 5.11 which relates to the tissue type found in each mammogram. That is, the purpose of MRA is not to classify breast tissue abnormalities, but rather to give an indication of the tissue density.

Knowledge discovery from images often requires the maximisation of all of the information contained within the image. This means that initially large numbers of features are often extracted from the image. These features typically contain high levels of redundancy, irrelevancy, and noise. However, given that it is not known a-priori which features are most valuable and which are not, this is a necessary step. In the present application, the popular, and readily available, fuzzy rough feature selection (FRFS) method is employed in an attempt to identify the most valuable features such that the process of extracting large amounts of features can be avoided. The selected features then fed back into the extraction phase ensuring that only those features need to be identified. In addition to the feature selection approach, the integrated fuzzy rule based classification system described previously is applied to the image data. This classifier is compared with possible alternative approaches, demonstrating a significant increase in performance.

5.4.1 Experimental System Overview

As mentioned previously, the problem considered in this work is that of mammographic risk analysis, where mammographic breast tissue density information extracted from images is used to assess how likely a woman is to develop breast cancer.

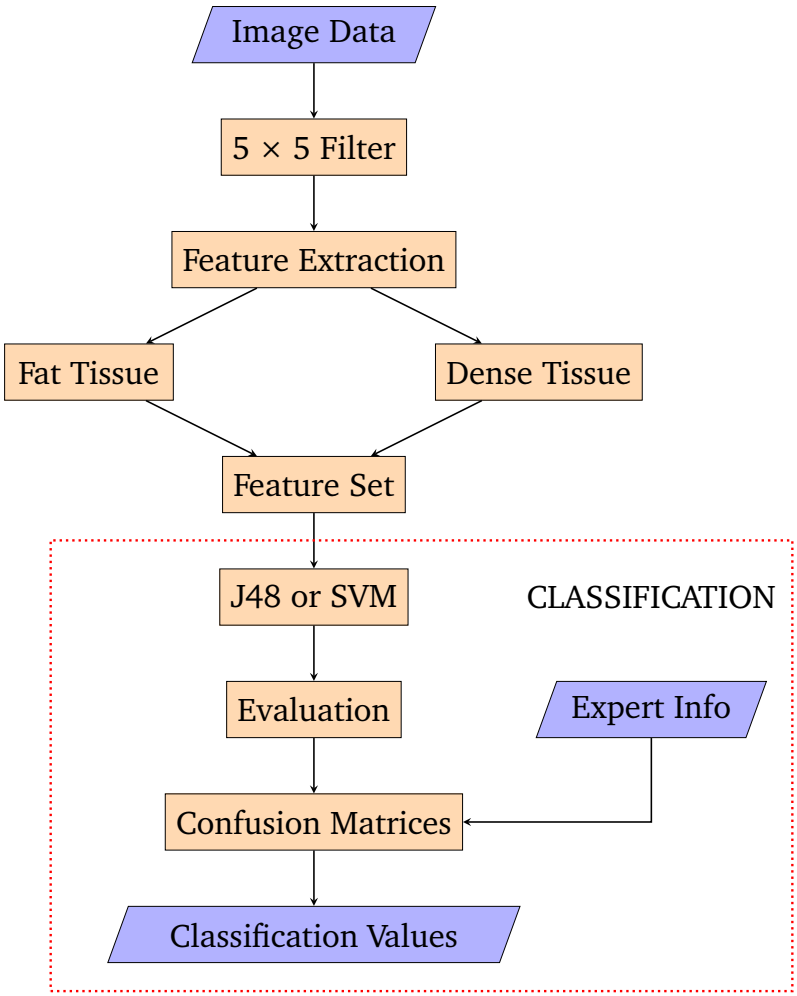


Figure 5.12: Mammographic density classification

The implemented application process is outlined in Figure 5.12, with detailed background described in [166]. The initial stages involve the segmentation and filtering of the mammographic images: all mammograms are pre-processed to identify the breast region and remove image background, labels, and pectoral muscle areas. This segmentation step results in a very minor loss of skin-line pixels in the breast area, however these pixels are not required for tissue estimation. Then, a feature extraction step is performed, fuzzy c-means (FCM) algorithm [130] is employed

which results in the division of the breast into two clusters. A co-occurrence matrix (which is essentially a 2D histogram) is next, used to derive a feature set which results in 10 features to describe morphological characteristics and 216 for the texture information (226 total). This feature set is labelled using the consensus opinion of 3 experts to assign a label to each object mammogram using the BIRADS classification [165]. This consensus is determined where the classification for a given mammogram, which two or three radiologists agreed upon (majority vote) is selected as the consensus class. If all experts classified a single mammogram differently, the median value is chosen as consensus opinion. The divergence in the opinion of the experts, is a major factor which often frustrates the use of automatic methods. This highlights the need to remove inter-observer (inter-operator) variability through the development of more autonomous approaches.

In this work the classification step is extended with a dimensionality reduction phase and a classification phase. The existing feature set is used, as is the consensus expert labelling of the data. Figure 5.13 shows the overall implementation of the mammographic data analysis process in which knowledge can be efficiently learned from the (mammographic) training data and applied to real-world risk assessment problems.

In this work, the focus lies in the implementation of fuzzy techniques for the dimensionality reduction and classifier learner steps. The approach for the feature extraction step employed here is documented in [166], however there is no reason why future work could not include a deep learning method to accomplish this (see conclusion chapter for further discussion).

Efficient and accurate classification of mammographic imaging is of high importance. Any improvement in accuracy for automatic mammographic classification systems can result in a reduction in the amount of required expert analysis thus improving the time taken to perform breast abnormality risk assessment. The data in mammographic imaging is real-valued and can be noisy. Any classifier employed must therefore have the ability to deal with such data. Conventional crisp methods require that the real-valued data be discretised and thus, may result in information loss. The methods described in this work require no discretisation and use only the information contained within the data.

The most common approach to developing expressive and human readable representations of knowledge is the use of IF-THEN production rules [167]. In order

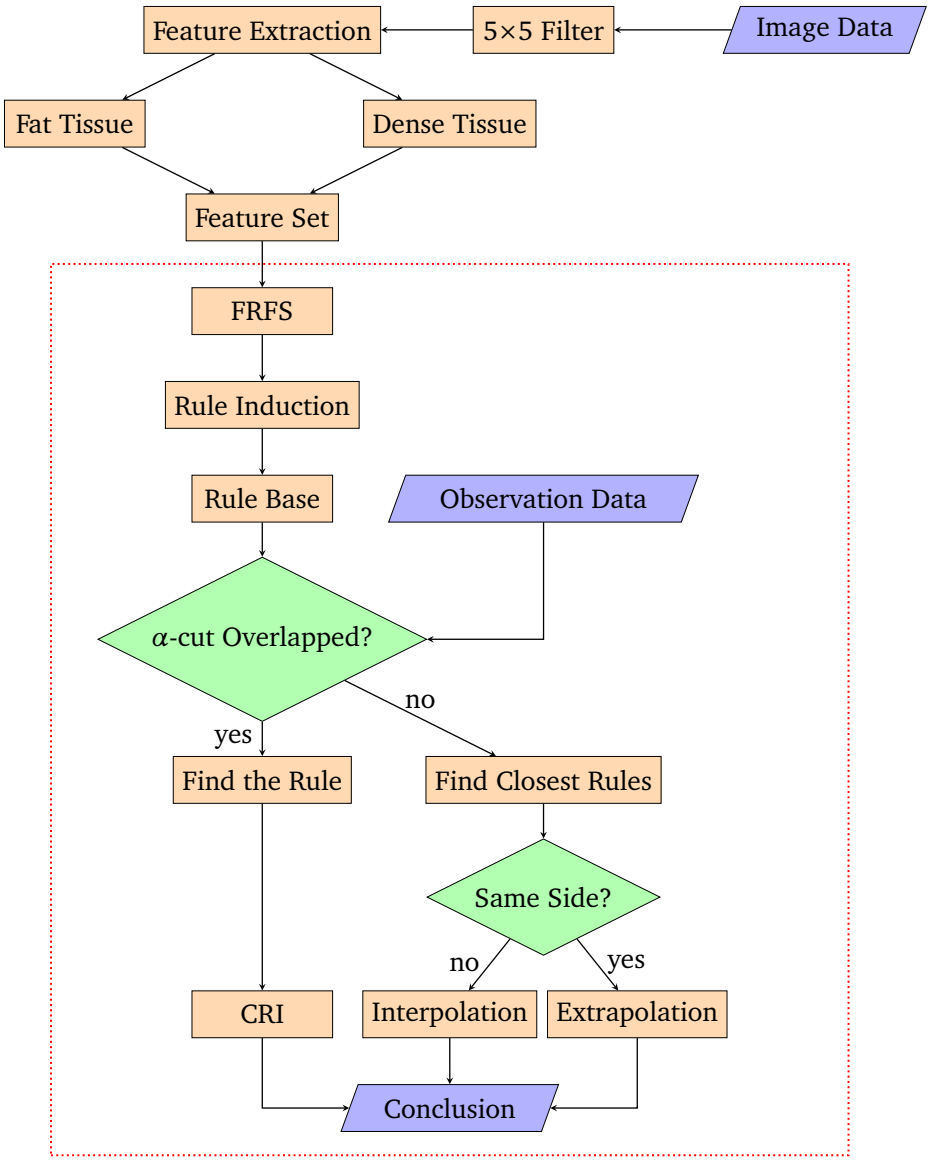


Figure 5.13: Unified framework for mammographic data analysis

to speed up rule induction learning algorithms (RIA) and reduce rule complexity, a preprocessing step is required. This is particularly important for tasks where learned rule sets need to be regularly updated to reflect the changes in the description of domain features. This step reduces the dimensionality of potentially very large feature sets while minimising the loss of information needed for rule induction. It has an advantageous side-effect in that it removes redundancy from the historical data. This also helps to simplify the design and implementation of the actual pattern classifier itself, by determining what features should be made available to the system. In addition, the reduced input dimensionality increases the processing speed of the

classifier, leading to better response times. Most significant, however, is the fact that the technique employed here preserves the semantics of the surviving features following the removal of any redundancy. This is essential in satisfying the requirement of user interpretability of the generated knowledge model, as well as ensuring the transparency of the classification process.

A simple approach described in Section 5.1 is utilized to perform fuzzy rule induction. In so doing, training data is equivalently translated into a fuzzy IF-THEN rule, resulting in a fuzzy rule base. However, in many real-world problems, data provided to conduct such learning may not completely cover the entire problem space. As such, the resultant rule base may have gaps scattered in the problem space, leading to the situation where rule-firing by matching a given observation with the learned clusters becomes void. In this case, as argued previously, fuzzy rule interpolation offers a promising means to perform approximate inference. Depending on the nature of the rule base either fuzzy inference (CRI) or interpolation (FRI) may be employed to draw conclusions from given unclassified images.

5.4.2 Experimentation

In this section the results of applying the previously described classifier and alternative conventional classifiers are presented and discussed. Initially the classifiers are applied to the unreduced extracted feature data - i.e., data on which feature selection has not been utilised. Classification is then performed on data which has been reduced with individual classifiers as shown in Figure 5.14.

5.4.2.1 Mammographic Risk Analysis Data

The dataset considered here is available in the public domain: the Mammographic Image Analysis Society (MIAS) database [168], composed of Medio-Lateral-Oblique (MLO) left and right mammograms from 161 women (322 objects). Each mammogram object is represented by 281 features extracted using the process described [166]. The spatial resolution of the images is $50\mu m \times 50\mu m$ and quantized to 8 bits with a linear optical density in the range 0 - 3.2 (Optical density, also known as Dynamic Range, is the scanner's ability to "see" all tones available. The total tonal measurement is on a scale of 0.0 (white) to 4.0 (black)).

The class labels for each mammogram are assigned by three experts' consensus opinion. There are four discrete labels for representing density classes, encoded as

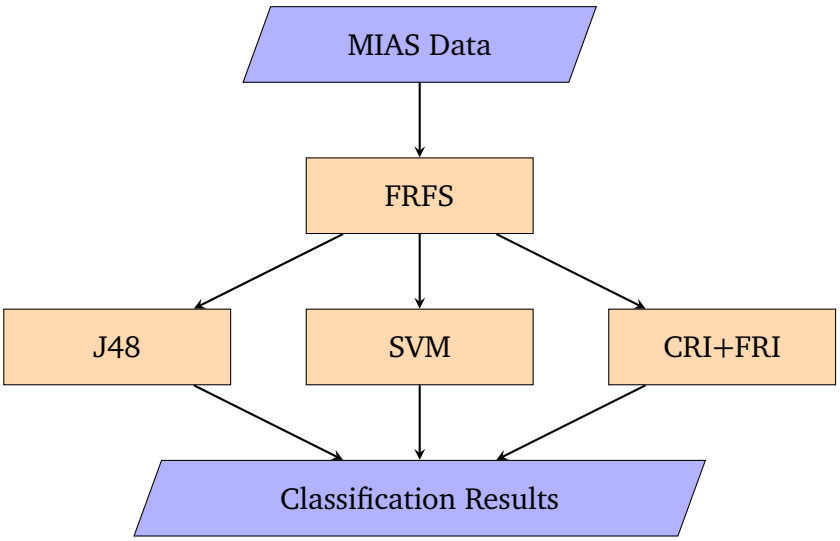


Figure 5.14: Experimental setup

integers from 1 to 4 with 1 representing a breast that is entirely fatty and 4 a breast that is extremely dense.

5.4.2.2 Experimental Setup

In systematically conducting the experiments, both two-fold cross validation (2-FCV) and ten-fold cross validation (10-FCV) are employed. 2-FCV is performed 100 times and 10-FCV is performed 10 times in order to lessen the impact of random factors within the algorithms, these evaluations are then aggregated to produce the final experimental outcomes as shown in Tables 5.15, 5.16, 5.18 and 5.19. Paired t-test is utilised to assess any statistical significance of the differences between the algorithms' results, with the parameter $p = 0.05$. To reduce the overall computation cost for building the fuzzy rule bases and also, for performing the inference with the learned rule bases, the feature selection tools FRFS is used here to pre-process the data reducing the data dimensionality. Then FCM is run on the dimensionality-reduced training set to induce fuzzy classification rules and membership functions. The number of clusters K is set to 3, 4 and 5.

5.4.2.3 Unreduced Data

The classification accuracy results for the unreduced data are presented in this section. This is achieved by applying each of the classifiers to the dataset, giving a background against which to make subsequent comparative studies.

Table 5.15 lists the averaged correct classification rates over these runs for the given dataset in 2-FCV using different classifiers. Clearly, the accuracy using the proposed classification system is significantly better than that achieved using J48 [169] and SVM [131, 166]. Both J48 and SVM are sensitive to the completeness of the training dataset, if the dataset can't cover all the possible values of the input variables, the rule-base created from the dataset is too sparse then it affects the prediction and result of the system due to insufficient information. Note that when $K = 5$, the proposed method delivers the best performance. This is because in this case, the rule base generated from the dataset is more dense than those achieved when K is 3 or 4.

Considering the classification accuracy results illustrated in Table 5.16. The proposed approach seems to have a clear advantage over J48 when $K = 5$. The results achieved using CRI+FRI are much better than those achieved using SVM. In 10-FCV, the number of data used to build the rule base is larger than that used in the 2-FCV. So the accuracy is improved compared with the accuracy achieved in the 2-FCV. Again CRI+FRI is more robust in the presence of incomplete dataset.

Table 5.15: 2-fold cross validation accuracy (%) based on unreduced dataset: The better performance on mammographic dataset is highlighted in boldface, the sign "(*)/(v)"indicates that corresponding result achieved using proposed method is significantly ($p \leq 0.05$) worse/better than that using other classifiers

J48	SVM	CRI+FRI (K=3)	CRI+FRI (K=4)	CRI+FRI (K=5)
61.58±3.26	59.81±0.73	64.84±3.22	64.53±2.86	**65.90±3.72(v)**

Table 5.16: 10-fold cross validation accuracy (%) based on unreduced dataset: The better performance on mammographic dataset is highlighted in boldface, the sign "(*)/(v)"indicates that corresponding result achieved using proposed method is significantly ($p \leq 0.05$) worse/better than that using other classifiers

J48	SVM	CRI+FRI (K=3)	CRI+FRI (K=4)	CRI+FRI (K=5)
66.15±1.73	63.85±0.64	64.02±1.23	66.05±2.65	**67.50±0.83(v)**

5.4.2.4 Reduced Data

In this section the results of classifying the MIAS dataset following feature selection are presented. Classification accuracy results are provided for different classifiers in

both 2-FCV and 10-FCV. In Table 5.17, the indices of the selected features are shown. Note that a substantial level of dimensionality reduction (97.17%) is achieved.

Table 5.17: Indices of selected features

1, 5, 156, 200, 205, 231, 234

Table 5.18 lists the averaged correct classification rates over these runs for the given datasets in 2-FCV. Clearly, the accuracy using the proposed classification system is better than that achieved using J48 and SVM.

Table 5.19 lists the averaged correct classification rates over these runs for the given datasets in 10-FCV. It is clear that the accuracy using the proposed classification system is improved than that achieved using conventional classifiers.

The general trends across all cases are rather similar, although the accuracy is lower than that achieved using unreduced data. In this case, feature selection leads to the loss of useful information to some extent.

Table 5.18: 2-fold cross validation accuracy (%) based on reduced dataset: The better performance on mammographic dataset is highlighted in boldface, the sign "(*)/(v)"indicates that corresponding result achieved using proposed method is significantly ($p \leq 0.05$) worse/better than that using other classifiers

J48	SVM	CRI+FRI (K=3)	CRI+FRI (K=4)	CRI+FRI (K=5)
60.00±2.53	45.96±2.54	56.15±3.95	**61.33±4.39(v)**	60.99±3.74

Table 5.19: 10-fold cross validation accuracy (%) based on reduced dataset: The better performance on mammographic dataset is highlighted in boldface, the sign "(*)/(v)"indicates that corresponding result achieved using proposed method is significantly ($p \leq 0.05$) worse/better than that using other classifiers

J48	SVM	CRI+FRI (K=3)	CRI+FRI (K=4)	CRI+FRI (K=5)
58.75±2.29	52.17±1.00	59.44±2.3	**63.00±2.09(v)**	62.31±2.18

5.5 Summary

Feature pattern-based fuzzy IF-THEN rules offer an expressive and interpretable form of imprecise data-driven classification. However, generation of fuzzy rules that may jointly cover the full, or at least a wide range of, problem domain from experienced data is subject to the completeness of such data. Missing information will lead to gaps within the resultant explicit knowledge, which would reduce the practical effectiveness of the intelligent classifiers which rely on the use of the resulting sparse rule bases. This chapter has presented a conceptually simple system which supports the development of potentially powerful fuzzy rule-based classification systems, by organically integrating conventional fuzzy rule-based inference and fuzzy rule interpolation. To be self-complete, it has also described a rule induction algorithm although any other fuzzy rule learning mechanism may be used as an alternative. The work has been systematically tested on a range of benchmark datasets, demonstrating the efficacy of the proposed approach. This chapter has also demonstrated the application of fuzzy rule interpolation-aided classification system for mammographic risk analysis. In particular, it has demonstrated how the classification accuracy for mammographic risk-analysis can be increased significantly compared with that achievable using conventional classifiers.

Chapter 6

Conclusion

This chapter presents a summary of the research as detailed in the preceding chapters. Having reviewed and compared a number of deep learning techniques in the literature, two novel learning architectures have been proposed to minimize the computational effort while obtaining informative features. Systematic experiments demonstrate that the newly proposed methods perform better than traditional deep learning networks on the popular MNIST dataset of handwritten digits. The thesis has also demonstrated that the developed fuzzy rule interpolation-aided reasoning system has effectively boosted the performance when applied to a challenging real-world problem: mammographic risk analysis. The proposed approach renders the reasoning system more accurate and faster by exploiting both CRI and FRI. The capabilities and potential of the developed applications have been experimentally validated, and compared with conventional rule-based inference work. The conclusion also presents a number of initial thoughts about the directions for future research.

6.1 Summary of Thesis

Chapter 2 first gives a thorough review of the most influential and successful deep learning networks, including Deep Belief Networks (DBNs) [2], Convolutional Neural Nets (CNNs) [4], and Deep Spatio-Temporal Inference Network (DeSTIN) [5]. Unfortunately, there is a main drawback among these types of network. A significant computation effort is demanded when using them to do the feature extraction. For

example, as a representative application of DeSTIN, the existing work for handwritten digit recognition employs a network of 4 layers [6]. This may introduce considerable overheads on computation and therefore, may offset the potential benefit on the efficiency gained by the entire feature extraction process. An alternative approach is desirable.

Chapter 3 has presented a novel approach for image classification, by integrating the concepts of deep machine learning and feature interpolation. In particular, a recently introduced learning architecture, the Deep Spatio-Temporal Inference Network (DeSTIN) is employed to perform feature extraction for support vector machine (SVM) based image classification. Linear interpolation and Newton polynomial interpolation are each applied to support the classification. This approach converts feature sets of an originally low-dimensionality into those of a significantly higher dimensionality while gaining overall computational simplification. The work is tested against the popular MNIST dataset of handwritten digits. Experimental results indicate that the proposed approach is highly promising.

Chapter 4 has presented a novel approach to developing a learning network that is of simple topological structure for pattern recognition, through the exploitation of a standard data clustering mechanism. This work has been tested using the popular MNIST dataset, in comparison with four different deep learning techniques. Systematic experimental results demonstrate that the proposed approach is capable of efficiently extracting features suitable for subsequent image pattern recognition tasks, ensuring high accuracy.

In the second part of Chapter 2, a detailed literature review that covers the fuzzy inference systems (FIS), compositional rules of inference (CRI) and fuzzy rule interpolation (FRI) was presented. CRI is a classical inference approach in systems using dense rule bases and important CRI methods: Mamdani inference, TSK inference and Type-2 inference have been outlined. However, in most real life applications, rule bases are sparse and FRI is a quite effective approach for reasoning with sparse rule bases. For this reason, a survey of several typical FRI approaches was made in which the key common notions and mechanisms of the reviewed algorithms were extracted.

Feature pattern-based fuzzy IF-THEN rules offer an expressive and interpretable form of imprecise data-driven classification. However, the effectiveness of using fuzzy

rules generated from experienced data is subject to the completeness of such data. Missing information will lead to gaps within the resultant explicit knowledge, which may reduce the accuracy of the intelligent pattern classifiers that rely on the use of such sparse rule bases. Chapter 5 has presented a conceptually simple system which supports the development of potentially powerful fuzzy rule-based classification systems, by organically integrating conventional fuzzy rule-based inference and fuzzy rule interpolation. It applies α-cut to efficiently check whether direct inference can be performed using compositional rule of inference (CRI) in spite of the sparsity in the rule base. If more than one α-cut match between a given observation and any rule antecedent are found, then the rule which has the largest overlap is fired to derive the conclusion, using CRI. As such, it employs an interpolation method only when interpolation is essential, that is when there is no matching between the observation and any rule in the rule-base. This helps expedite the operation of the integrated system whilst improving the accuracy of the overall inference mechanism. The work has been systematically tested on a range of benchmark datasets, demonstrating the efficacy of the proposed approach.

Furthermore, an application of the proposed fuzzy rule interpolation-aided framework to the problem domain of mammographic risk analysis has been presented in Chapter 5.

6.2 Future Work

Although promising, much can be done to further improve the work presented in this thesis.The following addresses a number of interesting issues whose successful solution will go towards establishing the current research on a more robust foundation.

6.2.1 Short Term Tasks

This section discusses extensions, enhancements or ongoing tasks that could be readily implemented if additional time were available.

6.2.1.1 DESTINI Improvement

In this research work, either linear interpolation or Newton interpolation is applied to the DeSTIN extracted features. It is interesting to investigate whether either of

the two simple interpolation mechanisms will work when different original feature extraction methods are used. Also, a combined application of both linear and Newton interpolation may help further enrich the feature space. In addition, it is very interesting to apply the work to more complex application domains (e.g., to Mars images which vary significantly in terms of intensity, scale and rotation, and are blurred with measurement and transmission noise [44]). Finally, it is worth exploring whether imposing a certain selection of interpolated features may further reduce the overall computation cost for classification [155, 149].

6.2.1.2 CLSN Improvement

Although CLSN delivers good results when applied to the MNIST dataset. It would be interesting to examine whether interpolation techniques could be utilised on the extracted features to help enrich the feature space without causing much increase in computation. Also, it would be useful to explore the possibility of extending the work to more complex problem domains [170, 171]. The potential of this work may be further strengthened if dynamic movement detection would be enabled as and when a sequence of images are presented [172].

6.2.1.3 Reasoning System Improvement

The proposed fuzzy rule interpolation-aided classification system is promising. Further work remains in a number of aspects. In particular, it would be very interesting to apply the proposed approach to more complex application domains (e.g., to problems involving big data). When dealing with a complicated problem, however sparse, the learned rule base may contain a very large number of rules. Thus, a more efficient integration of compositional rule inference and rule interpolation may be required that can identify certain rules to be removed without affecting the overall classification performance (this being thanks to the power of rule interpolation). Also, to have a common ground for fair comparison, this work has employed a standard clustering based rule induction algorithm to generate the initial rule base, without fine-tuning the learned rules or the membership functions that define the feature values. Alternative learning mechanisms with optimisation [173] may be employed to further improve the present approach. In this research work, for the purpose of preliminary investigation and experimentation, the T-FRI approach is employed. To further generalize the proposed reasoning system, is interesting to investigate the

use of some different FRI techniques [27, 29, 174, 26, 175, 176, 177]. This is of course a systematic approach that requires a lot of additional experimentation.

6.2.2 Longer Term Developments

This section proposes several future directions of research.

6.2.2.1 Dynamic Partition

There is one important assumption in the proposed reasoning system: an initial fixed partitioning level for a given attribute. For the current implementation, this is sufficient to evaluate the potential of proposed system. However, fixed partitioning limits the generalised concept of this approach and also affects the accuracy of the proposed framework. This is because it decides the number of partitions for every attribute of instance at an early stage irrespective of all the later operations used in the system. This may permanently direct the later operations and affect their outcomes. It is therefore important to be able to obtain the best partitioning during the reasoning process in order to find better quality clusters and, eventually, more precise new rules for reasoning.

6.2.2.2 A Unified Reasoning System

The present feature extraction method adopted in the proposed work may introduce a lot of redundant and noisy features. So a feature selection process is needed. This step involves a considerable extra computation, which may restrict the approach to be applied to a real-time application. The features extracted from CLSN are effective features and the dimensionality of the feature sets is far less than that of current approach. The benefit of adopting CSLN is that the amount of time and computational effort required in the feature extraction phase can be reduced.

In addition, the use of fewer features means that any algorithms employed in both the training and testing phases of the classifier are potentially more accurate as there are fewer noisy features present. Furthermore, fewer features means less computational overhead and hence the task is performed in less time.

Finally, a unified reasoning system is foreseen as one of the most important further developments of this research. Such an integrated, systematic approach may enjoy the benefits of deep learning, CRI and FRI when given a complex data domain, fully addressing the issues from feature extraction through to data classification.

Appendix A

Publications Arising from the Thesis

A number of publications have been generated from the research carried out within the PhD project. Below lists the resultant publications that are in close relevance to the thesis, including both papers already published and articles submitted for review.

A.1 Journal Articles

1. Y. Zhang and C. Shang, Combining newton interpolation and deep learning for image classification, Electronics Letters, vol. 51, no. 1, pp. 40-42, 2014.
2. Y. Zhang, C. Shang, F. Chao, N. Naik and Q. Shen, Enriching data-driven fuzzy rule-based classification with fuzzy rule interpolation, Under review for journal publication.
3. Y. Zhang, C. Shang, and Q. Shen, Clustering supported learning network with application to handwritten digit recognition, Under review for journal publication.

A.2 Conference Papers

4. Y. Zhang, C. Shang, and Q. Shen, Interpolating destin features for image classification, Proceedings of the 2013 UK Workshop on Computational Intelligence, pp. 292-298, 2013.

5. Y. Zhang, C. Shang, and Q. Shen, Interpolating deep spatio-temporal inference network features for image classification, Proceedings of the 2014 IEEE International Joint Conference on Neural Networks, pp. 1819-1826, 2014.

6. Y. Zhang, C. Shang, and Q. Shen, Interpolation aided fuzzy image classification, Proceedings of the 2015 IEEE International Conference on Fuzzy Systems, pp. 1-7, 2015.

Bibliography

[1] R. Bellman, "Dynamic programming and lagrange multipliers," *Proceedings of the National Academy of Sciences*, vol. 42, no. 10, pp. 767–769, 1956.

[2] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[3] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, "Greedy layer-wise training of deep networks," *Advances in Neural Information Processing Systems*, vol. 19, p. 153, 2007.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] I. Arel, D. Rose, and B. Coop, "Destin: A scalable deep learning architecture with application to high-dimensional robust pattern recognition," in *Proc. AAAI Workshop Biologically Inspired Cognitive Architectures (BICA)*, 2009.

[6] T. P. Karnowski, I. Arel, and D. Rose, "Deep spatiotemporal feature learning with application to image classification," in *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*. IEEE, 2010, pp. 883–888.

[7] D. Dubois and H. Prade, "What are fuzzy rules and how to use them," *Fuzzy Sets and Systems*, vol. 84, no. 2, pp. 169–185, 1996.

[8] J. Lawry, "A framework for linguistic modelling," *Artificial Intelligence*, vol. 155, no. 1, pp. 1–39, 2004.

[9] D. Driankov, H. Hellendoorn, and M. Reinfrank, *An introduction to fuzzy control*. Springer Science & Business Media, 2013.

[10] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic*. Prentice hall New Jersey, 1995, vol. 4.

[11] L.-X. Wang, *A course in fuzzy systems*. Prentice-Hall press, USA, 1999.

[12] J. Buckley, W. Siler, and D. Tucker, "A fuzzy expert system," *Fuzzy Sets and Systems*, vol. 20, no. 1, pp. 1–16, 1986.

[13] X.-J. Ma, Z.-Q. Sun, and Y.-Y. He, "Analysis and design of fuzzy controller and fuzzy observer," *Fuzzy Systems, IEEE Transactions on*, vol. 6, no. 1, pp. 41–51, 1998.

[14] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[15] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[16] K. Nozaki, H. Ishibuchi, and H. Tanaka, "A simple but powerful heuristic method for generating fuzzy rules from numerical data," *Fuzzy Sets and Systems*, vol. 86, no. 3, pp. 251–270, 1997.

[17] L.-X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 22, no. 6, pp. 1414–1427, 1992.

[18] Y. Yuan and H. Zhuang, "A genetic algorithm for generating fuzzy classification rules," *Fuzzy Sets and Systems*, vol. 84, no. 1, pp. 1–19, 1996.

[19] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 1, pp. 28–44, 1973.

[20] S.-M. Chen, "A new approach to handling fuzzy decision-making problems," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 18, no. 6, pp. 1012–1016, 1988.

[21] ——, "A weighted fuzzy reasoning algorithm for medical diagnosis," *Decision Support Systems*, vol. 11, no. 1, pp. 37–43, 1994.

[22] I. Turksen and Z. Zhong, "An approximate analogical reasoning approach based on similarity measures," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 18, no. 6, pp. 1049–1056, 1988.

[23] ——, "An approximate analogical reasoning schema based on similarity measures and interval-valued fuzzy sets," *Fuzzy Sets and Systems*, vol. 34, no. 3, pp. 323–346, 1990.

[24] D. S. Yeung and E. C. C. Tsang, "A comparative study on similarity-based fuzzy reasoning methods," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 27, no. 2, pp. 216–227, 1997.

[25] K. W. Wong, D. Tikk, T. D. Gedeon, and L. T. Kóczy, "Fuzzy rule interpolation for multidimensional input spaces with applications: A case study," *Fuzzy Systems, IEEE Transactions on*, vol. 13, no. 6, pp. 809–819, 2005.

[26] Y.-C. Chang, S.-M. Chen, and C.-J. Liau, "Fuzzy interpolative reasoning for sparse fuzzy-rule-based systems based on the areas of fuzzy sets," *Fuzzy Systems, IEEE Transactions on*, vol. 16, no. 5, pp. 1285–1301, 2008.

[27] L. Yang and Q. Shen, "Closed form fuzzy interpolation," *Fuzzy Sets and Systems*, vol. 225, pp. 1–22, 2013.

[28] L. Kóczy, K. Hirota, and L. Muresan, "Rule interpolation by α-level sets in fuzzy approximate reasoning," *J. BUSEFAL, Automne, URA-CNRS*, vol. 46, pp. 115–123, 1991.

[29] S.-H. Cheng, S.-M. Chen, and C.-L. Chen, "Adaptive fuzzy interpolation based on ranking values of polygonal fuzzy sets and similarity measures between polygonal fuzzy sets," *Information Sciences*, 2016.

[30] Z. Huang and Q. Shen, "Fuzzy interpolative reasoning via scale and move transformations," *Fuzzy Systems, IEEE Transactions on*, vol. 14, no. 2, pp. 340–359, 2006.

[31] ——, "Fuzzy interpolation and extrapolation: A practical approach," *Fuzzy Systems, IEEE Transactions on*, vol. 16, no. 1, pp. 13–28, 2008.

[32] Y. LeCun and C. Cortes, "The mnist database of handwritten digits," 1998.

[33] Y. Zhang, C. Shang, and Q. Shen, "Interpolating destin features for image classification," in *Computational Intelligence (UKCI), 2013 13th UK Workshop on*. IEEE, 2013, pp. 292–298.

[34] Y. Zhang and C. Shang, "Combining newton interpolation and deep learning for image classification," *Electronics Letters*, vol. 51, no. 1, pp. 40–42, 2014.

[35] Y. Zhang, C. Shang, and Q. Shen, "Interpolating deep spatio-temporal inference network features for image classification," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 1819–1826.

[36] ——, "Interpolation aided fuzzy image classification," in *Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1–7.

[37] F. Bray, P. McCarron, and D. M. Parkin, "The changing global patterns of female breast cancer incidence and mortality," *Childhood*, vol. 4, no. 5, pp. 229–239, 2004.

[38] E. Jougla, G. Salem, S. Gancel, and V. Michel, "Health statistics-atlas on mortality in the european union," *European Communities*, 2003.

[39] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[40] A. Rockel, R. W. Hiorns, and T. P. Powell, "The basic uniformity in structure of the neocortex," *Brain*, vol. 103, no. 2, pp. 221–244, 1980.

[41] G. Wallis and H. Bülthoff, "Learning to recognize objects," *Trends in Cognitive Sciences*, vol. 3, no. 1, pp. 22–31, 1999.

[42] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.

[43] K. Huang and S. Aviyente, "Wavelet feature selection for image classification," *Image Processing, IEEE Transactions on*, vol. 17, no. 9, pp. 1709–1720, 2008.

[44] C. Shang and D. Barnes, "Fuzzy-rough feature selection aided support vector machines for mars image classification," *Computer Vision and Image Understanding*, vol. 117, no. 3, pp. 202–213, 2013.

[45] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, Massachusetts, 1996, vol. 1.

[46] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[47] S. Nissen, "Implementation of a fast artificial neural network library (fann)," *Report, Department of Computer Science University of Copenhagen (DIKU)*, vol. 31, 2003.

[48] M. A. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[49] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[50] R. J. Douglas and K. A. Martin, "Neuronal circuits of the neocortex," *Annu. Rev. Neurosci.*, vol. 27, pp. 419–451, 2004.

[51] D. J. Felleman and D. C. Van Essen, "Distributed hierarchical processing in the primate cerebral cortex," *Cerebral Cortex*, vol. 1, no. 1, pp. 1–47, 1991.

[52] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[53] T. S. Lee, D. Mumford, R. Romero, and V. A. Lamme, "The role of the primary visual cortex in higher level vision," *Vision Research*, vol. 38, no. 15, pp. 2429–2454, 1998.

[54] D. George, "How the brain might work: A hierarchical and temporal model for learning and recognition," Ph.D. dissertation, Stanford University, 2008.

[55] W. A. Phillips and W. Singer, "In search of common foundations for cortical computation," *Behavioral and Brain Sciences*, vol. 20, no. 04, pp. 657–683, 1997.

[56] T. S. Lee and D. Mumford, "Hierarchical bayesian inference in the visual cortex," *JOSA A*, vol. 20, no. 7, pp. 1434–1448, 2003.

[57] T. Dean, G. Carroll, and R. Washington, "On the prospects for building a working model of the visual cortex," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 22, no. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007, p. 1597.

[58] J.-C. Chappelier and A. Grumbach, "Rst: a connectionist architecture to deal with spatiotemporal relationships," *Neural Computation*, vol. 10, no. 4, pp. 883–902, 1998.

[59] F. H. Rauscher, G. L. Shaw, and K. N. Ky, "Listening to mozart enhances spatial-temporal reasoning: towards a neurophysiological basis," *Neuroscience Letters*, vol. 185, no. 1, pp. 44–47, 1995.

[60] H. Wang and L. Cai, "On adaptive spatial-temporal processing for airborne surveillance radar systems," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 30, no. 3, pp. 660–670, 1994.

[61] J. C. Niebles, H. Wang, and L. Fei-Fei, "Unsupervised learning of human action categories using spatial-temporal words," *International Journal of Computer Vision*, vol. 79, no. 3, pp. 299–318, 2008.

[62] J. W. Miller and P. H. Lommel, "Biomimetic sensory abstraction using hierarchical quilted self-organizing maps," in *Optics East 2006*. International Society for Optics and Photonics, 2006, pp. 63 840A–63 840A.

[63] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust object recognition with cortex-like mechanisms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 3, pp. 411–426, 2007.

[64] F. Huang and Y. LeCun, "Large-scale learning with svm and convolutional netw for generic object recognition," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[65] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 3, pp. 328–339, 1989.

[66] Y.-N. Chen, C.-C. Han, C.-T. Wang, B.-S. Jeng, and K.-C. Fan, "The application of a convolution neural network on face and license plate detection," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3. IEEE, 2006, pp. 552–555.

[67] B. Kwolek, "Face detection using convolutional neural networks and gabor filters," in *Artificial Neural Networks: Biological Inspirations–ICANN 2005*. Springer, 2005, pp. 551–556.

[68] F. H. C. Tivive and A. Bouzerdoum, "A new class of convolutional neural networks (siconnets) and their application of face detection," in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, vol. 3. IEEE, 2003, pp. 2157–2162.

[69] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis." in *ICDAR*, vol. 3, 2003, pp. 958–962.

[70] S. Sukittanon, A. C. Surendran, J. C. Platt, and C. J. Burges, "Convolutional networks for speech detection." in *Interspeech*. Citeseer, 2004.

[71] H. Mobahi, R. Collobert, and J. Weston, "Deep learning from temporal coherence in video," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 737–744.

[72] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th International Conference on Machine Learning*. ACM, 2007, pp. 791–798.

[73] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[74] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[75] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proceedings of the 24th International Conference on Machine Learning*. ACM, 2007, pp. 473–480.

[76] I. Sutskever and G. E. Hinton, "Learning multilevel distributed representations for high-dimensional sequences," in *International Conference on Artificial Intelligence and Statistics*, 2007, pp. 548–555.

[77] A. J. Lockett and R. Miikkulainen, "Temporal convolution machines for sequence learning," *To Appear*, pp. 737–747, 2009.

[78] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in Neural Information Processing Systems*, 2009, pp. 1096–1104.

[79] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*. ACM, 2008, pp. 1096–1103.

[80] D. Mart, "A computational investigation into the human representation and processing of visual information," 1982.

[81] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[82] K. Fukushima, "Neocognitron for handwritten digit recognition," *Neurocomputing*, vol. 51, pp. 161–180, 2003.

[83] ——, "Restoring partly occluded patterns: a neural network model," *Neural Networks*, vol. 18, no. 1, pp. 33–43, 2005.

[84] ——, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[85] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.

[86] J. Hawkins and S. Blakeslee, "On intelligence (adapted ed.)," 2004.

[87] S. Behnke, *Hierarchical neural networks for image interpretation*. Springer Science & Business Media, 2003, vol. 2766.

[88] M. Osadchy, Y. L. Cun, and M. L. Miller, "Synergistic face detection and pose estimation with energy-based models," *The Journal of Machine Learning Research*, vol. 8, pp. 1197–1215, 2007.

[89] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[90] J. M. Mendel, "Fuzzy logic systems for engineering: a tutorial," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 345–377, 1995.

[91] S. Mitra, S. K. Pal, and P. Mitra, "Data mining in soft computing framework: a survey," *Neural Networks, IEEE Transactions on*, vol. 13, no. 1, pp. 3–14, 2002.

[92] W.-H. Hsiao, S.-M. Chen, and C.-H. Lee, "A new interpolative reasoning method in sparse rule-based systems," *Fuzzy Sets and Systems*, vol. 93, no. 1, pp. 17–22, 1998.

[93] I. Turksen and Y. Tian, "How to select combination operators for fuzzy expert systems using cri," 1992.

[94] M. Sugeno and G. Kang, "Structure identification of fuzzy model," *Fuzzy Sets and Systems*, vol. 28, no. 1, pp. 15–33, 1988.

[95] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 1, pp. 116–132, 1985.

[96] A. Hamam and N. D. Georganas, "A comparison of mamdani and sugeno fuzzy inference systems for evaluating the quality of experience of hapto-audio-visual applications," in *Haptic Audio visual Environments and Games, 2008. HAVE 2008. IEEE International Workshop on*. IEEE, 2008, pp. 87–92.

[97] A. Fernández and F. Herrera, "Linguistic fuzzy rules in data mining: follow-up mamdani fuzzy modeling principle," in *Combining Experimentation and Theory*. Springer, 2012, pp. 103–122.

[98] O. Castillo and P. Melin, *Soft computing and fractal theory for intelligent manufacturing*. Springer Science & Business Media, 2003, vol. 117.

[99] N. N. Karnik and J. M. Mendel, "Centroid of a type-2 fuzzy set," *Information Sciences*, vol. 132, no. 1, pp. 195–220, 2001.

[100] F. Liu, "An efficient centroid type-reduction strategy for general type-2 fuzzy logic system," *Information Sciences*, vol. 178, no. 9, pp. 2224–2236, 2008.

[101] S. Coupland and R. John, "Geometric type-1 and type-2 fuzzy logic systems," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 1, pp. 3–15, 2007.

[102] D. Wu and J. M. Mendel, "A comparative study of ranking methods, similarity measures and uncertainty measures for interval type-2 fuzzy sets," *Information Sciences*, vol. 179, no. 8, pp. 1169–1192, 2009.

[103] Z. C. Johanyák, D. Tikk, S. Kovács, and K. W. Wong, "Fuzzy rule interpolation matlab toolbox-fri toolbox," in *Fuzzy Systems, 2006 IEEE International Conference on*. IEEE, 2006, pp. 351–357.

[104] E. H. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," *Computers, IEEE Transactions on*, vol. 100, no. 12, pp. 1182–1191, 1977.

[105] Y. Shi and M. Mizumoto, "Some considerations on koczy's fuzzy interpolative reasoning method," in *Fuzzy Systems, 1995. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., Proceedings of 1995 IEEE Int*, vol. 4. IEEE, 1995, pp. 2117–2122.

[106] L. T. Koczy and K. Hirota, "Size reduction by interpolation in fuzzy rule bases," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 27, no. 1, pp. 14–25, 1997.

[107] M. Mizumoto and H.-J. Zimmermann, "Comparison of fuzzy reasoning methods," *Fuzzy Sets and Systems*, vol. 8, no. 3, pp. 253–283, 1982.

[108] Z. C. Johanyák and S. Kovács, "A brief survey and comparison on various interpolation based fuzzy reasoning methods," *Acta Polytechnica Hungarica*, vol. 3, no. 1, pp. 91–105, 2006.

[109] ——, "Survey on various interpolation based fuzzy reasoning methods," *Production Systems and Information Engineering*, vol. 3, no. 1, pp. 39–56, 2006.

[110] I. Perfilieva, D. Dubois, H. Prade, F. Esteva, L. Godo, and P. Hoďáková, "Interpolation of fuzzy data: Analytical approach and overview," *Fuzzy Sets and Systems*, vol. 192, pp. 134–158, 2012.

[111] J. A. Robinson, "A machine-oriented logic based on the resolution principle," *Journal of the ACM (JACM)*, vol. 12, no. 1, pp. 23–41, 1965.

[112] L. A. Zadeh, "Quantitative fuzzy semantics," *Information Sciences*, vol. 3, no. 2, pp. 159–176, 1971.

[113] D. Dubois and H. Prade, "Gradual inference rules in approximate reasoning," *Information Sciences*, vol. 61, no. 1, pp. 103–122, 1992.

[114] F. Klawonn and J. L. Castro Peña, "Similarity in fuzzy reasoning," *Mathware & Soft Computing. 1995 Vol. 2 Núm. 3*, 1995.

[115] S. Jenei, "Interpolation and extrapolation of fuzzy quantities revisited–an axiomatic approach," *Soft Computing*, vol. 5, no. 3, pp. 179–193, 2001.

[116] S. Jenei, E.-P. Klement, and R. Konzel, "Interpolation and extrapolation of fuzzy quantities–the multiple-dimensional case," *Soft Computing*, vol. 6, no. 3-4, pp. 258–270, 2002.

[117] S. Kovács, "Extending the fuzzy rule interpolation" five" by fuzzy observation," in *Computational Intelligence, Theory and Applications*. Springer, 2006, pp. 485–497.

[118] Z. Q. Wu, M. Masaharu, and Y. Shi, "An improvement to kóczy and hirota's interpolative reasoning in sparse fuzzy rule bases," *International Journal of Approximate Reasoning*, vol. 15, no. 3, pp. 185–201, 1996.

[119] S.-M. Chen and Y.-K. Ko, "Fuzzy interpolative reasoning for sparse fuzzy rule-based systems based on-cuts and transformations techniques," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1626–1648, 2008.

[120] I. Arel, D. Rose, and T. Karnowski, "A deep learning architecture comprising homogeneous cortical circuits for scalable spatiotemporal pattern inference," in *NIPS 2009 Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[121] S. Young, I. Arel, T. P. Karnowski, and D. Rose, "A fast and stable incremental clustering algorithm," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*. IEEE, 2010, pp. 204–209.

[122] V. Vapnik, "Statistical learning theory. 1998," 1998.

[123] J. C. Platt, *12 Fast Training of Support Vector Machines using Sequential Minimal Optimization*. MIT Press, 1999.

[124] ——, "Sequential minimal optimization: A fast algorithm for training support vector machines," in *Advances in Kernel Methods-Support Vector Learning*, 1998.

[125] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[126] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[127] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Applied Statistics*, pp. 100–108, 1979.

[128] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[129] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 881–892, 2002.

[130] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2, pp. 191–203, 1984.

[131] J. Platt *et al.*, "Fast training of support vector machines using sequential minimal optimization," *Advances in kernel methods?support vector learning*, vol. 3, 1999.

[132] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[133] M. Jiang, Y. Ding, B. Goertzel, Z. Huang, C. Zhou, and F. Chao, "Improving machine vision via incorporating expectation-maximization into deep spatio-temporal learning." in *IJCNN*, 2014, pp. 1804–1811.

[134] J. Cendrowska, "Prism: An algorithm for inducing modular rules," *International Journal of Man-Machine Studies*, vol. 27, no. 4, pp. 349–370, 1987.

[135] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[136] Y. Yuan and M. J. Shaw, "Induction of fuzzy decision trees," *Fuzzy Sets and Systems*, vol. 69, no. 2, pp. 125–139, 1995.

[137] M. Guetova, S. Hölldobler, and H.-P. Störr, "Incremental fuzzy decision trees," in *KI 2002: Advances in Artificial Intelligence*. Springer, 2002, pp. 67–81.

[138] S.-J. Lee, X.-J. Zeng, and H.-S. Wang, "Generating automatic fuzzy system from relational database system for estimating null values," *Cybernetics and Systems: An International Journal*, vol. 40, no. 6, pp. 528–548, 2009.

[139] D. Wang, X.-J. Zeng, J. Keane *et al.*, "An evolving-construction scheme for fuzzy systems," *Fuzzy Systems, IEEE Transactions on*, vol. 18, no. 4, pp. 755–770, 2010.

[140] H. Zhang and Z. Bien, "Adaptive fuzzy control of mimo nonlinear systems," *Fuzzy Sets and Systems*, vol. 115, no. 2, pp. 191–204, 2000.

[141] S. Wu, M. J. Er, and Y. Gao, "A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks," *Fuzzy Systems, IEEE Transactions on*, vol. 9, no. 4, pp. 578–594, 2001.

[142] P. P. Angelov, "An evolutionary approach to fuzzy rule-based model synthesis using indices for rules," *Fuzzy Sets and Systems*, vol. 137, no. 3, pp. 325–338, 2003.

[143] P. P. Angelov and R. A. Buswell, "Automatic generation of fuzzy rule-based models from data by genetic algorithms," *Information Sciences*, vol. 150, no. 1, pp. 17–31, 2003.

[144] H.-J. Zimmermann, *Fuzzy Set Theory and its Applications*. Springer Science & Business Media, 2011.

[145] T. J. Ross, *Fuzzy Logic with Engineering Applications*. John Wiley & Sons, 2009.

[146] T. Terano, K. Asai, and M. Sugeno, *Applied Fuzzy Systems*. Academic Press, 2014.

[147] S.-M. Chen and S.-W. Chen, "Fuzzy forecasting based on two-factors second-order fuzzy-trend logical relationship groups and the probabilities of trends of fuzzy logical relationships," *Cybernetics, IEEE Transactions on*, vol. 45, no. 3, pp. 391–403, 2015.

[148] K. A. Rasmani and Q. Shen, "Data-driven fuzzy rule generation and its application for student academic performance evaluation," *Applied Intelligence*, vol. 25, no. 3, pp. 305–319, 2006.

[149] R. Jensen and Q. Shen, "New approaches to fuzzy-rough feature selection," *Fuzzy Systems, IEEE Transactions on*, vol. 17, no. 4, pp. 824–838, 2009.

[150] M. Dash and H. Liu, "Consistency-based search in feature selection," *Artificial Intelligence*, vol. 151, no. 1, pp. 155–176, 2003.

[151] D. Newman, S. Hettich, C. Blake, and C. Merz, "Uci repository of machine learning databases. university of california, department of information and computer science, irvine, ca (1998)," *MLRepository. html*, 2010.

[152] Y. Bengio and Y. Grandvalet, "Bias in estimating the variance of k-fold cross-validation," in *Statistical Modeling and Analysis for Complex Data Problems*. Springer, 2005, pp. 75–95.

[153] X. Wang, J. Yang, X. Teng, W. Xia, and R. Jensen, "Feature selection based on rough sets and particle swarm optimization," *Pattern Recognition Letters*, vol. 28, no. 4, pp. 459–471, 2007.

[154] R. Leardi, R. Boggia, and M. Terrile, "Genetic algorithms as a strategy for feature selection," *Journal of Chemometrics*, vol. 6, no. 5, pp. 267–281, 1992.

[155] R. Diao and Q. Shen, "Feature selection with harmony search," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 6, pp. 1509–1523, 2012.

[156] S. Buseman, J. Mouchawar, N. Calonge, and T. Byers, "Mammography screening matters for young women with breast carcinoma," *Cancer*, vol. 97, no. 2, pp. 352–358, 2003.

[157] J. Roehrig, T. Doi, A. Hasegawa, B. Hunt, J. Marshall, H. Romsdahl, A. Schneider, R. Sharbaugh, and W. Zhang, "Clinical results with r2 imagechecker system," in *Digital Mammography*. Springer, 1998, pp. 395–400.

[158] J. N. Wolfe, "Risk for breast cancer development determined by mammographic parenchymal pattern," *Cancer*, vol. 37, no. 5, pp. 2486–2492, 1976.

[159] N. Boyd, J. Byng, R. Jong, E. Fishell, L. Little, A. Miller, G. Lockwood, D. Tritchler, and M. J. Yaffe, "Quantitative classification of mammographic densities and breast cancer risk: results from the canadian national breast screening study," *Journal of the National Cancer Institute*, vol. 87, no. 9, pp. 670–675, 1995.

[160] V. A. McCormack and I. dos Santos Silva, "Breast density and parenchymal patterns as markers of breast cancer risk: a meta-analysis," *Cancer Epidemiology Biomarkers & Prevention*, vol. 15, no. 6, pp. 1159–1169, 2006.

[161] F. Aghdasi, R. Ward, J. Morgan-Parkes, and B. Palcic, "Feature selection for classification of mammographic microcalcification clusters," in *Engineering in Medicine and Biology Society, 1993. Proceedings of the 15th Annual International Conference of the IEEE*. IEEE, 1993, pp. 58–59.

[162] R. Highnam and M. Brady, "Mammographic image analysis kluwer academic," 1999.

[163] A. Hassanien, "Fuzzy rough sets hybrid scheme for breast cancer detection," *Image and Vision Computing*, vol. 25, no. 2, pp. 172–183, 2007.

[164] M. Roffilli, "Advanced machine learning techniques for digital mammography," *Departament of Computer Science. University of Bologna, Bologna (Italy), Tech. Rep*, 2006.

[165] C. J. D'orsi, A. C. of Radiology, A. C. of Radiology, B.-R. Committee *et al.*, *Illustrated Breast Imaging Reporting and Data System:(illustrated BI-RADS)*. American College of Radiology, 1998.

[166] A. Oliver, J. Freixenet, R. Marti, J. Pont, E. Perez, E. R. Denton, and R. Zwiggelaar, "A novel breast tissue density classification methodology," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 12, no. 1, pp. 55–65, 2008.

[167] N. Kwak and C.-H. Choi, "Input feature selection for classification problems," *Neural Networks, IEEE Transactions on*, vol. 13, no. 1, pp. 143–159, 2002.

[168] J. Suckling, J. Parker, D. Dance, S. Astley, I. Hutt, C. Boggis, I. Ricketts, E. Stamatakis, N. Cerneaz, S. Kok *et al.*, "The mammographic image analysis society digital mammogram database," in *Exerpta Medica. International Congress Series*, vol. 1069, 1994, pp. 375–378.

[169] J. R. Quinlan, *C45*. Morgan Kaufmann, 1992.

[170] W. Gu, C. Xiang, Y. Venkatesh, D. Huang, and H. Lin, "Facial expression recognition using radial encoding of local gabor features and classifier synthesis," *Pattern Recognition*, vol. 45, no. 1, pp. 80–91, 2012.

[171] A. Fernández, Á. Gómez, F. Lecumberry, Á. Pardo, and I. Ramírez, "Pattern recognition in latin america in the ?big data? era," *Pattern Recognition*, vol. 48, no. 4, pp. 1185–1196, 2015.

[172] L. Liu, L. Shao, and P. Rockett, "Boosted key-frame selection and correlated pyramidal motion-feature representation for human action recognition," *Pattern Recognition*, vol. 46, no. 7, pp. 1810–1818, 2013.

[173] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, and B. K. Panigrahi, "Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 1, pp. 89–106, 2011.

[174] P. Baranyi, L. T. Kóczy, and T. T. D. Gedeon, "A generalized concept for fuzzy rule interpolation," *Fuzzy Systems, IEEE Transactions on*, vol. 12, no. 6, pp. 820–837, 2004.

[175] S.-M. Chen and W.-C. Hsin, "Weighted fuzzy interpolative reasoning based on the slopes of fuzzy sets and particle swarm optimization techniques," *Cybernetics, IEEE Transactions on*, vol. 45, no. 7, pp. 1250–1261, 2015.

[176] S. Garcia-Jimenez, H. Bustince, E. Hullermeier, R. Mesiar, N. R. Pal, and A. Pradera, "Overlap indices: Construction of and application to interpolative fuzzy systems," *Fuzzy Systems, IEEE Transactions on*, vol. 23, no. 4, pp. 1259–1273, 2015.

[177] S. Jin, R. Diao, C. Quek, and Q. Shen, "Backward fuzzy rule interpolation," *Fuzzy Systems, IEEE Transactions on*, vol. 22, no. 6, pp. 1682–1698, 2014.