ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

DESIGN AND DEPLOYMENT OF DEEP LEARNING BASED FUZZY LOGIC SYSTEMS

Ph.D. THESIS

Aykut BEKE

Department of Control and Automation Engineering

Control and Automation Engineering Programme

AUGUST 2023

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

DESIGN AND DEPLOYMENT OF DEEP LEARNING BASED FUZZY LOGIC SYSTEMS

Ph.D. THESIS

Aykut BEKE
(504182102)

Department of Control and Automation Engineering

Control and Automation Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Tufan KUMBASAR

AUGUST 2023

ISTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DERİN ÖĞRENME TABANLI BULANIK SİSTEMLERİN GELİŞTİRİLMESİ VE UYGULANMASI

DOKTORA TEZİ

Aykut Beke
(504182102)

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Tufan KUMBASAR

AĞUSTOS 2023

Aykut BEKE, a Ph.D. student of İTU Graduate School of Science Engineering and Technology student ID 504182102, successfully defended the thesis entitled "DESIGN AND DEPLOYMENT OF DEEP LEARNING BASED FUZZY LOGIC SYSTEMS", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Tufan KUMBASAR** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. İbrahim EKSİN** Istanbul Technical University	.....
	Assoc. Prof. Dr. İlker ÜSTOĞLU Istanbul Technical University	.....
	Asst. Prof. Dr. Gürkan SOYKAN Bahcesehir University	.....
	Asst. Prof. Dr. Onur VAROL Sabanci University	.....

Date of Submission : 1 June 2023
Date of Defense : 29 August 2023

To my family,

FOREWORD

This thesis is an outcome of the works that I completed in Artificial Intelligence and Intelligent System (AI^2S) laboratory. I've spent wonderful years in the AI^2S laboratory and was lucky to meet amazing people. Firstly, I would like to thank all the members of the AI^2S laboratory for their support. I also want to thank my beloved wife Merve BEKE and the dearest friend Fethi CANDAN for their continuous help and confidence in me.

Finally, I would like to give my appreciation and gratitude to my supervisor Assoc. Prof. Dr. Tufan KUMBASAR for his help, encouragements and guidance without his effort and help this thesis would not been completed.

August 2023

Aykut BEKE
(Control System Engineer)

TABLE OF CONTENTS

Page

ABBREVIATIONS

ABC : Artificial Bee Colony
ACO : Ant Colony Optimization
APL : Adaptive Piecewise Linear
BN : Batch Normalization
CNN : Convolutional Neural Network
CSCM : Center of Sets Calculation Method
DL : Deep Learning
DNN : Deep NN
ELM : Extreme Learning Machine
ELU : Exponential Linear Unit
FC : Fully Connected
FLS : Fuzzy Logic System
FOU : Footprint of Uncertainty
FRUs : Fuzzy Rectifying Units
FS : Fuzzy Set
GA : Genetic Algorithms
GS : Greedy Sampling
GT2-FLS : General Type-2 Fuzzy Logic System
IQR : Inter Quartile Range
IT2 : Interval Type-2
IT2-FAL : IT2 Fuzzy Activation Layer
IV : Interval Valued
IVL : IV Linear
KM : Karnik-Mendel
LMF : Lower MF
MAE : Mean Absolute Error
MBS : Mini Batch Size
MF : Membership Function
MSE : Mean Square Error
NN : Neural Network
NT : Nie-Tan
PI : Prediction Interval
PICP : PI Coverage Probability
PINAW : PI Normalized Averaged Width
PReLU : Parametric ReLU
PSO : Particle Swarm Optimization
QR : Quantile Regression
ReLU : Rectified Linear Unit
RMSE : Root MSE
RL : Reinforcement Learning
RWN : Random Weight NN
S-DNN : Stacked DNN
SM : Simple Method
SMF : Secondary Membership Function

SIT2 **:** Single input IT2
SVM **:** Support Vector Machine
SVR **:** Support Vector Regression
TR **:** Type Reduced
TSK **:** Takagi Sugeno Kang
T1 **:** Type-1
T2 **:** Type-2
UMF **:** Upper MF

SYMBOLS

$f(x)$: Activation function representation
a : Negative slope parameter
K_f : Input scaling factor
$\tilde{y}_c$: Output of IT2-FAL
$\varphi(.)$: Output of SIT2-FRU
ρ : Momentum
γ : Learning rate
L : Loss function
λ : Regularization hyperparameter
ℓ_τ : Tilted loss
$\overline{\tau}$: Upper quantile level
$\underline{\tau}$: Lower quantile level
q^τ : Quantile function
ϵ_n : Quantile residue
y_{IT2} : IT2-FLS output
y_{GT2} : GT2-FLS output
$c_{p,m}$: Center parameter of gauss MF
$h_{p,m}$: Height parameter of LMF
$\sigma_{p,m}$: Sigma parameter of gauss MF
$a_{p,m}$: Consequent MF parameter a
$b_{p,m}$: Consequent MF parameter b
R_p : p^{th} rule of the rule base.
$\tilde{A}$: Type-2 fuzzy set.
J_x : Primary membership function of *x*.
μ_A : Membership function of *A*
$\mu_{\tilde{A}}$: Secondary membership.
$\overline{\mu}_{\tilde{A}}$: Upper membership function.
$\underline{\mu}_{\tilde{A}}$: Lower membership function.
$\underline{y}_p$: Lower end boundary.
$\overline{y}_p$: Upper end boundary.
$\tilde{Y}$: Interval output of T2-FLS.
$\underline{f}_p$: Lower firing.
$\overline{f}_p$: Upper firing.
$\tilde{F}$: Interval firing of T2-FLS.
θ : Learnable parameter set of IT2-FLS
θ_A : Antecedent learnable parameter set of IT2-FLS
θ_C : Consequent learnable parameter set of IT2-FLS
θ_{GT2} : Learnable parameter set of GT2-FLS
θ_{GT2-A} : Antecedent learnable parameter set of GT2-FLS
θ_{GT2-C} : Consequent learnable parameter set of GT2-FLS
H-C : Antecedent H-type FS Consequent C-type MF
H-L : Antecedent H-type FS Consequent L-type MF
H-IV : Antecedent H-type FS Consequent IV-type MF
H-IVL : Antecedent H-type FS Consequent IVL-type MF

S-C : Antecedent S-type FS Consequent C-type MF
S-L : Antecedent S-type FS Consequent L-type MF
S-IV : Antecedent S-type FS Consequent IV-type MF
S-IVL : Antecedent S-type FS Consequent IVL-type MF
HS-C : Antecedent HS-type FS Consequent C-type MF
HS-L : Antecedent HS-type FS Consequent L-type MF
HS-IV : Antecedent HS-type FS Consequent IV-type MF
HS-IVL : Antecedent HS-type FS Consequent IVL-type MF
$\boldsymbol{\varphi_d}$: Expected coverage
$\boldsymbol{N_{bs}}$: Number of minibatch size
$\boldsymbol{lr}$: Learning rate
$\boldsymbol{L_1}$: Mean absolute error norm
$\boldsymbol{L_2}$: Mean square error norm
$\boldsymbol{L_{log-cosh}}$: Logarithmic hyperbolic cosine norm
$\mathbf{S_p}$: Consequents
$\boldsymbol{\alpha_k}$: α plane of the GT2-FLS
$\boldsymbol{\delta_{p,m}}$: SMF parameters of GT2-FLS

LIST OF TABLES

Page

LIST OF FIGURES

Page

DESIGN AND DEPLOYMENT OF DEEP LEARNING BASED FUZZY LOGIC SYSTEMS

SUMMARY

In the past decade, significant progress has been made in the field of Deep Learning (DL), driven by innovative learning methods, novel layer structures, and the use of graphics cards for enhanced processing power. This progress has led to the training of neural network models with numerous hidden layers and neurons, resulting in breakthroughs in various domains such as semantic segmentation, object detection, and classification. Deep Neural Networks (DNNs) have proven to be highly effective in machine learning and artificial intelligence applications. DNNs offer advantages over traditional machine learning techniques, including the ability to learn features at multiple layers, which allows them to capture complex features of input data. Through forward pass and backpropagation, DNNs extract meaningful features and outperform other methods in many tasks. As a result, DNNs have gained popularity and are widely used in commercial and industrial applications, contributing to advancements in machine learning.

Fuzzy Logic Systems (FLSs) have been employed to various fields and applications over the last years. FLSs use linguistic Fuzzy Sets (FSs) and fuzzy rules, enabling the modeling of human-like reasoning and decision-making processes. This has led to advancements in the development of intelligent control systems capable of effectively handling nonlinear and uncertain dynamics. Besides, FLSs have been applied in image processing, leveraging the FSs to represent uncertain data. FLSs provide robust image analysis, pattern recognition, and image understanding, contributing to advancements in computer vision and image processing applications. Overall, FLSs have been extensively utilized in modeling various systems and phenomena. Their ability to handle uncertainty provides a flexible and interpretable modeling approach, capturing complex relationships and uncertainties in real-world systems. Conventional FLSs, known as Type-1 FLSs (T1-FLSs), have limitations in representing uncertainty. To address this, Type-2 Fuzzy Sets (T2-FSs) have been introduced as an alternative, offering a more flexible representation.

T2-FSs can better handle nonlinear and uncertain systems, and T2-Fuzzy Logic Systems (T2-FLSs) have the potential to handle complex problems. However, learning T2-FLSs presents challenges due to their design complexity and the need to learn the parameters associated with fuzzy sets. Different approaches have been proposed, including adapting pre-trained T1-FLSs to T2-FLSs and employing evolutionary algorithms or Neural Network (NN) approaches to optimize the parameters of Interval T2-FLSs (IT2-FLSs). These approaches aim to simplify the design complexity and improve the performance of T2-FLSs. Despite advancements, integrating neural networks and evolutionary algorithms with T2-FLSs faces challenges when applied to extensive datasets. The curse of dimensionality and the increasing number of parameters in T2-FLSs brings some difficulties that is not possible to solve with the

current approaches. Recent research has focused on combining FLSs and deep neural networks to overcome these challenges, leading to the development of hybrid models that leverage the strengths of both generalization capabilities of the DNNs and the power of the mini-batch sampled optimization algorithms.

In this thesis study, a novel approach is proposed to learn the parameters of T2-FLSs using deep learning-based parameter learning methods. The proposed approach aims to handle extensive datasets and construct models with both a good prediction accuracy and the ability to handle the uncertainties. In the scope of this thesis, specifically, three studies are conducted: the first study (i) is titled with "Learning with Type-2 Fuzzy Activation Functions to Enhance the Performance of Deep Neural Networks", in the second study (ii), we propose a framework which is titled with "More Than Accuracy: A Composite Learning Framework for Interval Type-2 Fuzzy Logic Systems" and in the (iii) last study, we propose reliable uncertainty quantification for GT2-FLSs named as "Towards Reliable Uncertainty Quantification and High Precision with General Type-2 Fuzzy Systems".

In the first study (i), we introduce a new method called IT2 Fuzzy Activation Layer (IT2-FAL) that aims to enhance the learning performance of DNNs. The IT2-FAL consists of Single Input IT2 (SIT2) Fuzzy Rectifying Units (FRUs) which used as activation units within the DNN structure to improve learning capabilities. We construct a closed-form representation of the SIT2-FRU structure, and an analysis is conducted to understand how the parameters of this structure influence the generation of input-output mappings. The research findings demonstrate that these mappings can be regarded either as hyperparameters to be set or as parameters to be learned. We provide a learning algorithm to these hyperparameters using DL based frameworks.

To evaluate the effectiveness of the proposed IT2-FAL, a comparison is made against existing activation units like ReLU, PReLU, and ELU. The novel SIT2-FRU not only addresses the vanishing gradient problem but also exhibits a fast convergence rate. It achieves this by pushing the mean activation close to zero through the processing of inputs defined in the negative quadrant. This property of SIT2-FRU enables DNNs to exhibit improved learning behavior. The experiments conducted using the selected benchmark datasets show the efficiency and superiority of the IT2-FAL approach. By incorporating the IT2-FAL and its activation units (SIT2-FRU components), DNNs can enhance their learning capabilities and benefit from a more robust and flexible network structure. The proposed approach has the potential to improve the performance of DNNs as the experimental results revealed and it also gives opportunity to enhance the learning capabilities of DNNs.

The second study (ii) introduces a novel composite learning approach that utilizes type-reduced sets of Interval Type-2 Fuzzy Logic Systems (IT2-FLSs) to capture uncertainty and establish Prediction Intervals (PIs). Unlike mainstream training approaches that primarily focus on accuracy, the objective of this new approach is to not only achieve high prediction accuracy but also effectively address and capture uncertainty by exploiting the type-reduced sets of IT2-FLSs. In order to achieve such a goal, we identify three main challenges in this context: (1) the capability to handle uncertainty, (2) the construction of a composite loss function, and (3) the development of a learning algorithm that addresses the training complexity while considering the definitions of IT2-FLSs.

In (1), to address these challenges, the proposed approach exploits the type-reduced set of IT2-FLSs by combining quantile regression and DL parameter learning methods

with IT2-FLSs. The ability of IT2-FLSs to process uncertainty depends on the methods employed for calculating the center-of-sets, while their representation capability is determined by the structure of their antecedent and consequent membership functions. In the scope of thesis, we introduce various parametric IT2-FLSs and defines the learnable parameters for all IT2-FLSs, along with their constraints that need to be satisfied during the training process. In (2), the construction of the loss function is defined which involves construction of a multi-objective loss that is subsequently converted into a constrained composite loss. This composite loss comprises the log-cosh loss component, which aims to optimize accuracy, and a tilted loss component that focuses on the representation of uncertainty. Notably, the tilted loss explicitly utilizes the type-reduced set. In (3), a DL approach is presented for training IT2-FLSs using unconstrained optimizers. The study also introduces parameterization techniques to convert the constrained optimization problem of IT2-FLSs into an unconstrained one without violating the definitions of fuzzy sets.

In order to evaluate the effectiveness of the proposed approach, comprehensive comparative results are provided. In the thesis, we provide a hyperparameter sensitivity analysis and inter/intra-model comparisons conducted on various benchmark datasets. These evaluations shed light on the performance and robustness of the proposed novel approach in handling uncertainty and achieving high prediction accuracy for regression problems.

In the third study (iii), we present a new learning approach for α-plane based General Type-2 Fuzzy Logic Systems (GT2-FLSs) to improve pointwise prediction accuracy and generate reliable Prediction Intervals (PIs). The approach focuses on exploiting the shape and size of the Secondary Membership Functions (SMFs) through a novel composite loss function.

The novel composite loss function consists of two main components: an uncertainty quantification-focused loss and an accuracy-focused term. Within the uncertainty-focused loss, only the type-reduced set of IT2-FLS associated with the $\alpha_0 = 0$ plane, known as the FOU, is explicitly utilized. This allows the SMF size parameters of the GT2-FLS to quantify uncertainty and learn PIs. For the accuracy-focused part, two alternative loss terms are provided. In one approach, the aggregated output of the GT2-FLSs is used directly, while in the other approach, only the output associated with the $\alpha_K = 1$ level is utilized. In both cases, the SMF shape parameters of the GT2-FLS are enforced to enable pointwise prediction with high precision. Thus, different roles are assigned to the IT2-FLS associated with α-planes within the proposed loss function. Since the output of the $\alpha_0 = 0$ plane does not contribute to the output calculation of the GT2-FLS, a partially independent learning of the GT2-FSs becomes possible, allowing for capturing uncertainty while maintaining high accuracy.

We present a DL based parameter learning approach for GT2-FLSs to facilitate efficient learning to be able to handle the complex parameter learning problem of the GT2-FLSs and also in the presence of high-dimensional and complex data. This is achieved by defining an unconstrained learning problem. We also proposed novel parameterization tricks such that the definitions of GT2-FSs are not violated. We also provide statistical comparative analyses using benchmark datasets in order to demonstrate the superiority of the proposed learning approach. The results of these analyses show the potential of learning GT2-FLS with the proposed DL based approach as a promising solution for reliable uncertainty quantification with high precision in real-world applications.

DERİN ÖĞRENME TABANLI BULANIK SİSTEMLERİN GELİŞTİRİLMESİ VE UYGULANMASI

ÖZET

Son yıllarda, yenilikçi öğrenme yöntemleri, yeni katman yapıları ve gelişmiş işlem gücü için grafik kartlarının kullanılmasıyla Derin Öğrenme (DÖ) alanında önemli ilerlemeler kaydedildi. Bu ilerleme, çok sayıda gizli katman ve nöron içeren sinir ağı modellerinin eğitimine yol açmış, semantik bölümleme, nesne algılama ve sınıflandırma gibi çeşitli alanlarda çığır açmıştır. Derin Yapay Sinir Ağlarının (DYSA), makine öğrenimi ve yapay zeka uygulamalarında oldukça etkili olduğu kanıtlanmıştır. DYSA'lar, girdi verilerinin karmaşık özelliklerini yakalamalarına olanak tanıyan birden çok katmandaki özellikleri öğrenme yeteneği de dahil olmak üzere geleneksel makine öğrenimi tekniklerine göre avantajlar sunar. DYSA'lar, ileri geçiş ve geri yayılım yoluyla anlamlı özellikler çıkarır ve birçok görevde diğer yöntemlerden daha iyi performans gösterir. Sonuç olarak, DYSA'lar popülerlik kazandı ve ticari ve endüstriyel uygulamalarda yaygın olarak kullanıldı ve makine öğrenimindeki ilerlemelere katkıda bulundu.

Bulanık Mantık Sistemleri (BMS'ler), son yıllarda çeşitli alanlarda ve uygulamalarda kullanılmıştır. BMS'ler dilbilimsel Bulanık Kümeler (BK'ler) ve bulanık kurallar kullanarak insan benzeri muhakeme ve karar verme süreçlerinin modellenmesini sağlar. Bu, doğrusal olmayan ve belirsiz dinamikleri etkili bir şekilde idare edebilen akıllı kontrol sistemlerinin geliştirilmesinde ilerlemelere yol açmıştır. Ayrıca, belirsiz verileri temsil etmek için BK'lerden yararlanarak görüntü işlemede BMS'ler uygulanmıştır. BMS'ler, güçlü görüntü analizi, örüntü tanıma ve görüntü anlama sağlayarak bilgisayar görüşü ve görüntü işleme uygulamalarındaki gelişmelere katkıda bulunur. Genel olarak, BMS'ler çeşitli dinamik sistemlerin ve belirsizliği yüksek veri kümelerinin modellenmesinde kapsamlı bir şekilde kullanılmıştır. Belirsizliği ele alma becerileri, gerçek dünya sistemlerindeki karmaşık ilişkileri ve belirsizlikleri yakalayan esnek ve yorumlanabilir bir modelleme yaklaşımı sağlar. Tip-1 BMS'ler (T1-BMS'ler) olarak bilinen geleneksel BMS'lerin belirsizliği temsil etmede bazen yetersiz kaldığı bilinmektedir. Bu probleme çözüm için, daha esnek bir temsil sunan Tip-2 Bulanık Kümeler (T2-BK'ler) alternatif olarak önerilmiştir.

T2-BK'ler doğrusal olmayan ve belirsiz sistemleri daha iyi ifade edebilir ve T2-BMS'ler karmaşık problemlerin üstesinden gelmede daha iyi bir potansiyele sahiptir. Bununla birlikte, T2-BMS'leri öğrenmek, tasarım karmaşıklıkları ve bulanık kümelerle ilişkili parametreleri öğrenme ihtiyacı nedeniyle daha zorlayıcı bir problemdir. Aralık Değerli T2 BMS'lerin (ADT2-BMS'lerin) öğrenimi için, T1-BMS'lerde kullanılan klasik yöntemlerde olmak üzere, evrimsel algoritmaların veya YSA yaklaşımlarının kullanılmasında dahil olduğu farklı yaklaşımlar önerilmiştir. Bu yaklaşımlar, tasarım karmaşıklığını basitleştirmeyi ve T2-BMS'lerin performansını iyileştirmeyi amaçlar. Gelişmelere rağmen YSA'ları ve evrimsel algoritmaları T2-BKM'lerle entegre etmek T2-BMS'lerin parametre çokluğundan ve öğrenimlerindeki

karmaşıklıktan dolayı oldukça zorlayıcıdır. Ayrıca, T2-BMS'lerde uğraşılan verinin boyutu ve artan parametre sayısı sebepleriyle mevcut yaklaşımlarla çözülmesi mümkün olmayan bazı zorluklar mevcuttur. Son araştırmalar, bu zorlukların üstesinden gelmek için BMS'leri ve DYSA'ları bir araya getirerek bu problemleri çözmeye odaklandı. Bu yaklaşım ile hem DYSA'ların genelleştirme yeteneklerinin güçlü yanlarından hem de mini-örneklemeli (mini-bacth) optimizasyon algoritmalarının gücünden yararlanan hibrit modellerin geliştirilmesine yol açtı.

Bu tez çalışmasında, derin öğrenme tabanlı parametre öğrenme yöntemleri kullanılarak T2-BMS'lerin parametrelerinin öğrenilmesi için yeni bir yaklaşım önerilmiştir. Önerilen yaklaşım, kapsamlı veri setlerini ele almayı ve hem iyi bir tahmin doğruluğu hem de belirsizlikleri daha iyi ifade etme becerisine sahip modeller oluşturmayı amaçlamaktadır. Bu tez kapsamında spesifik olarak üç çalışma yapılmıştır: ilk çalışma (i) "Learning with Type-2 Fuzzy Activation Functions to Enhance the Performance of Deep Neural Networks", ikinci çalışmada (ii), "More Than Accuracy: A Composite Learning Framework for Interval Type-2 Fuzzy Logic Systems" başlıklı bir yapı önerildi ve (iii) son çalışmada, Genel T2 BMS'ler (GT2-BMS) için "Towards Reliable Uncertainty Quantification and High Precision with General Type-2 Fuzzy Systems" olarak adlandırılan güvenilir belirsizlik ölçümü yöntemine sahip bir yöntem önerildi.

İlk çalışmada (i), DYSA'ların öğrenme performansını artırmayı amaçlayan ADT2 Bulanık Aktivasyon Katmanı (ADT2-BAK) adlı yeni bir yöntemi tanıtıyoruz. ADT2-BAK, öğrenme yeteneklerini geliştirmek için DYSA yapısı içinde aktivasyon birimleri olarak kullanılan Tek girişli IT2 (TADT2) Bulanık Doğrultma Birimlerinden (BDB'lar) oluşur. TADT2-BDB yapısının kapalı biçimli bir temsilini bu tez çalışmasında sunduk ve bu yapının parametrelerinin girdi-çıktı eşlemelerinin oluşturulmasını nasıl etkilediğini anlamak için bir analiz yaptık. Araştırma bulguları, bu eşlemelerin ayarlanması gereken hiperparametreler veya öğrenilmesi gereken parametreler olarak kabul edilebileceğini göstermektedir. DÖ tabanlı yapılar kullanarak bu hiperparametreler için yeni bir öğrenme algoritmasını bu çalışmada sunmaktayız.

Önerilen ADT2-BAK'in etkinliğini değerlendirmek için ReLU, PReLU ve ELU gibi mevcut aktivasyon birimleriyle bir karşılaştırmayı bu çalışma sunmaktadır. Yeni TADT2-BDB, yalnızca yok olan türev (vanishing gradient) problemini ele almakla kalmaz, aynı zamanda daha hızlı bir yakınsama oranı sağlar. Bunu, negatif bölgede tanımlanan girdilerin işlenmesi yoluyla ortalama aktivasyonu sıfıra yaklaştırarak başarır. TADT2-BDB'nin bu özelliği, DYSA'ların daha yüksek bir öğrenme performansı sergilemesini sağlar. Seçilen veri kümeleri kullanılarak yapılan deneyler, ADT2-BAK yaklaşımının etkinliğini ve üstünlüğünü göstermektedir. ADT2-BAK aktivasyon birimleri ile DYSA'ların öğrenme yeteneklerini geliştirebilir ve daha sağlam ve esnek bir ağ yapısı sunabilir. Önerilen yaklaşım, deneysel sonuçların ortaya koyduğu gibi DYSA'ların performansını iyileştirme potansiyeline sahiptir ve ayrıca DYSA'ların öğrenme yeteneklerini geliştirme fırsatı verir.

İkinci çalışma (ii), belirsizliği yakalamak ve Tahmin Aralıklarını (TA'lar) oluşturmak için tipi indirgenmiş ADT2-BMS kullanan yeni bir bileşik öğrenme yaklaşımı sunar. Öncelikle model doğruluğuna odaklanan genel eğitim yaklaşımlarının aksine, bu yeni yaklaşımın amacı yalnızca yüksek tahmin doğruluğu elde etmek değil, aynı zamanda tipi indirgenmiş ADT2-BK'lerden yararlanarak belirsizliği etkili bir şekilde ele almak ve yakalamaktır. Böyle bir amaca ulaşmak için, bu bağlamda üç ana zorluk belirledik:

(1) belirsizliği ele alma yeteneği, (2) bir bileşik amaç fonksiyonunun oluşturulması ve (3) belirsizliği ele alan bir öğrenme algoritmasının ADT2-BMS'ler için geliştirilmesi.

(1)'de, bu zorlukların üstesinden gelmek için önerilen yaklaşım quantile regresyon ve DÖ bazlı parametre öğrenme yöntemlerini ADT2-BMS'ler ile birleştirerek ADT2-BMS'lerin tipi indirgenmiş çıkışlarından yararlanır. ADT2-BMS'lerin belirsizliği işleme yeteneği, kümelerin merkezini hesaplamak için kullanılan yöntemlere bağlıdır, temsil yetenekleri ise öncül ve sonuç üyelik fonksiyonlarının yapısı tarafından belirlenir. Tez kapsamında, çeşitli parametrik ADT2-BMS'ler tanıtmaktayız ve tüm ADT2-BMS'ler için öğrenilebilir parametreleri ve bunların eğitim sürecinde karşılanması gereken kısıtlamalarını tanımlamaktayız. (2)'de, sonrasında kısıtlanmış bileşik bir amaç fonksiyonuna dönüştürülen çok amaçlı bir amaç fonksiyonunun oluşturulmasını içeren amaç fonksiyonunun yapısı tanımlanmaktadır. Bu bileşik amaç fonksiyonu, doğruluğu optimize etmeyi amaçlayan log-cosh bileşenini ve belirsizliğin temsiline odaklanan bir tilted fonksiyon bileşenini içerir. Özellikle, tilted fonksiyonu, tipi-indirgenmiş ADT2-BK'lerin aralık çıkışlarını açık bir şekilde kullanmaktadır. (3)'te, kısıtlı olmayan optimizasyon yöntemleri kullanarak ADT2-BMS'leri eğitmek için bir DÖ tabanlı yaklaşımı sunulmaktadır. Çalışma ayrıca, ADT2-BMS'lerin kısıtlı optimizasyon problemini, bulanık kümelerin tanımlarını ihlal etmeden, kısıtlamasız bir probleme dönüştürmek için yeniden parametreleştirme tekniklerini tanıtmaktadır. Önerilen yaklaşımın etkinliğini değerlendirmek için kapsamlı karşılaştırmalı sonuçlar sunulmaktadır. Tezde, çeşitli kıyaslama veri kümeleri üzerinde yürütülen bir hiperparametre duyarlılık analizi ve modeller arası/içi karşılaştırmalar sağlamaktayız. Bu değerlendirmeler, önerilen yeni yaklaşımın belirsizlikle başa çıkma ve regresyon problemleri için yüksek tahmin doğruluğu elde etme performansına ışık tutmaktadır.

Çalışma (iii)'te, tahmin doğruluğunu iyileştirmek ve güvenilir TA'lar oluşturmak için α-düzlem tabanlı GT2-BMS'ler için yeni bir öğrenme yaklaşımı sunuyoruz. Yaklaşım, yeni bir bileşik amaç fonksiyonu aracılığıyla İkincil Üyelik Fonksiyonlarının (İÜF) şeklinden ve boyutundan faydalanır. Yeni amaç fonksiyonu iki ana bileşenden oluşur: belirsizlik odaklı bir amaç ve doğruluk odaklı terim. Belirsizlik odaklı amaç, yalnızca FOU olarak bilinen $\alpha_0 = 0$ düzlemiyle ilişkili tip indirgenmiş ADT2-BK'nin aralık çıkışını kullanılır. Bu, GT2-BMS'nin İÜF'nin boyut parametrelerinin belirsizliği ölçmesine ve aralık tahminlerinin öğrenmesine olanak tanır. Doğruluk odaklı kısım için iki alternatif amaç terimi önerilmiştir. Bir yaklaşımda, GT2-BMS'lerin durulaştırılmış çıktısı doğrudan kullanılırken, diğer yaklaşımda yalnızca $\alpha_K = 1$ seviyesiyle ilişkili çıktı kullanılır. Her iki durumda da, GT2-BMS'nin İÜF'nin parametreleri, yüksek hassasiyetle tahmin sağlamak için zorlanır. Böylece, önerilen amaç fonksiyonu içinde α-düzlemleriyle ilişkili ADT2-BMS'ye farklı roller atanır. $\alpha_0 = 0$ düzleminin çıktısı, GT2-BMS'nin çıktı hesaplamasına katkıda bulunmadığından, GT2-BK'lerin kısmen bağımsız bir şekilde öğrenilmesi mümkün olur ve yüksek doğruluğu korurken belirsizliğin yakalanmasını sağlar.

GT2-BMS'lerin öğrenme problemini ve ayrıca yüksek boyutlu ve karmaşık veriler ile başa çıkabilmek için GT2-BMS'ler için DÖ tabanlı bir parametre öğrenme yaklaşımı sunuyoruz. Bu, kısıtlı olmayan bir öğrenme problemi tanımlanarak başarılır. Ayrıca GT2-BK'lerin tanımlarının ihlal edilmemesi için bir yeniden parametreleştirme yöntemi önerdik. Önerilen öğrenme yaklaşımının üstünlüğünü göstermek için çeşitli veri kümelerini kullanarak istatistiksel karşılaştırmalı analizler yapıldı. Bu analizlerin sonuçları, yüksek hassasiyetle güvenilir belirsizlik ölçümü için umut verici bir çözüm olarak önerilen DÖ tabanlı yaklaşımla GT2-BMS'yi öğrenme potansiyelini göstermektedir.

1. INTRODUCTION

In the past decade, significant advancements have been made in the field of Deep Learning (DL), driven by the development of innovative learning methods and novel layer structures, along with the utilization of graphics cards for enhanced processing power [1-5]. This progress has enabled the training of neural network models with numerous hidden layers and neurons, leading to a substantial increase in the number of learnable parameters that can be optimized. As a result, Deep Learning has achieved remarkable breakthroughs in diverse domains, including semantic segmentation, object detection and classification, and handwritten character recognition. Deep Neural Networks (DNN)s have proven to be highly effective in various applications of machine learning and artificial intelligence [6-8].

DNNs, or Deep Neural Networks, are neural network structures with three or more hidden layers. They have become a significant advancement in the field of machine learning. In a generic DNN structure, the input vector is processed through the hidden layers to produce the output vector. In regression or classification problems, DNNs can be designed as a linear estimation function. However, for more complex problems, DNNs utilize nonlinear activation functions to create a nonlinear global approximator [1-11]. DNNs offer several advantages over traditional machine learning techniques. One key advantage is their ability to learn features at multiple layers. This allows DNNs to capture and represent complex features of the input data, leading to improved performance in solving intricate problems.

DNNs achieve this by computing their internal parameters through a forward pass and iteratively refining them during backpropagation, which effectively extracts meaningful features from the input data. These capabilities make DNNs superior to other methods in many tasks [12-16]. As a result of their effectiveness, DNNs have gained popularity and are now widely applied in various commercial and industrial applications. They have proven to be a valuable and powerful tool in solving complex problems and have brought significant advancements to the field of machine learning.

Over the past five decades, the utilization of Fuzzy Logic Systems (FLSs) has witnessed significant growth in various fields and applications. FLSs have been extensively used in various areas [17-22]. FLSs have been particularly used in control system design, where they offer a flexible and intuitive approach to handle complex and uncertain systems [17-19]. They provide a framework for incorporating linguistic variables and fuzzy rules to capture and model human-like reasoning and decision-making processes. This has led to advancements in the development of intelligent control systems that can effectively handle nonlinear and uncertain dynamics.

In decision-making processes, FLSs provide a formal framework for incorporating and aggregating uncertain and imprecise information from multiple sources. They offer a means to model and reason with uncertain knowledge, allowing decision-makers to make informed choices in complex and uncertain environments [20]. Furthermore, FLSs have been applied in image processing tasks, where they used in handling fuzzy sets to represent the uncertain data. By utilizing fuzzy techniques, FLSs enable robust image analysis, pattern recognition, and image understanding, contributing to advancements in computer vision and image processing applications [21, 22]. To summarize, FLSs have been extensively utilized in modeling various systems and phenomena, by leveraging their power to handling uncertainty, FLSs provide a flexible and interpretable modeling approach that can capture complex relationships and uncertainties in real-world systems. This has enabled improved understanding, analysis, and prediction in diverse domains [17-19].

Even though, conventional FLSs (also known as the Type-1 FLSs (T1-FLSs)) have been extensively and successfully employed into various applications, they may not be sufficient to handle high levels of nonlinearity and uncertainty [23-26]. One of the main reasons for this lack of efficiency in representing uncertainty is that the T1-FLSs is represent the membership degree of an element to a set using a crisp value [24]. In order to solve this problem, researchers have proposed the Type-2 Fuzzy Sets (T2-FSs) as an alternative to conventional T1-FSs. T2-FSs offer a more flexible and expressive representation by allowing the membership grade of an element to be a FS as well [24-28]. This added flexibility enables T2-FSs to better handle nonlinear and uncertain systems, as they can capture and model higher levels of uncertainty compared to T1-FSs [24].

T2-FLSs has a big potential to handle complex problems since they use T2-FSs in their antecedent parts which allows them to enhance the performance in dealing with uncertainties and nonlinearity [24, 25]. While they share a similar structure with their T1 counterparts, T2-FLSs introduce an additional level of complexity due to the use of T2-FSs. The design of T2-FLSs becomes challenging due to various factors, including the constraints imposed by T2-FS definitions, the large number of parameters that need to be tuned, and the intricacies of the Center of Sets Calculation Method (CSCM) [24, 25].

Learning of T2-FLSs presents unique difficulties compared to T1-FLSs, mainly coming from their design complexity [29, 30]. The construction of T2-FSs adds an extra degree of complexity, as it requires learning the parameters associated with these fuzzy sets. Moreover, the CSCM, which is responsible for determining the centroids and spreads of the IT2-FSs, adds another level of intricacy to the learning process [29-33]. To address these challenges and simplify the design complexity, different approaches have been proposed. One common approach is to adapt pre-trained T1-FLSs to T2-FLSs by blurring the membership functions of the T1-FLSs, allowing them to handle uncertainties. This technique leverages the existing knowledge embedded in the T1-FLSs and extends it to T2-FLSs, reducing the need for extensive parameter tuning. Another approach is to employ evolutionary algorithms or Neural Network (NN) approaches to optimize the parameters of Interval T2-FLSs (IT2-FLSs). These methods utilize the optimization capabilities of evolutionary algorithms or the learning capabilities of NNs to fine-tune the parameters and improve the performance of T2-FLSs [33]. In order to tackle the challenges posed by the complex design and learning of IT2 neuro-fuzzy systems, researchers have explored various approaches, including the integration of neural networks and evolutionary algorithms. For instance, self-evolving IT2 neuro-fuzzy systems have been developed, where the system evolves and adapts its structure and parameters over time [34]. Fuzzy clustering and evolutionary algorithms have also been employed to optimize the structure and parameters of IT2 neuro-fuzzy systems [35]. Additionally, self-organizing IT2 neuro-fuzzy systems have been proposed to autonomously organize their structure and adapt to changing environments [36]. Another approach involves simultaneous feature selection and system identification in IT2 neuro-fuzzy systems, leveraging the capabilities of neural networks for improved performance [37]. To address the challenges associated with

extensive datasets, support vector machines (SVMs) have been integrated with IT2 neuro-fuzzy systems. This integration aims to reduce the complexity of the system structure and improve the learning performance [38]. By incorporating SVMs, which are effective in handling high-dimensional data, the curse of dimensionality can be mitigated. However, SVM based methods still are not able to handle large number of sample data.

Despite the advancements in integrating neural networks and evolutionary algorithms with T2-FLSs, these methods encountered challenges when applied to extensive datasets [39, 40]. One major difficulty arises from the curse of dimensionality, which refers to the increased complexity and computational requirements when dealing with high-dimensional data. As the number of input features increases, the data becomes sparser, making it harder for the learning algorithms to capture meaningful patterns and relationships. Another issue is the increasing number of parameters in T2-FLSs, especially for General T2-FLSs (GT2-FLSs), which depend on the size of the rules. As the number of rules grows, the complexity of the system also escalates, leading to a higher number of parameters that need to be optimized. This parameter increase can result in longer training times, increased computational resources, and a higher risk of overfitting [40-43].

Recent research has demonstrated successful integration of FLSs and IT2-FLSs with DL techniques. These advancements have led to the development of novel approaches that combine the strengths of FLSs and neural networks to tackle various challenges [39-43]. One such approach is the introduction of a Takagi-Sugeno type deep fuzzy network, which leverages the capabilities of both FLSs and neural networks [39]. This hybrid model aims to benefit from the interpretability and rule-based reasoning of FLSs while harnessing the learning and generalization capabilities of neural networks. By combining these two paradigms, researchers have search for an enhancement in the performance and explainability of the models. In the context of handling big data regression problems, concepts from deep learning, such as mini-batch gradient descent, regularization, and AdaBound, have been extended to Takagi-Sugeno FLSs [40]. These extensions aim to address the challenges associated with large-scale datasets, enabling efficient training and improved generalization performance. Additionally, researchers have proposed an IT2 fuzzy logic-based stacked autoencoder deep neural network [41]. This model incorporates IT2-FLSs into the architecture of

stacked autoencoders, a popular deep learning technique. The use of IT2-FLSs in this framework aims to generate explainable artificial intelligence models by providing interpretable linguistic rules [43]. Even though DL has been widely adopted for its ability to learn complex patterns and features from data, the specific usage of DL algorithms to learn parameters of IT2/GT2-FLSs directly is still relatively uncommon. The main focus has been on using the powerful abilities of IT2-FLSs, such as their ability to handle uncertainty and nonlinearity, within the DL structures. In this context, the focus has been generally on integrating IT2-FLSs into DL architectures [44-46] while the employing DL based algorithm for directly learning the IT2-FLSs has been relatively uncommon. However, we started to see some new works in which DL based algorithms are employed to learn the parameters of the IT2/GT2-FLSs [47-50].

In this thesis study, we propose a novel approach to learn the parameters of the T2-FLSs (IT2-FLSs and GT2-FLSs). In the new approach, we use widely used deep learning parameter learning methods, we will introduce the new approach in previous section 3, to learn the parameters of the T2-FLSs. In the novel approach we propose a mini-batch sampled deep learning parameter learning method since the conventional parameter learning methods for T1/T2-FLSs face difficulties in learning extensive datasets because of problems such as the curse of dimensionality and an increasing number of parameters that depend on the size of the rules. The proposed structure is used in different ways to end up with models have ability to both high prediction accuracy and handle uncertainties. In the first study, namely “Learning with Type-2 Fuzzy Activation Functions to Improve the Performance of Deep Neural Networks”, we propose to use Single input IT2-FLS as the activation functions of the DNN by aiming to increase the learning performance of the DNNs. In the second study which has a title as “More than Accuracy: A Composite Learning Framework for Interval Type-2 Fuzzy Logic Systems”, we introduce a novel composite learning framework specifically designed for training regression models using IT2-FLSs with the primary objective; achieving a high accuracy performance while also having the ability to representing and handling the uncertainties. In the final work titled as “Towards Reliable Uncertainty Quantification and High Precision with General Type-2 Fuzzy Systems”, we present a step towards learning reliable uncertainty quantification and high precision performance via α-plane based General Type-2 GT2-FLSs.

In the study titled with “Learning with Type-2 Fuzzy Activation Functions to Improve the Performance of Deep Neural Networks”, we propose a novel approach called IT2 Fuzzy Activation Layer (IT2-FAL) to enhance the learning performance of DNNs. Our IT2-FAL is composed of Single input IT2 (SIT2) Fuzzy Rectifying Units (FRUs) which are used in DNN structure as activation units to improve the learning capabilities. We will introduce the internal structure of the IT2-FAL in section 3.4 and discuss its properties and learning methodology. Specifically, we will derive a closed-form representation of the SIT2-FRU structure and analyze how its parameters impact the generation of input-output mappings. Through our investigations, we will demonstrate that these mappings can be treated either as hyperparameters to be set or as parameters to be learned. To evaluate the effectiveness of the proposed IT2-FAL, we will compare its performance with existing activation units such as ReLU, PReLU, and ELU. We will conduct experiments using selected benchmark datasets to demonstrate the efficiency and superiority of our IT2-FAL approach. Using the IT2-FAL and its activation units, SIT2-FRU components, we give DNNs the chance to improve their learning capabilities and also provide a more robust and flexible networks structure. Our proposed approach has the potential to improve the performance of DNNs across various applications and domains.

The traditional training approach for IT2-FLSs has primarily focused on optimizing accuracy, by using the potential benefits of the FOU to handle uncertainty. In these conventional approaches, the FOU is treated as additional design parameters to be trained, rather than utilizing it as an opportunity to capture and represent uncertainty effectively. The work “More than Accuracy: A Composite Learning Framework for Interval Type-2 Fuzzy Logic Systems” presents a new novel composite learning approach by exploiting the type-reduced sets of IT2-FLSs to capture uncertainty and establish Prediction Intervals (PIs) unlike the mainstream training approaches. The objective of this new approach is not only to achieve high accuracy in predictions but also to effectively address and capture uncertainty through the use of type-reduced sets. In order to achieve our goal, we identify three key challenges that need to be addressed:

1. The first challenge is related to the capability of IT2-FLSs in handling uncertainty. IT2-FLSs offer the potential to represent and account for uncertainty due to their utilization of type-reduced sets. However, effectively

harnessing this capability and incorporating it into the learning process is a significant challenge.

2. The second challenge lies in the construction of an appropriate loss function. Traditional training approaches for IT2-FLSs have focused primarily on accuracy optimization, neglecting the importance of explicitly considering and minimizing uncertainty. Developing a loss function that incorporates both accuracy and uncertainty components is crucial to ensure that the trained IT2-FLS can effectively cover uncertainty in its predictions.
3. The third challenge is developing an algorithm that can handle the learning complexity associated with the size of the dataset while also considering the constraints imposed by IT2-FLSs. IT2-FLSs are characterized by a large number of parameters and complex design constraints, making the learning process computationally demanding. Developing an efficient algorithm that can effectively handle these complexities is essential for training IT2-FLSs in practical settings.

The capability of IT2-FLSs to handle uncertainty is influenced by two key components: inference and structure. These components play a significant role in determining how uncertainty is captured and processed within the IT2-FLS. In this thesis, we focus on exploring different approaches to uncertainty processing, starting with the Center of Sets Calculation Method (CSCM). We introduce three CSCMs: Karnik-Mendel (KM) [24], Nie-Tan (NT) [51], and the Simple Method (SM) that directly utilizes the boundary FLS. These CSCMs define how uncertainty is computed and incorporated into the IT2-FLS inference process. Furthermore, we construct various IT2-FLSs by considering the structure of the antecedent MFs and the rule consequents. Specifically, we examine IT2-FSs with different sizes of FOU for the antecedents, and we also investigate the use of crisp or Interval Valued (IV) functions for the rule consequents. This allows us to create IT2-FLSs that capture uncertainty either solely through their antecedents or by utilizing IT2 and IV FSs in both the antecedents and consequents.

In order to construct the loss function for training IT2-FLSs, we adopt a multi-objective optimization approach. The loss function consists of two components: an empirical loss term for accuracy and a tilted loss term, denoted as l_τ, for capturing the

uncertainty. The tilted loss function, commonly used in Quantile Regression (QR), allows us to define a desired quantile range to be covered, thus providing a measure of uncertainty. To simplify the learning process and avoid increasing complexity, we transform the multi-objective loss into a composite loss using a hyperparameter. This allows us to balance the emphasis between accuracy and uncertainty coverage. It is important to choose an appropriate empirical risk function that aligns well with the tilted loss l_τ, as both loss components share parameters during training. In this context, we propose using the $L_{log-cosh}$ norm as the empirical risk function. This gives opportunity to align better with l_τ and also, we combine advantageous characteristics of both L_1 and L_2 norms using $L_{log-cosh}$ [52]. The proposed loss function explicitly exploits the type-reduced set ($[\underline{y}, \overline{y}]$) and the defuzzified output (y) to train IT2-FLSs that can effectively capture uncertainty while achieving high mean accuracy performance.

To address the challenges of learning IT2-FLSs, we need to consider the complexities associated with the size and dimensionality of the data, as well as the constraints defined by the learnable parameters IT2-FLS since they are parameters of IT2-FSs. DL offers an efficient approach to tackle the complexity of large datasets. However, directly applying DL to IT2-FLS learning is not feasible due to learnable parameters must satisfy some constraints, since they are also the parameters of IT2-FSs, which makes the problem a constrained optimization problem. In order to overcome this limitation, we propose novel parameterization tricks that allow us to transform the constrained optimization problem into an unconstrained one. By employing parameterization tricks, we can reformulate the learning problem for IT2-FLSs in a way that is compatible with DL algorithms. This transformation enables us to leverage the power of DL techniques to effectively learn the parameters of IT2-FLSs from data.

In the work, “Towards Reliable Uncertainty Quantification and High Precision with General Type-2 Fuzzy Systems”, we propose a new learning approach for α-plane based GT2-FLSs that exploits the shape and size of the Secondary MF (SMFs) through a novel composite loss to increase the accuracy of the pointwise prediction while generating reliable PIs.

The novel composite loss function is composed of an uncertainty quantification-focused loss and an accuracy-focused term. Within the uncertainty-focused loss term,

we explicitly used only the type-reduced set of IT2-FLS associated with the $\alpha_0 = 0$ plane, i.e. the FOU. Thus, we imposed the SMF size parameters of the GT2-FS to quantity the uncertainty and thus to learn PIs. For the accuracy-focused part, we provided two alternative loss terms. In one of them, we used directly the aggregated output of the GT2-FLSs while in the other one, we only used the output associated with the $\alpha_K = 1$ level. For both alternatives, we enforced the SMF shape parameters of the GT2-FS to learn pointwise prediction with high precision. Thus, we assigned different roles for the IT2-FLS associated with α-planes within the proposed loss function. As the output of the $\alpha_0 = 0$ plane is not contributing to the output calculation of the GT2-FLS, an (partially) independent learning of the GT2-FSs is possible for capturing uncertainty while ending up with high accuracy. For efficient learning in the presence of high dimensional and complex data, we also present a DL approach for GT2-FLSs by defining an unconstraint learning problem that does not violate the definitions of GT2-FSs via parameterization tricks. To show the superiority of the proposed learning approach, statistical comparative analyses will be presented in section 6, using benchmark datasets. The results show the potential of learning GT2-FLS as a promising solution for reliable uncertainty quantification in real-world applications with high precision.

2. PRELIMINARY INFORMATION ON FUZZY SETS AND SYSTEMS

In this part of the thesis, we will give some preliminary information about the fuzzy sets and systems. In the following section (2.1), Type-1 fuzzy sets will be introduced firstly and then conventional, also T1-FLS is used, fuzzy logic systems and their internal structure will be introduced. After that we will extend the constructed theory on T1-FSs and FLSs into to Type-2 fuzzy sets and Type-2 fuzzy logic systems. In general, T2-FLSs can be grouped as General T2-FLSs and Interval T2-FSLs which can be seen as a special form of the GT2-FSLs. In section 2.2, firstly, we will give a brief insight on IT2-FSs and then introduce the IT2-FLSs. Finally, we will cover the GT2-FS in section 2.2.2 and conclude this chapter by presenting the internal structure of the GT2-FLSs.

2.1 A Brief Overview on Type-1 Fuzzy Sets and Systems

Unlike the classical logic, based on binary 0 and 1 values, fuzzy logic allows an element to member of a set with a degree of membership between 0 and 1 [53]. The theory is proposed as a part of the fuzzy set theory [53]. T1-FS can be defined a set with uncertain boundaries [53]. Assume a universe of discourse (X) and its members are represented with x. Therefore, definition of a T1-FS can be described as:

$$X = \{x\} \tag{2.1}$$

The membership degree ($\mu_A(x)$) is used to describe a T1-FS (A) in X, which is attributed to each element x in X. Therefore, we can define a T1-FS (A) lies in X and be expressed using the following equation [24]:

$$A = \{(x, \mu_A(x)) | x \in X\} \tag{2.2}$$

Here, the equation defines a T1 fuzzy set A in X and x is the members of the X. Thus, X also can be defined the value set of x. Furthermore, $\mu_A(x)$ is a function which defines the how a x value belongs to X and values of $\mu_A(x)$ defines the membership degree of

x. In an alternative form, the definition of a T1-FS can be also formulized as follows [54]:

$$\mu_A: X \rightarrow [0,1] \tag{2.3}$$

A T1-FS (A) can also be defined for various notations. It is common to use the following notation for continuous universe X, in fuzzy society, and it is shown as below [24]:

$$A = \int_{x \in X} \mu_A(x)/x \tag{2.4}$$

The notation above shows that the membership degree of x, denoted as $\mu_A(x)$, is bounded between 0 and 1. In this notation, "/" term is used to associate the elements in X universe, while $\int$ is used to denote union over x values for X universe. In literature, MFs determines the membership degree of the input x. The MFs are generally selected commonly used functions such as triangles, gaussians, trapezoids, sigmoids, etc. Additionally, the α-cut of a T1-FS is represented as A_α, where α is the α-cut level (i.e., $\alpha \in [0,1]$), and is defined as follows [24]:

$$A^\alpha = \{x | \mu_A(x) > \alpha\} \tag{2.5}$$

In fuzzy literature, the most commonly used MF are the triangle T1-FSs and Gaussian type T1-FSs. A triangle T1-FS is defined as follows:

$$\mu_A(x) = \begin{cases} \dfrac{x-l}{c-l} & l < x \leq r \\ \dfrac{r-x}{r-c} & c < x \leq r \\ 0 & otherwise \end{cases} \tag{2.6}$$

Here, l and r represent the left support point and right support point, respectively, while the c stands for the center of the triangle T1-FS. Figure 2.1a demonstrates the illustration of a triangle type T1-MF.

Mathematical definition of a Gaussian T1-FS can be made as follows:

$$\mu_A(x) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})} \tag{2.7}$$

where c is the mean while the σ stands for the standard deviation values of the Gaussian type T1-MF. T1 Gaussian MF is shown in Figure 2.1b.

A FLS is a sophisticated system that uses fuzzy rules in a rule table to map the inputs to the outputs. FLSs have various applications, including reasoning, decision-making, modeling, and control [25, 26]. In literature, we encounter with two types of FLSs; the first one is the Mamdani type FLSs and the other one is Takagi-Sugeno-Kang type FLSs. In the literature, they are the most commonly used ones, yet in the scope of this thesis we only consider the Takagi-Sugeno-Kang type FLSs.

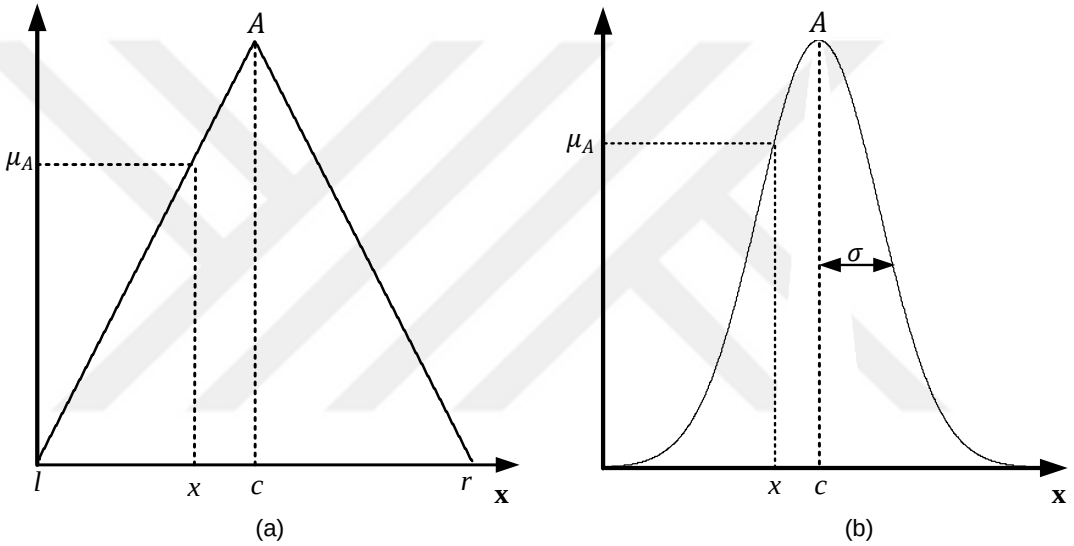


Figure 2.1 : An Illustration of the commonly used T1-FSs, (a) triangle T1-FS, (b) Gaussian T1-FS.

The IF-THEN rules in both FLSs have a similar antecedent part, but the consequent part varies because different consequent MFs are utilized. The Mamdani type and Takagi Sugeno Kang (TSK) type FLSs are differentiate in their consequent MFs. The Mamdani one uses T1-FSs as their consequent MFs, while the TSK one employ linear functions or crisp values as their MFs. In this thesis context, we consider only TSK type FLSs. Thus, the FLS structures defined in this thesis could be used T1, IT2, GT2 antecedent part, while the consequents parts could be defined with singletons, linear functions or interval valued functions.

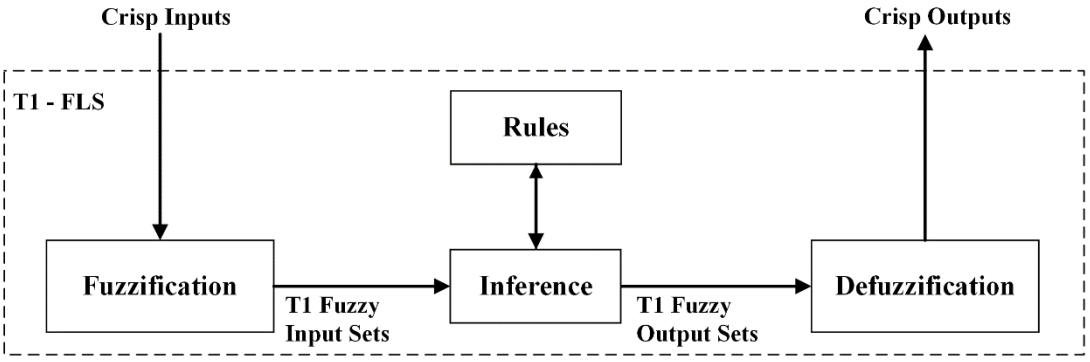


Figure 2.2 : An Illustration of a type-1 fuzzy logic system.

Figure 2.2 shows the overall structure of the T1-FLS, which comprises four primary components. As seen in Figure 2.2, a crisp input is fuzzified by the fuzzification component, rules are saved in the rule-base, inference engine process the T1-FS inputs and deffuzzification component generates the crisp output. The process can be summarized as the fuzzification component is in charge of transforming precise values into T1 fuzzy sets. After fuzzification of the inputs, the inference engine and rule-base collaborate to produce a fuzzy output value. Subsequently, the defuzzifier transforms this fuzzy output into a crisp value.

The definition of a TSK type fuzzy rule can be defined as follows [24]:

$$R_p : \text{If } x_1 \text{ is } A_{i,1} \text{ and } \dots \text{and } x_M \text{ is } A_{i,M}, \text{Then } y_{T1} \; is \; y_p = a_{p,m}x_m + b_p \tag{2.8}$$

In this rule structure, R_p is the pth rule of this rule definition, ($p \in \{1,2,\dots,P\}$), where the number of all rules is represented with P. Then, the inputs are symbolized with x_m, ($m \in \{1,2,\dots,M\}$) and total input number is symbolized with M, while the output of the T1-FLS, which is defuzzified and represented in crisp form, is denoted as y_{T1}. Besides, the antecedent Membership Functions (MFs) are defined with $A_{i,m}$ ($i \in \{1,2,\dots,I\}$) and the total number of antecedent MFs are shown with I. Consequents MFs can be selected as a crisp number, as we set $a_{p,m} = 0$ for $\forall\, p, m$, or we can select it as a linear function as we set select $y_p = a_{p,m}x_m + b_p$. Thus, y is the membership value of the consequent for the rule Rp. The defuzzified output y_{T1} can be computed by taking the weighted sum of the crisp values associated with the consequent membership values, where the degree of firings serves as its weight. This sum is then divided by the total firings which can be formulated as [24]:

$$y_{T1} = \frac{\sum_{p=1}^{P} f_p y_p}{\sum_{p=1}^{P} f_p} \tag{2.9}$$

in the above equation the firing strength is denoted by f_p and it is described as:

$$f_p = \mu_{A_{i,1}}(x_1) \cap \ldots \cap \mu_{A_{i,M}}(x_M) \tag{2.10}$$

Here, $\cap$ operator represents the intersection, in literature it is also knowns as the t-norm, which can be defined with a min operator or numerical product. It is also select a different t-norm operator among the proposed ones in literature, however we only consider to use min or product in the scope of thesis.

2.2 A Brief Overview on Type-2 Fuzzy Sets and Systems

The definition of T2-FLSs is coming from the T2-FSs in which they defined in the antecedent part of the T2-FLSs. T2-FSs can be seen as the extended version of their T1-FS counterpart and they generally use similar mathematical operators [24-26], [55-68]. T2-FSs have more design parameters compared to T1 counterparts and these extra design parameters makes them more useful tools, yet designing a T2-FSs is much more complex than the T1 ones [24]. In contrast to T1-FSs, which use a crisp number to represent the degree of membership, T2-FSs use another T1-FS to define the degree of membership of an element [24-26]. One particular kind of T2-FS is the IT2-FS. For IT2-FSs, the membership degrees generally constructed with T1-FSs for the primary ones and interval valued FSs, their membership degrees always equal to one, for secondary ones. Consequently, it can be concluded that IT2-FSs is extended version of T1-FSs, while they are also a special kind of GT2-FSs. T2-FSs share similarities with T1-FSs in that they also define the primary membership function ($\mu(x)$) for the input variable. However, T2-FSs use J_x to represent the primary membership of x, instead of the membership degree ($\mu(x)$) used in T1-FSs.

In equation 2.2 we describe a T1-FS which has an input variable (x) and the membership function ($\mu(x)$), while a T2-FS is described with a primary MF and their corresponding membership degrees also with a secondary MF and their degrees of memberships as demonstrated in equation 2.11. In a T2-FS, the secondary variable (u) belongs to a primary membership (J_x), and the primary memberships serve as inputs

for the secondary membership function ($\mu_{\tilde{A}}(x,u)$). If the secondary MF is defined with an interval valued function, it leads to the formation of an IT2-FS [56-60]. In general, a T2-FS ($\tilde{A}$) can be constructed as in following [24]:

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) | \forall x \in X, \ \forall u \in J_x \subseteq [0,1]\} \tag{2.11}$$

In the above equation, the primary elements of the T2-FS $\tilde{A}$ is defined with x and the primary universe is defined with X. The secondary variable of the T2-FS is symbolized with u and its universe of discourse is defined with $U \in [0,1]$. The membership function of the T2-FS is defined with $\mu_{\tilde{A}}(x,u)$. At that point T2-FSs are differentiate from the T1 counterparts since their membership grades are formed in a 3D domain ($X \times U_X$). On the other hand, the membership grade of a T1 membership function ($\mu_A(x)$) lies in 2D universe (X) [55].

Similar to its T1 equivalent, a T2-FS $\tilde{A}$ can be represented using fuzzy set notation for continuous universe (X) as described in below:

$$\tilde{A} = \int_{x \in X} \int_{u \in J_x} \mu_{\tilde{A}}(x,u)/(x,u), \ \ J_x \subseteq [0,1] \tag{2.12}$$

Here, $\iint$ represents the union of the all x and u values. The degree of membership for a T2-FS is calculated by both the primary MF (J_x) and the secondary MF ($\mu_{\tilde{A}}(x,u)$). More specifically, the membership grade of a type-2 MF is defined with these two factors which are defined as

$$\mu_{\tilde{A}(x)}(u) = \int_{u \in [0,1]} \mu_{\tilde{A}}(x,u)/(u) \tag{2.13}$$

$$J_x = \{(x,u) | \ u \in [0,1], \ \mu_{\tilde{A}}(x,u) > 0\} \tag{2.14}$$

here $\tilde{A}(x)$ stands for the FS of the secondary one. Also, the primary membership can be described as follows:

$$J_x = [\underline{\mu}_{\tilde{A}}(x), \overline{\mu}_{\tilde{A}}(x)] \tag{2.15}$$

Where $\underline{\mu}_{\tilde{A}}(x)$ and $\overline{\mu}_{\tilde{A}}(x)$ represent the Lower MF (LMF) and Upper MF (UMF) of the T2-FS. Thus, it can be stated that J_x is defined by the integration of the LMF and UMF.

The distinction between T1 /T2-MF comes from the number of membership degrees. T1-MFs have a single membership degree, represented by a value for J_x, while T2-MFs have an infinite number of membership degrees, represented by a union set of intervals for J_x.

Additionally, the characteristic of SMF of the T2-FS affects the value of the crisp output. It also defined the type of the FS wheatear it is a T1, an IT2 or GT2 FSs. If the SMF is a crisp number than the FS is characterized as a T1-FS. In the case of an IT2-FS, the SMF is an interval values FS with all the membership degrees are equal to 1. If the SMF is defined with a T1-FS, then the T2-FS is categorized as a GT2-FS.

Another advantageous property of the T2-FSs are defined with the area between the LMF and UMF namely FOU. FOU actually can be seen as the combinations of all J_x values and can be formulized as:

$$\bigcup_{x \in X} J_x = \left\{ (x,u) \middle|\, x \in X,\ u \in \left[\underline{\mu}_{\tilde{A}}(x), \overline{\mu}_{\tilde{A}}(x) \right] \right\} \tag{2.16}$$

Here, the above equation indicates that the region between LMF and UMF defines the FOU. Moreover, LMF and UMF is the edges of the FOU.

2.2.1 Interval type-2 fuzzy sets and systems

As we previously mentioned IT2-FSs can be defined as a particular for a GT2-FSs where the SMFs are interval valued FSs with all the secondary membership grades are equal to 1. An IT2-FS can be formulized as in following [24]:

$$\tilde{A} = \{(x,u), \mu_{\tilde{A}}(x,u) = 1 |\ \forall x \in X,\ \forall\, u \in J_x\} \tag{2.17}$$

In fact, an IT2-FS is combination of an upper MF and a lower MF. Besides, the region between the LMF and UMF defines the foot print. In this context, it is possible to visualize IT2-FS in 2D space similar to T1FSs. In Figures 2.3, 2.4 illustrations of the triangle IT2-MF and Gaussian type IT2-MF are shown, respectively.

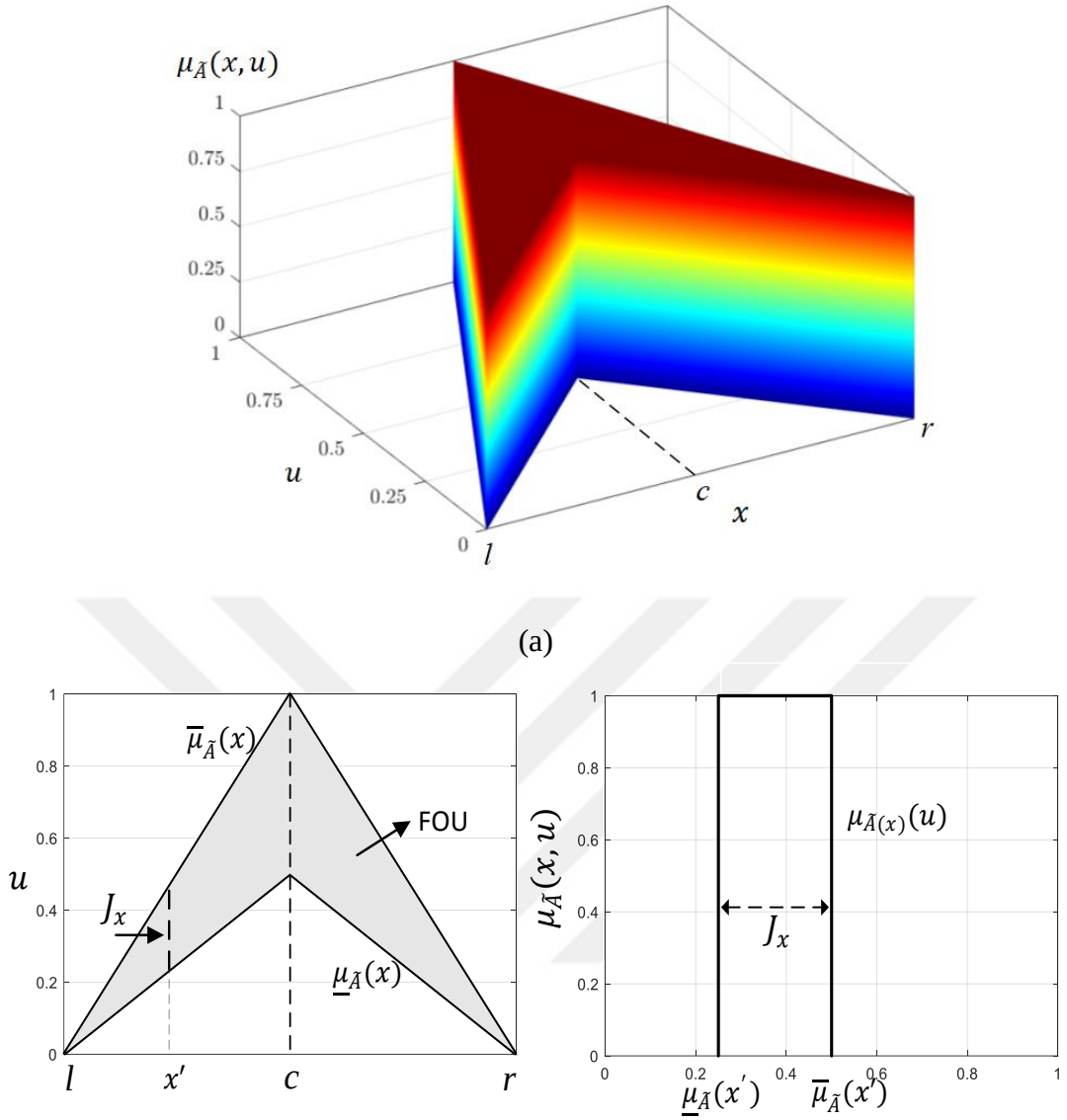
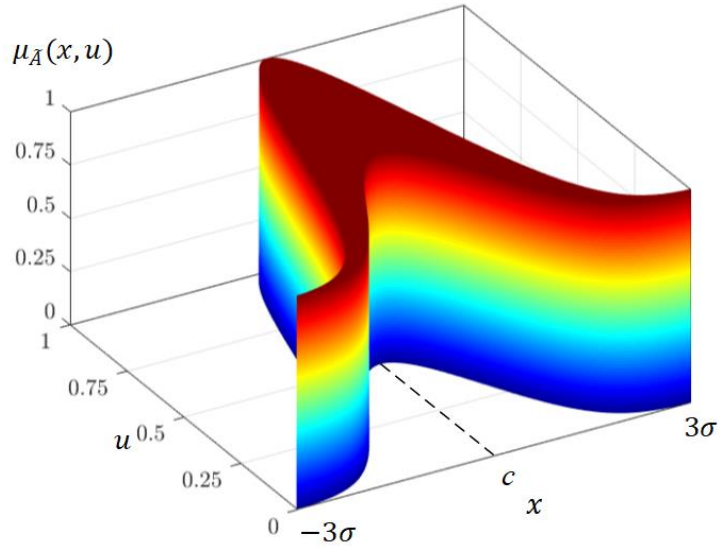
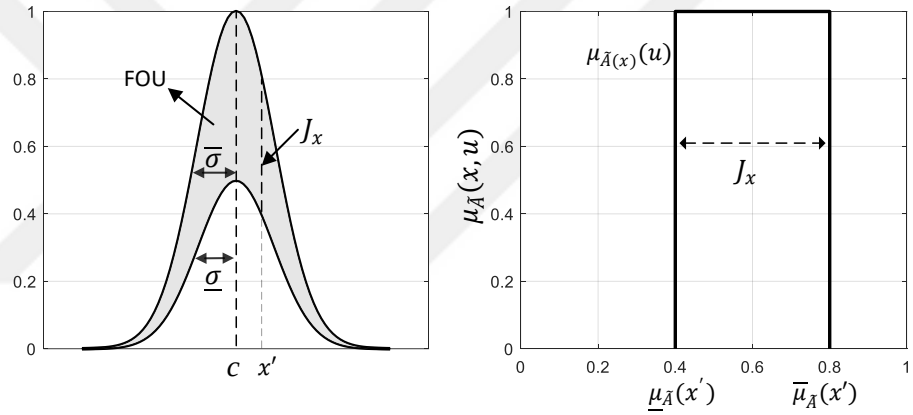


Figure 2.3 : Illustration of a triangle IT2-MF (a) IT2-FS, (b) Primary MF, (c) Secondary MF.

In Figure 2.5 illustrates the differences between T1 and IT2 in their SMF. In the case of T1-MFs, there is no FOU since we can see a T1-FS as their LMF ($\underline{\mu}_{\tilde{A}}(x)$) and UMF ($\overline{\mu}_{\tilde{A}}(x)$) is same and height of the LMF is set to one. Thus, this situation, as seen in Figure 2.5a, results with a crisp SMF. On the other hand, IT2-FSs are characterized by a predetermined interval which generates their FOU which is covered by the LMF ($\underline{\mu}_{\tilde{A}}(x)$) and UMF ($\overline{\mu}_{\tilde{A}}(x)$). This feature enables greater flexibility in designing the IT2-FSs, and designing SMFs is not a viable option, as all secondary grades are assigned a value of 1, as showed in Figure 2.5b.

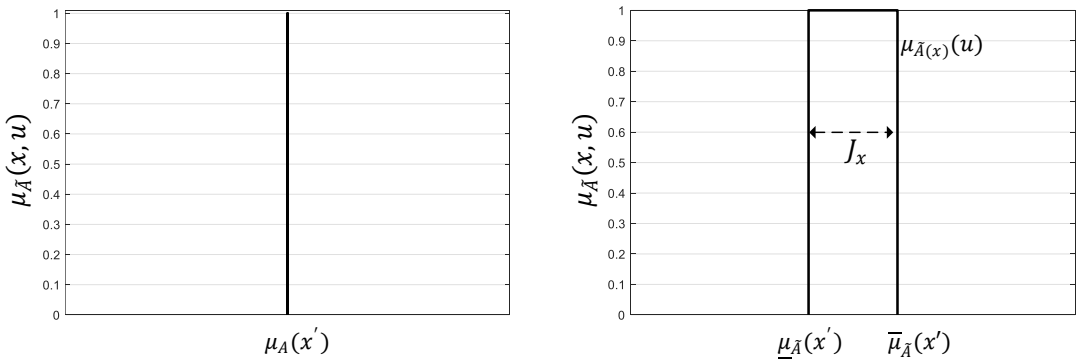


(a)



(b) (c)

Figure 2.4 : Illustration of a Gaussian IT2-MF (a) IT2-FS, (b) Primary MF, (c) Secondary MF.



(a) (b)

Figure 2.5 : SMFs of (a) T1-FS, (b) IT2-FS.

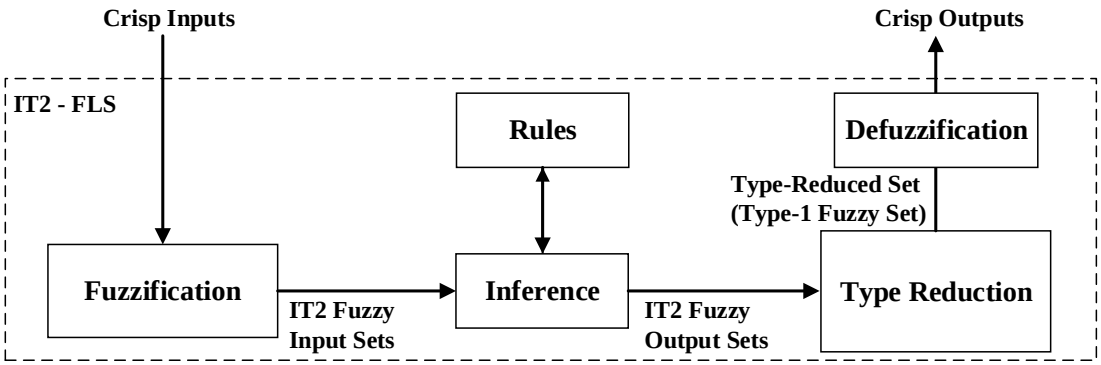


Figure 2.6 : An Illustration of an interval type-2 fuzzy logic system.

IT2-FLSs is constructed with IT2-FS in their antecedent memberships, which sets them apart from T1-FLSs. Despite this difference, the structure of the IT2-FLS is parallel to T1-FLS. Similar to T1-FLS, as seen in Figure 2.6 in an IT2-FLS, a crisp input is fuzzify by the fuzzification component, rules are saved in the rule-base, inference engine process the IT2-FS inputs and deffuzification component generates the crisp output. However, the IT2-FLS has an additional component called a type-reducer. This part of the system is responsible for converting IT2-FSs into T1-FSs. After this operation, T1-FSs are used by the defuzzifier which gives the crisp output. Figure 2.6 provides a block diagram that illustrates the components of an IT2-FLS.

In IT2-FLSs, the type reducer has an important job in completing the output calculation, but it causes a significant increase in computational cost that is considered the primary bottleneck of the system [24]. It should be noted that there are various type reduction methods are available in literature. KM type reduction [61] and NT [51] type reduction methods are the widely used methods in literature. Therefore, we also picked these two methods in this thesis to be employ as the type reducer for the employed IT2-FLSs.

IT2-FLSs employ If-Then rule structures to describe both antecedent and consequent information, with IT2-FSs used to represent these components [58]. To calculate the system output, crisp inputs are initially transformed into IT2-FSs. Once the input values are provided in IT2 fuzzy format, they are subjected to rule processing and inference to generate IT2 fuzzy output values. Subsequently, the type reduction component coverts these IT2-FSs to T1-FSs, and the output is calculated using the defuzzifier [58].

The IT2-FLSs utilized in this thesis study employ a Takagi-Sugeno type FLS. To define the antecedent MFs of the IT2-FLS, IT2-FSs are employed, whereas the consequent membership functions are determined through crisp, linear, or IV-FSs. As a result, the rule definition of the IT2-FLS is expressed in the following manner:

$$R_p : \text{If } x_1 \text{ is } \tilde{A}_{i,1} \text{ and } \dots \text{and } x_m \text{ is } \tilde{A}_{i,m} \text{ , Then } y_{IT2}^1 \; is \; \tilde{Y}_p^1 \; and \dots y_{IT2}^T \; is \; \tilde{Y}_p^T \tag{2.18}$$

Here, $\tilde{Y}_p^t = [\underline{y}_p^t, \overline{y}_p^t]$ are consequent MFs that are defined as:

$$\begin{aligned} \underline{y}_p^t &= \sum_{m=1}^{M} a_{p,m} x_m + a_{p,0} - \sum_{m=1}^{M} b_{p,m} |x_m| - b_{p,0} \\ \overline{y}_p^t &= \sum_{m=1}^{M} a_{p,m} x_m + a_{p,0} + \sum_{m=1}^{M} b_{p,m} |x_m| + b_{p,0} \end{aligned} \tag{2.19}$$

For simplicity we drop the t defined for $\tilde{Y}_p^t = [\underline{y}_p^t, \overline{y}_p^t]$, y_{IT2}^t and we use $\tilde{Y}_p^t = [\underline{y}_p \, , \overline{y}_p \,]$, y_{IT2} to point out for the one particular output of the IT2-FLS, all operations are same for the other outputs.

In this rule structure, R_p is the pth rule of this rule definition, ($p \in \{1,2, \dots, P\}$), where the number of all rules is represented with P. Then, the inputs are symbolized with x_m, ($m \in \{1,2, \dots, M\}$) and total input number is symbolized with M, while the output of the IT2-FLS, which is defuzzified and represented in crisp form, is denoted as y_{IT2}. Moreover, the IT2 antecedent Membership Functions (MFs) are defined with $\tilde{A}_{i,m}$ ($i \in \{1,2, \dots, I\}$) and the total number of antecedent MFs are shown with I. For an IT2-FLSs, the antecedent IT2-FSs are described with an upper MF $\overline{\mu}_{\tilde{A}_{i,m}}(x_m)$ and a lower MF $\underline{\mu}_{\tilde{A}_{i,m}}(x_m)$ unlike its T1 counterpart. Moreover, T is the total number of outputs of the IT2-FLS. y_{IT2} is the defuzzified crisp output of IT2-FLS for t. Consequents MFs can be selected as a crisp number, as we set $a_{p,m} = 0$, $b_{p,m} = 0$ and $b_{p,0} = 0$ for $\forall \, p, m$, or we can select it as an linear functions as $y_p = a_{p,m} x_m + a_{p,0}$ by setting $b_{p,m} = 0$ and $b_{p,0} = 0$. Thus, $\tilde{Y}_p$ is the membership function of the consequent for the rule R_p.

Remark: It is also worth to mention that the consequent MFs can be selected as IV-FSs. We can set $a_{p,m} = 0$, $b_{p,m} = 0$ for $\forall\, p, m$ to define the consequent MFs with IV-FSs.

The total firing interval of a rule is defined as

$$\tilde{F}_p = [\underline{f}_p\, , \overline{f}_p] \tag{2.20}$$

In the above equation $\underline{f}_q$ and $\overline{f}_q$represent the following equations:

$$\underline{f}_p(\boldsymbol{x}) = \underline{\mu}_{\tilde{A}_{i,1}}(x_1) \cap \ldots \cap \underline{\mu}_{\tilde{A}_{i,M}}(x_M) \tag{2.21}$$

$$\overline{f}_p(\boldsymbol{x}) = \overline{\mu}_{\tilde{A}_{i,1}}(x_1) \cap \ldots \cap \overline{\mu}_{\tilde{A}_{i,M}}(x_M) \tag{2.22}$$

Here, $\cap$ denotes the t-norm operator that can be the algebraic product or minimum operator. It should be remarked that rule firings in IT2-FLS have two values: the lower firing value and the upper interval value. In the thesis, we define UMF with a baseline T1-FS and the LMFs $\underline{\mu}_{\tilde{A}_{i,m}}$ in terms of the UMF $\overline{\mu}_{\tilde{A}_{i,m}}$ and the height of the LMFs as we employ in different works of ours [62-65]:

$$\overline{\mu}_{\tilde{A}_{i,m}}(x_m) = \mu_{A_{i,m}}(x_m) \tag{2.23}$$

$$\underline{\mu}_{\tilde{A}_{i,m}}(x_m) = h_{i,m}\overline{\mu}_{\tilde{A}_{i,m}}(x_m) \tag{2.24}$$

Here, the FOU design parameter of IT2-FLSs, can be denoted also as $h_{i,m}$, is also known as the parameter associated with the IT2-FLC $\tilde{A}$ in its most general form [24-26], [55-60]. This parameter is responsible for defining the size of the FOU by changing the shaded area [61-65]. When the height parameter $h_{i,m}$ gets close to 1, the FOU area decreases and IT2-FS gets closer to a T1-FS. More specifically, $h_{i,m} = 1$ the IT2-FS is reduced to a T1-FS ($\overline{\mu}_{\tilde{A}_{i,m}} = \underline{\mu}_{\tilde{A}_{i,m}}$) and on the other hand as the $h_{i,m}$ gets close to zero FOU area increases. These flexibility gives IT2-FLSs a powerful ability to handle unknowns or uncertainties.

To sum up, firings are represented as an interval, making it impossible to compute the interval directly through the defuzzification process. In this context, the crisp output value of an IT2-FLS can be calculated as follows:

$$y_{IT2} = \frac{(\underline{y}(x) + \overline{y}(x))}{2} \tag{2.25}$$

Here, $\underline{y}(\boldsymbol{x})$ and $\overline{y}(x)$ are left and right endpoints of the type-reduced set and as we mentioned earlier the most common used methods to calculate these points are KM [61], NT [51] and also a method that directly uses the endpoints of the type-reduced set namely Simple Method (SM) and they are formulized as follows:

- SM method [49] uses the boundary FLSs to calculate the type reduced set as follows:

$$\underline{y}(\boldsymbol{x}) = \frac{\sum_{p=1}^{P} \underline{f}_p(\boldsymbol{x}) \underline{y}_p}{\sum_{p=1}^{p} \underline{f}_p(\boldsymbol{x})} \quad \overline{y}(\boldsymbol{x}) = \frac{\sum_{p=1}^{P} \overline{f}_p(\boldsymbol{x}) \overline{y}_p}{\sum_{p=1}^{P} \overline{f}_p(\boldsymbol{x})} \tag{2.26}$$

- KM method [61] calculates the type reduced set as a mixture of $\underline{f}_p$ and $\overline{f}_p$ via following formulation:

$$\underline{y}(\boldsymbol{x}) = \frac{\sum_{p=1}^{L} \overline{f}_p(\boldsymbol{x}) \underline{y}_p + \sum_{p=L+1}^{P} \underline{f}_p(\boldsymbol{x}) \underline{y}_p}{\sum_{p=1}^{L} \overline{f}_p(\boldsymbol{x}) + \sum_{p=L+1}^{P} \underline{f}_p(\boldsymbol{x})} \tag{2.27}$$

$$\overline{y}(\boldsymbol{x}) = \frac{\sum_{p=1}^{R} \underline{f}_p(\boldsymbol{x}) \overline{y}_p + \sum_{r=R+1}^{P} \overline{f}_p(\boldsymbol{x}) \overline{y}_p}{\sum_{p=1}^{R} \underline{f}_p(\boldsymbol{x}) + \sum_{p=R+1}^{P} \overline{f}_p(\boldsymbol{x})} \tag{2.28}$$

  where L and R are left and right switching points and found by the KM algorithm iteratively.

- The NT method calculates the output directly as:

$$y_{IT2}(\boldsymbol{x}) = \frac{\sum_{p=1}^{P} [\underline{f}_p(\boldsymbol{x}) \underline{y}_p + \overline{f}_p(\boldsymbol{x}) \overline{y}_p]}{\sum_{p=1}^{P} \underline{f}_p(\boldsymbol{x}) + \sum_{p=1}^{P} \overline{f}_p(\boldsymbol{x})} \tag{2.29}$$

  In this thesis, we define the following functions to define an equivalent type reduced set:

$$\underline{y}(\boldsymbol{x}) = \frac{2 \sum_{p=1}^{P} \underline{f}_p(\boldsymbol{x}) \underline{y}_p}{\sum_{p=1}^{P} \underline{f}_p(\boldsymbol{x}) + \sum_{p=1}^{P} \overline{f}_p(\boldsymbol{x})} \tag{2.30}$$

$$\overline{y}(\boldsymbol{x}) = \frac{2 \sum_{p=1}^{P} \overline{f}_p(\boldsymbol{x}) \overline{y}_p}{\sum_{p=1}^{P} \underline{f}_p(\boldsymbol{x}) + \sum_{p=1}^{P} \overline{f}_p(\boldsymbol{x})} \tag{2.31}$$

2.2.2 General type-2 fuzzy sets and systems

The GT2-FS represents the most general form of T2-FSs, and is also denoted as ($\tilde{A}$). Its definition is given by equation 2.12. The GT2-FS's primary and/or secondary membership grades are determined based on the primary membership of x, denoted as J_x, and the SMF ($\mu_{\tilde{A}(x)}(u)$). The SMF and primary membership are defined in a similar way as in the case of IT2-FS. In Figures 2.7 and 2.8 illustrations of the triangle GT2-MF and Gaussian type GT2-MF are shown, respectively.

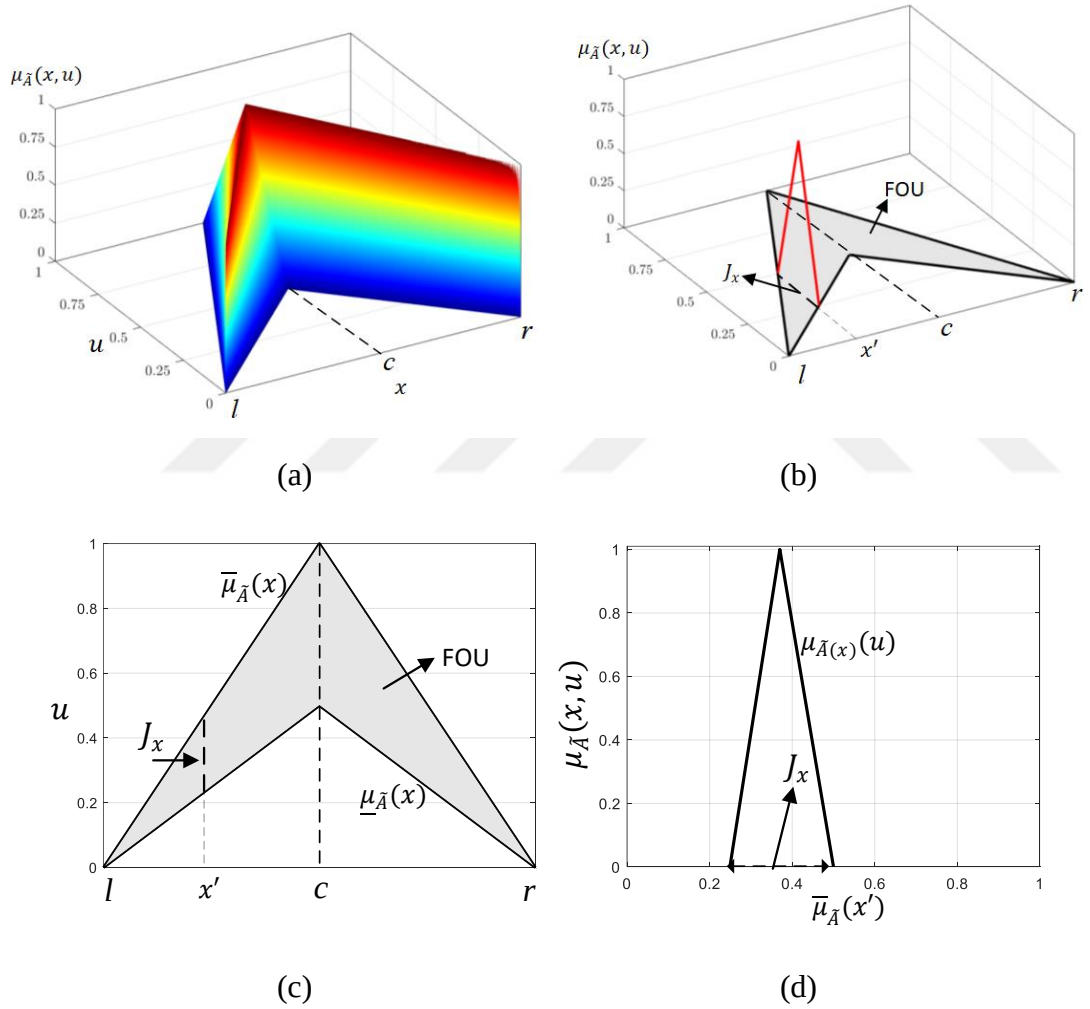


Figure 2.7 : Illustration of a triangle GT2-MF (a) 3D GT2-FS, (b) 3D GT2-FS for a unique x' (C) Primary MF, (d) Secondary MF.

The GT2-FSs has much more sophisticated visual properties than their T1 and IT2 equivalents, as demonstrated in Figures 2.7 and 2.8. In the Figures 2.7a and 2.8a, the common GT2-FSs (triangle type GT2 and Gaussian type GT2) are presented in 3D. Here, the x-axis is the primary input variable (x), y-axis represents the secondary

variable (u) and the z-axis is the secondary membership grade ($\mu_{\tilde{A}}(x,u)$). The Figures 2.7b and 2.8b illustrates 3D representation of the triangle type GT2 and Gaussian type GT2-FSs for a unique primary input (x'). In these figures, the SMFs for the individual primary input (x') is drawn by the red.

Remark: It should be remark that for a Gaussian type GT2-FS the SMFs can be selected different from a Gaussian T1-FS such as a triangular or trapezoidal T1-FS.

It should be noted that the collections of all the individual SMFs (T1-FSs) corresponding to all x values resulted with same the illustrations as given in Figures 2.7a and 2.8a. Moreover, in the figures 2.7c, 2.7d, 2.8c and 2.8d, the separate illustration of the primary and secondary MFs are demonstrated. Here, the FOU, LMF, and UMF are can be seen shown in Figures 2.7c and 2.8c related to their triangle, and Gaussian T2-FSs, respectively. Besides, in Figures 2.7d and 2.8d, for a unique primary input (x'), the resulting SMF support (J_x) and corresponding triangle and Gaussian SMF ($\mu_{\tilde{A}(x)}(u)$) can be seen.

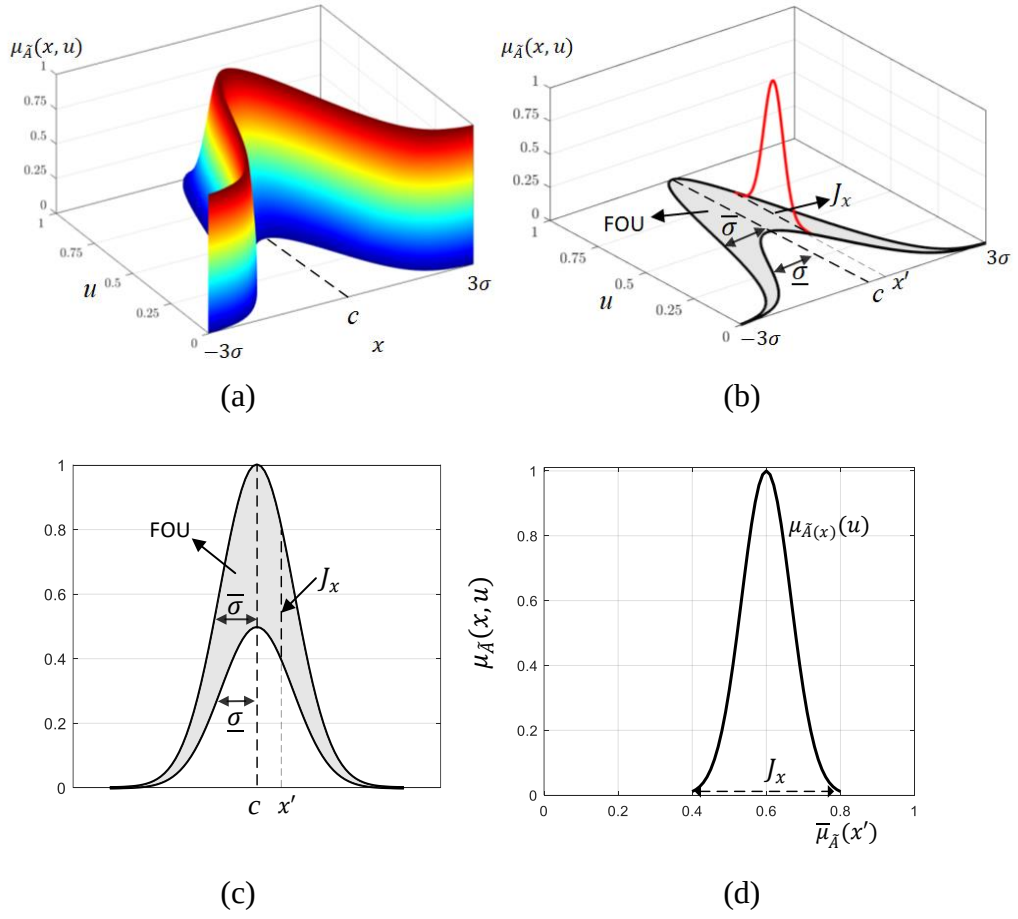


Figure 2.8 : Illustration of a Gaussian GT2-MF (a) 3D GT2-FS, (b) 3D GT2-FS for a unique x' (C) Primary MF, (d) Secondary MF.

Figure 2.9 shows the differences between T1, IT2 and GT2 in their SMF. In the case of T1-MFs, there is no FOU since we can see a T1-FS as their LMF ($\underline{\mu}_{\tilde{A}}(x)$) and UMF ($\overline{\mu}_{\tilde{A}}(x)$) is same and height of the LMF is set to one. Thus, this situation results with a crisp SMF as seen in Figure 2.9a.

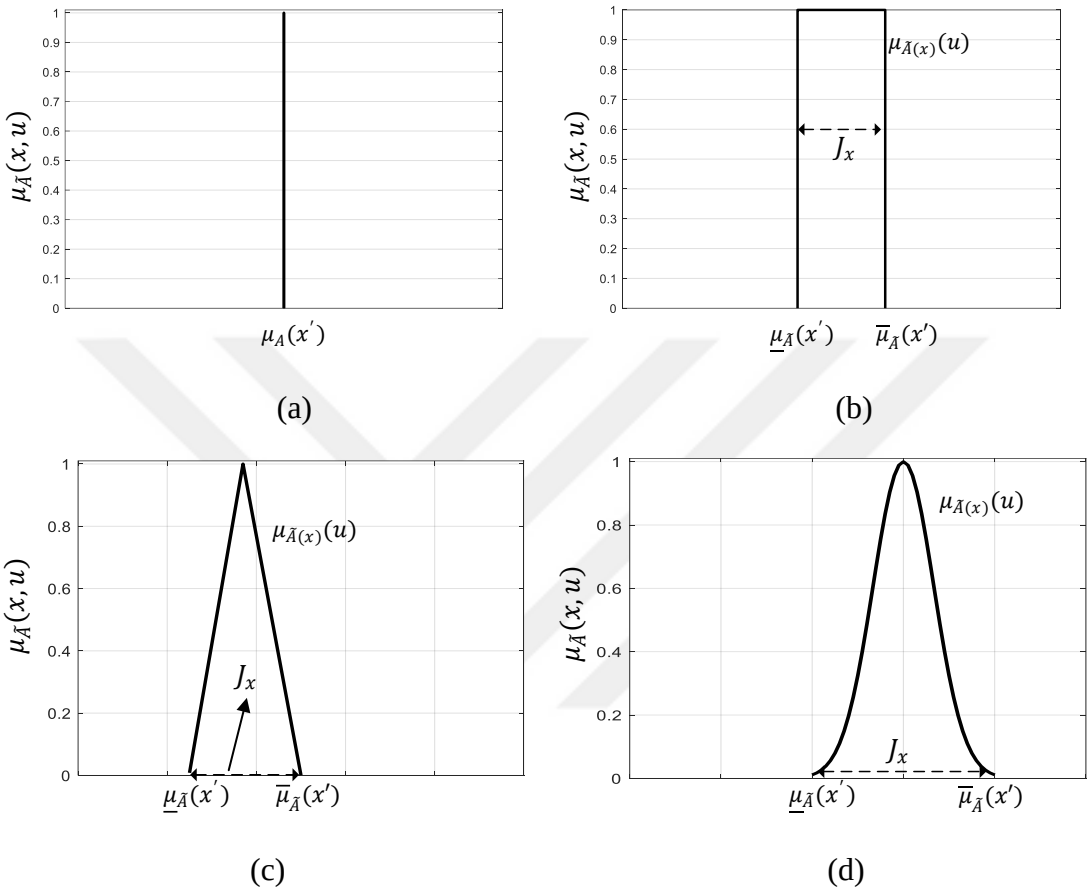


Figure 2.9 : Examples of the SMFs (a) a crisp FS, (b) IV-FS, (c) a triangle type T1-FS, (d) a Gaussian type T1-FS.

On the other hand, IT2-FSs have a predefined interval due to their FOU bounded by the LMF ($\underline{\mu}_{\tilde{A}}(x)$) and UMF ($\overline{\mu}_{\tilde{A}}(x)$). The height ($h$) of the LMF ($\underline{\mu}_{\tilde{A}}(x)$) determines the SMF support (J_x) width as height approaches 1 the width of the J_x approaches to the 0, on the other hand as the height goes to 0 the width of the J_x goes to 1. This provides additional design flexibility, and the SMF design is not feasible since all secondary grades are 1, as illustrated in Figure 2.9b. In contrast, in the case of GT2-FS, there are several possible SMF available since the SMF is created based on a T1-FS. It should be noted that Figure 2.9c and 2.9d illustrates only the triangular and Gaussian examples. SMFs add another flexibility to GT2-FSs compared to IT2 counterpart. Therefore, GT2-FLSs are more capable tools in handling uncertainties

that T1 and IT2 counterparts. However, they are more complex models and, in the application, their computational costs much bigger than the T1 and IT2 counterparts.

There are several ways to represent the GT2-FSs, the most commonly used one, in literature, merging all (horizontal) slices over the α levels (or z-slices) at the secondary layer [24]. The representation of collecting all points is the fundamental definition for all other representations and can be defined as:

$$\tilde{A} = \int_{x \in X} \mu_{\tilde{A}(x)}/x\,, \qquad \mu_{\tilde{A}(x)} = \mu_{\tilde{A}(x)}(u) = \int_{\underline{\mu}_{\tilde{A}}(x)}^{\overline{\mu}_{\tilde{A}}(x)} (f_x(u))/u \tag{2.32}$$

The $f_x(u)$ represents the secondary grade of the input variable x. The secondary grade can also be established using the employed T1-FSs, which describe SMFs defined within the range of LMF and UMF values, as follows:

$$f_x(u) = A_x, \qquad \tilde{A} = \{(u, \mu_{A_x}(u)) \mid u \in J_x\} \tag{2.33}$$

Here, A_x represents an individual T1-FS of a SMF.

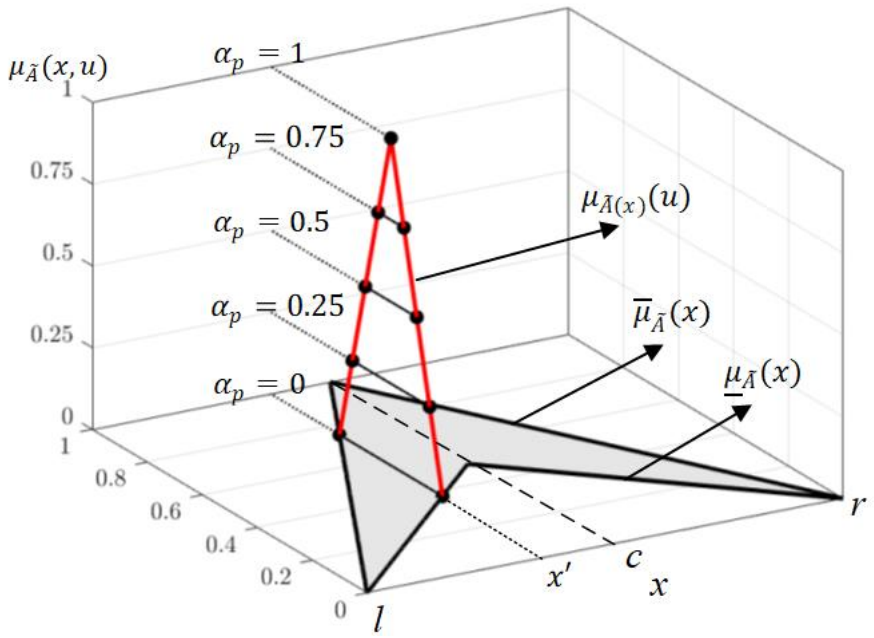


Figure 2.10 : Example of an α-plane of triangle GT2-FS employing triangle SMF.

The most widely used form of this representation involves combining all horizontal slices, which is also referred to as the α-plane or z-slice representation. Despite variations in their definitions, the outputs obtained from both the α-plane and z-slice

representations are identical, as confirmed by multiple sources [66-68]. In the thesis, we prefer to call α-plane representation to refer the horizontal slices of GT2-FSs. In Figure 2.10, the explanatory demonstration of the α-plane representation is illustrated for a triangle type GT2-FS.

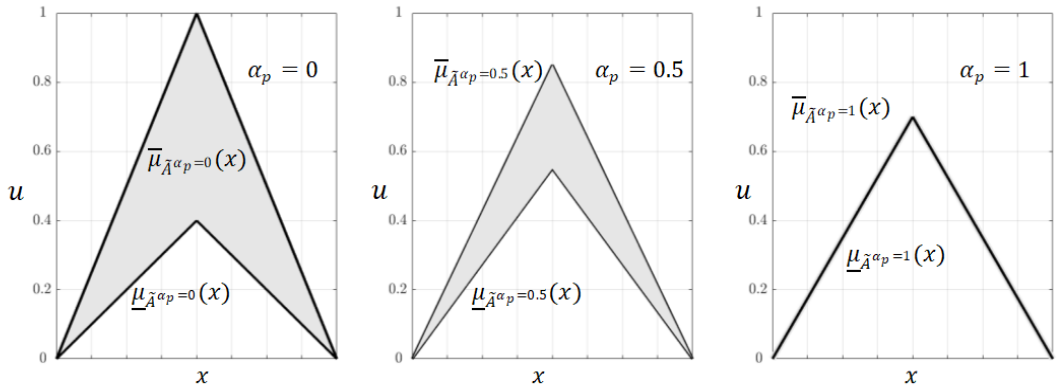


Figure 2.11 : Example of the FOU and Shape change with respect to different α-plane levels.

By inspiring the similarity between the α-plane representation of GT2-FSs and the α-cut representation of T1-FSs, it is possible to derive the α-planes of GT2-FS from the α-cuts of the T1-FS utilized in SMF [24]. In this context, α-plane representation of the GT2-FS using α-cuts of the T1-FS in SMFs can be defined as follows:

$$\tilde{A}^{\alpha_p} = \int_{x \in X} \tilde{A}^{\alpha_p}/x = \int_{x \in X} [\underline{\mu}_{\tilde{A}^{\alpha_p}}(x), \overline{\mu}_{\tilde{A}^{\alpha_p}}(x)]/x \tag{2.34}$$

where $\underline{\mu}_{\tilde{A}^{\alpha_p}}$ and $\overline{\mu}_{\tilde{A}^{\alpha_p}}$ represent the lower and upper MFs of $\tilde{A}^{\alpha_p}$ or in other words α-plane related to the LMF and UMF, respectively. For a triangle SMF, the corresponding LMF and UMF ($\underline{\mu}_{\tilde{A}^{\alpha_p}}$ and $\overline{\mu}_{\tilde{A}^{\alpha_p}}$) of a α-planes of GT2-FLS in the primary MF can be defined as follows [69]:

$$\underline{\mu}_{\tilde{A}^{\alpha_p}} = \underline{\mu}_{\tilde{A}} + w\left(\overline{\mu}_{\tilde{A}} - \underline{\mu}_{\tilde{A}}\right)\alpha_p \tag{2.35}$$

$$\overline{\mu}_{\tilde{A}^{\alpha_p}} = \overline{\mu}_{\tilde{A}} - (1-w)\left(\overline{\mu}_{\tilde{A}} - \underline{\mu}_{\tilde{A}}\right)\alpha_p \tag{2.36}$$

Here, w is the apex point of the triangle SMF. This equation indicates that we end up with unnormal IT2-FS for different levels of α_p. Therefore, we obtain primary MFs for different levels of α_p such that $\overline{\mu}_{\tilde{A}^{\alpha_p}}(x) < 1$.

In Figure Figures 2.10, we provide the α-plane representations of a triangle GT2-FS and selecting triangle SMFs. The SMF slice is shown for a unique input value x' which is shown on the FOU at the α-plane level $\alpha_p = 0$. The SMF slice is marked as red, and we show the α-plane levels with 5 α-cuts, similar to T1 counterpart, $\alpha_p = \{0, 0.25, 0.5, 0.75, 1\}$ and illustrated them in Figure 2.10. Similar to the support of an α-cut for T1-FS, which is determined by the α-cut level and the base support of the FS, the support of an α-plane for GT2-FS varies based on the α-plane level and the primary membership. As a result, the size and shape of the FOU of primary MF is changed for each α-plane level, as illustrated in Figure 2.11.

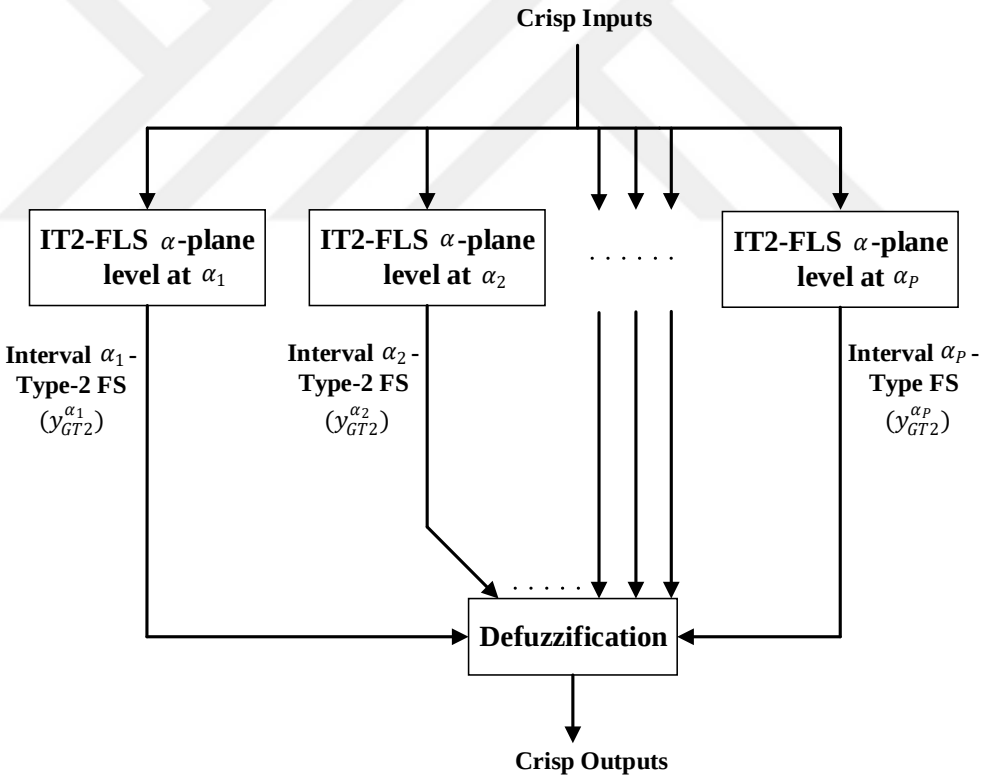


Figure 2.12 : The internal structure of the α-plane based GT2-FLS.

A fuzzy rule of a GT2-FLS is defined as follows [24]:

$$R_p : \text{If } x_1 \text{ is } \tilde{A}_{i,1} \text{ and } \ldots \text{and } x_m \text{ is } \tilde{A}_{i,m} \text{ , Then } y_{GT2}^1 \text{ is } \tilde{Y}_p^1 \text{ and } \ldots y_{GT2}^T \text{ is } \tilde{Y}_p^T \tag{2.37}$$

Here, $\tilde{Y}_p^t = [\underline{y}_p^t, \overline{y}_p^t]$ are consequent MFs that are defined as:

$$\underline{y}_p^t = \sum_{m=1}^{M} a_{p,m} x_m + a_{p,0} - \sum_{m=1}^{M} b_{p,m} |x_m| - b_{p,0}$$
$$\overline{y}_p^t = \sum_{m=1}^{M} a_{p,m} x_m + a_{p,0} + \sum_{m=1}^{M} b_{p,m} |x_m| + b_{p,0} \quad (2.38)$$

For simplicity we drop the t defined for $\tilde{Y}_p^t = [\underline{y}_p^t, \overline{y}_p^t]$, y_{GT2}^t and we use $\tilde{Y}_p^t = [\underline{y}_p, \overline{y}_p]$, y_{GT2} to point out for the one particular output of the IT2-FLS, all operations are same for the other outputs. In this rule structure, R_p is the pth rule of this rule definition, $(p \in \{1,2,\dots,P\})$, where the number of all rules is represented with P. Then, the inputs are symbolized with x_m, $(m \in \{1,2,\dots,M\})$ and total input number is symbolized with M, while the output of the IT2-FLS, which is defuzzified and represented in crisp form, is denoted as y_{GT2}. Moreover, the GT2 antecedent Membership Functions (MFs) are defined with $\tilde{A}_{i,m}$ $(i \in \{1,2,\dots,I\})$ and the total number of antecedent MFs are shown with I. Besides, T is the total number of outputs of the IT2-FLS. y_{GT2}^t is the defuzzified crisp output of IT2-FLS for t. Consequents MFs can be selected as a crisp number, as we set $a_{p,m} = 0$, $b_{p,m} = 0$ and $b_{p,0} = 0$ for $\forall\, p, m$, or we can select it as an linear functions as $y_p = a_{p,m} x_m + a_{p,0}$ by setting $b_{p,m} = 0$ and $b_{p,0} = 0$. Thus, $\tilde{Y}_p$ is the membership function of the consequent for the rule R_p.

The process of calculating an α-plane output for GT2-FLS is very similar to the process used to calculate an IT2-FLS output. In GT2-FLS, as it is illustrated in Figure 2.12, each α-plane represents an IT2-FLS at the associated α-plane level and the height of the secondary membership is denoted by α_k. The computations for each IT2-FLS are based on the associated firing levels of the α-plane. IT2-FLS at different α-plane levels have the same main operations, fuzzifier, rules, inference, type-reducer, and defuzzifier, as the IT2-FLSs demonstrated in Figure 2.6. Here, the only difference is that IT2-FLSs at different α-plane levels have changing membership degrees and firings with respect to that α-plane levels. Therefore, we end up with a new interval set in the related IT2-FS which is shaped by the α-plane associated LMF and UMF and they are symbolized as $\underline{\mu}_{\tilde{A}}^{\alpha}$ and $\overline{\mu}_{\tilde{A}}^{\alpha}$.

Using the representation of α-plane representation or it is also called horizontal slice, GT2-FLS (y_{GT2}) outputs can be expressed as a weighted average of the outputs of α-plane related T2-FLCs across the α-planes. This definition is outlined in [24] as follows:

$$y_{GT2} = \frac{\sum_{k=1}^{K} y_{GT2}^{\alpha_k} \alpha_k}{\sum_{K=1}^{K} \alpha_k} \tag{2.39}$$

Here, α_k indicates the level of α-plane, $y_{GT2}^{\alpha_k}$ output of the IT2-FLS at the level of α_k plane and k is an index for the corresponding α-plane. K stands for the total number of α-planes available (where k ranges from 1 to K). It is important to note that as the number of α-planes increases, the α-axis becomes more finely sampled, resulting in a more evenly spaced set of outputs. However, this also leads to an increase in computational complexity and longer computation time due to the higher number of α-plane samples.

3. LEARNING METHODS FOR FUZZY LOGIC SYSTEMS

In this particular section of the thesis, our main focus will be on presenting a comprehensive overview of the parameter learning techniques utilized for T1/T2-FLSs. For this purpose, we will provide a brief overview on the recent advances on learning methods for the T1/T2-FLSs. Then, we give some information about the commonly used deep learning layers and parameter learning methods. After that we introduce our proposed novel deep learning based FLSs.

3.1 Recent Advances on Learning of T1/T2 Fuzzy Logic Systems

Researchers have steered their attention to soft computing techniques for various problems like control, modelling, regression classification problems, etc. since the concerns of the conventional methods on computation time, accuracy and design complexity. The rule of thumb two soft computing methods are the neural networks and FLSs. The idea of fusing these two techniques is becoming increasingly popular in various scientific and engineering fields as a means to solve real-world problems [29]. FLSs can enhance the reasoning and inference of a learning machine, allowing for the qualitative, albeit imprecise, knowledge to be modeled and symbolically expressed using their linguistic FSs. Neural networks, on the other hand, provide parallelism, a better generalization capability and robustness in learning. Thus, combining these two systems increase ability to represent knowledge and provide a systematic learning which makes it a promising tool for various learning problems [29].

The TSK type FLSs are powerful tools against the problems like nonlinear system identification, regression, etc. The reason behind their success FLSs represent the global characteristics of the handled problem using the linear submodels and get the global characteristics by combining them [30]. The most widely used parameter learning method, in literature, for T1-FLSs is known as ANFIS [31, 32] which is actually a five layered network structure. It defines premise and consequent parameters through fuzzy if-then rules that relate the two sets of parameters. However, ANFIS has some disadvantages such as it requires a significant amount of computational power and producing complex models for even simple problems [31].

The parameter learning method for FLSs can be classified in five categories [33]; Gradient Descent Based Methods, Hybrid Learning Based Methods, Population Based Methods, Extreme Learning Machine (ELM) Based Methods and Support Vector Machine (SVM) Based Methods as shown in Figure 3.1.

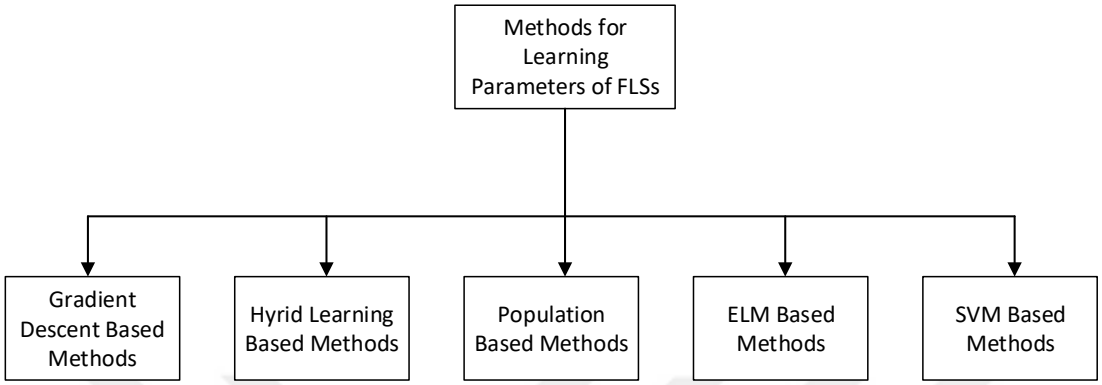


Figure 3.1 : Classification of the methods for learning the parameters of FLSs.

The gradient technique utilizes the steepest descent to determine the optimal solution based on the negative gradient of the objective function, and it relies heavily on back-propagation based learning techniques. The literature includes various approaches that utilize back-propagation to optimize the antecedent and consequent parameters of FLSs. For instance, in [70], the antecedent parameters are learned via a back-propagation algorithm while the consequent parameters are constants. In [71], the authors propose a modified back-propagation algorithm that incorporates adaptive learning rate to train the parameters. In [72], the conventional error function setting and learning formula of the gradient descent algorithm for a TSK type FLS are changed by reciprocating the widths of the corresponding Gaussian membership functions. In [73], a fuzzy-neuro system is proposed that employs local learning and fine-tunes the transformed consequent parameters using the gradient descent method. However, using gradient descent-based learning methods may be slow, inefficient, and prone to getting stuck in a local minimum, as indicated by these studies [74]. Therefore, better learning performance may require multiple iterative learning steps. Using gradient descent alone for training may result in weak firing strength, similar to the vanishing gradient problem observed in DNNs. Moreover, setting fuzzy rules initially, especially for large training data, can be challenging. Additionally, gradient-based techniques may be insensitive to long-term time dependencies, which can cause stability issues. [75].

Another technique to learn the parameters of the FLSs is the Hybrid methods. Solely Backpropagation might suffer against the problem that has a huge amount of input feature and samples. Moreover, it might not be able to learn the complex models that has huge amount of design parameters [33]. In order to overcome these issues hybrid learning based methods have been developed which utilize multiple learning techniques to efficiently train parameters and achieve stable and fast convergence. The majority of the hybrid methods is based on the self-organizing neuro-fuzzy systems. They aim to adapting changing conditions and achieve better performance in real-world applications by combining the parameter and structural tuning. In [76], authors proposed the self-constructing neural fuzzy inference network (SONFIN) which uses an online adaptation algorithm to update its rules as it learns from new data unlike other systems that have a fixed set of rules. The authors of the [77] have developed a hybrid learning algorithm that combines two different techniques, namely the recursive gradient descent method and a least square estimator which uses the singular value decomposition. The goal of this hybrid approach is to efficiently estimate the parameters in a system. Another hybrid method is proposed in [78] that uses the evolving approach for learning of the FLSs which modifies network structure based on evolving criteria in various aspects. By using these evolving criteria, the network can adapt and change over time to better fit the input data and provide more accurate predictions.

The other commonly used approach to learn the parameters of FLSs in literature is the population based methods. The main problem of the methods like backpropagation is the learning process could have stuck in a local minima and the model may perform overfitting. Population-based algorithms are more effective when derivatives are difficult or impossible to obtain. The population-based techniques which are proposed to tune the parameters of the FLSs can be categorized as follows; genetic algorithms (GA) [79], particle swarm optimization (PSO) [80] and artificial bee colony (ABC) [81]. We can also include techniques such as, ant colony optimization (ACO) [82] and differential evolution [83].

The Extreme Learning Machine (ELM) is a relatively new neural network method that has gained popularity due to its ability to address concerns about computational time and generalization compared to conventional NNs. The idea of combining the ELM and FLSs can improve the accuracy and reduce the complexity of the learning process.

In this approach, traditional neural networks are replaced by ELM, resulting in more flexibility in the networks. In the referenced [84-87] works, nice integration of the ELM with FLSs can be found.

FLSs that is combined with Support Vector Machines (SVMs) is a relatively new technique which utilizes the values of the fuzzy MFs to each input point and reformulates the mathematics of the SVM. This approach allows for different input points to make different contributions to the decision surface based on their respective fuzzy memberships. In literature, various works can be found used SVM and FLSs in different ways such as the work [87] integrates SVM with FLSs by using adaptive kernel functions, in [88] a fuzzy logic based SVM is developed which uses fuzzy clustering methods in order to reduce the amount of training data and the computation time for constrained optimization. The work [89] uses fuzzy SVM for a fault diagnosis application and optimizes the model parameters with PSO algorithm. The problem of the all methods mentioned above the number of the fuzzy rules increasing with the number of the support vectors. Therefore, for the complex problem we end up with very complex models.

The usage of the FLSs in handling uncertainty and also defining nonlinear input-output mappings has been studied extensively, in the literature [90, 91], because their success against the unknowns. IT2-FLSs have gained increasing attention in recent years due to their ability to handle uncertainties and define nonlinear input-output mappings that are not achievable by type-1 FLSs [25]. Applications in which uncertainties and complexities are high, such as control system design [25], [26], pursuing games [92], decision making [93] and time-series prediction [94], IT2- FLSs have found to be particularly useful.

IT2-FLSs take their power from IT2-FSs in which they employed in the antecedent parts of the IT2-FLSs, which gives an additional degree of freedom provided by the FOU and thus improve their performance against the uncertainties [25]. IT2-FLSs have a structure similar to their type-1 counterpart, with the only difference being an additional type reduction (TR) mechanism since IT2-FSs are used in IT2-FLSs [24, 25]. However, the design complexity of IT2-FLSs is high due to the large number of parameters to be tuned. Learning IT2-FLSs is more challenging compared to Type-1 (T1) FLSs due to their sophisticated design complexity. This complexity arises from the constraints resulting from the definitions for the IT2-FSs, a large number of

learnable parameters, and the complicated Center of Sets Calculation Method (CSCM). Consequently, to simplify the design complexity, the learning problem of IT2-FLSs is typically addressed by blurring pre-trained T1-FLSs and/or utilizing evolutionary or Neural Network (NN) approaches. To address this issue, researchers have employed neural networks, including a self-evolving IT2 neuro-fuzzy system [34], an IT2 neuro-fuzzy system that uses fuzzy clustering and evolutionary algorithms [35], a self-organizing IT2 neuro-fuzzy system [36], and an IT2 neuro-fuzzy system for simultaneous feature selection and system identification [37]. Support vector machines have also been integrated with IT2 neuro-fuzzy systems to reduce structure complexity [38]. However, these methods faced difficulties in learning extensive datasets because of problems such as the curse of dimensionality and an increasing number of parameters that depend on the size of the rules [39, 40].

Recent research has shown successful integration of FLSs and IT2-FLSs with deep learning (DL) [39-43]. A Takagi-Sugeno type deep fuzzy network has been proposed to combine the strengths of FLSs and neural networks [39], and mini-batch gradient descent, regularization, and AdaBound concepts have been extended to Takagi-Sugeno FLSs to solve big data regression problems [40]. An IT2 fuzzy logic-based stacked autoencoder deep neural network has also been proposed to generate explainable artificial intelligence models [41-43].

In conclusion, FLSs and IT2-FLSs have demonstrated their ability to handle uncertainty and define nonlinear input-output mappings in various applications. Although the design complexity of IT2-FLSs is high, researchers have utilized neural networks to address this issue. Recent research has further shown the potential of integrating FLSs and IT2-FLSs with DL, offering promise in generating explainable artificial intelligence models and addressing big data problems.

3.2 Brief Overview on the Commonly used Deep Learning Layers and Parameter Learning Methods

Over the last decade, the works on Deep Learning (DL) have made significant progress by utilizing the developed novel learning methods and new defined structures of layers, as well as utilizing graphics cards as processing units [1-5]. This has enabled training of neural network structures with numerous hidden layers and neurons, resulting in a large number of design parameters that can be optimized. As a result,

Deep Learning has made remarkable breakthroughs in various areas such as semantic segmentation, object detection and classification, and handwritten character detection [6,7]. Deep neural networks (DNNs) are highly effective in various applications of machine learning and artificial intelligence. A DNN is actually a type of neural network which consists of an input layer, numbers of hidden layers, and an output layer which is illustrated in Figure 3.2.

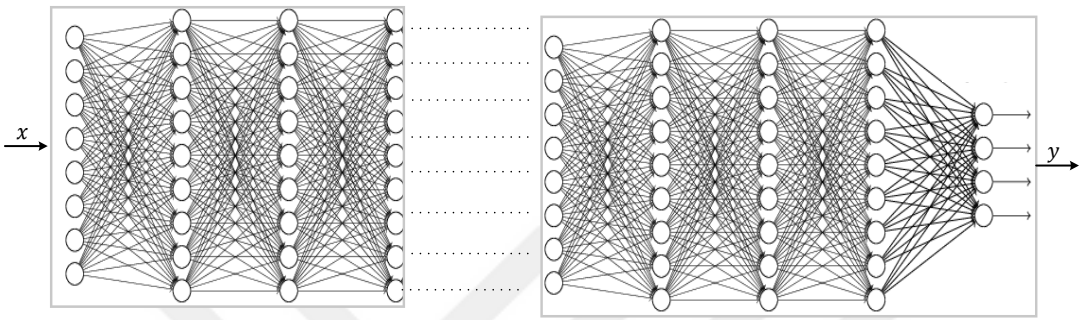


Figure 3.2 : An illustration of a generic DNN structure.

If the number of hidden layers is three or more, the system is referred to as a DNN. In Figure 3.2 a generic DNN structure is shown. Here $\boldsymbol{x} = [x_1, x_2, \dots, x_s]$ is the input vector and $\boldsymbol{y} = [y_1, y_2, \dots, y_o]$ is the output of the DNN structure. When dealing with regression analysis or classification problems, the constructed DNN structure can be formed as a linear estimation function such as $y = w^T x + b$. On the other hand, for more complex problems, DNN can be formed as a nonlinear global approximator using different activation functions. DNNs have brought significant advantages compared to traditional machine learning techniques [1]. The reason behind this improvement is that because they can learn the features at various levels of layers. This feature allows them to obtain the sophisticated features of the complex problems [1]. DNNs initially compute their internal parameters through a forward pass and then refine them iteratively during backpropagation to extract input data features effectively. These capabilities make DNNs superior to other methods in many tasks. As a result, DNNs are now a valuable and powerful tool in various commercial and industrial applications. Deep CNNs are commonly used for image and video recognition tasks due to their ability to learn and extract features from raw data [8].

Commonly used DNN structures can be categorized with respect to the handled problem. In other words, if we deal with an unsupervised learning problem then the commonly used DNN structures are deep autoencoders, recurrent DNNs, and

restricted Boltzmann machines [9-11]. In Figure 3.3, the restricted Boltzmann machines and autoencoders are illustrated. Deep autoencoders are another type of DNN that can be used for dimensionality reduction and feature extraction. Recurrent DNNs are ideal for processing sequential data, such as natural language processing and speech recognition, while restricted Boltzmann machines are used for unsupervised learning and feature extraction. On the other hand, if the problem is a supervised learning problem then the most widely used Deep Neural Networks include is the Convolutional Neural Networks (CNNs) [8], [12].

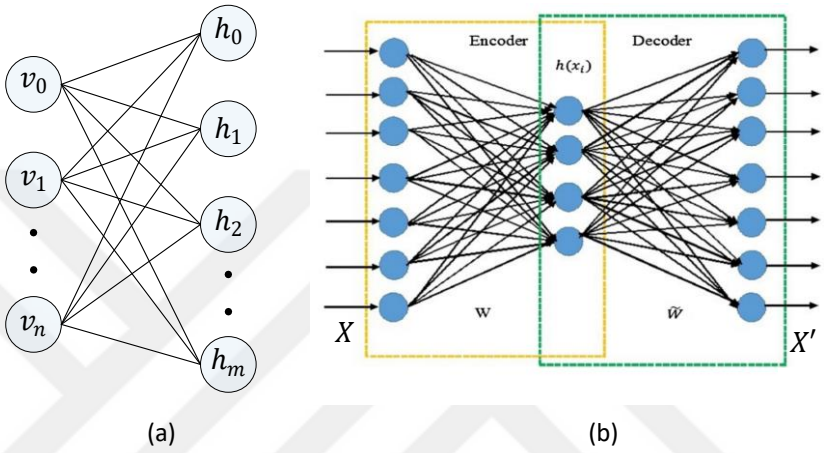


Figure 3.3 : Illustration of the DNNs used for unsupervised learning (a) Restricted Bolztman Machines, (b) Autoencoders.

Different from the regular neural networks, the neurons in the layers of a CNN are organized in three dimensions: width, height, and depth which is demonstrated in Figure 3.4. In this particular example, CNN have an input which has a dimension of 32x32x3. This implies that the input image has 32px width, 32px height and 3 represent the depth of the input and it generally refers to the 3 color channels of the R,G,B images.

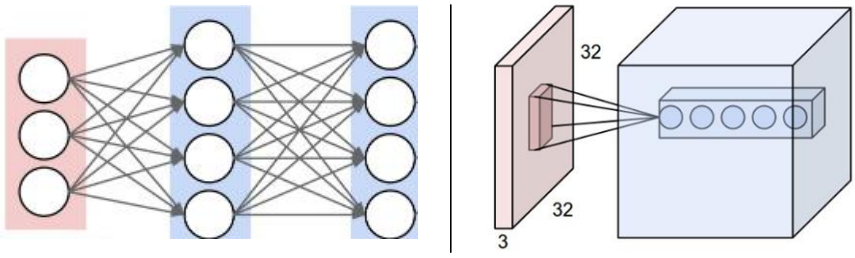


Figure 3.4 : An example of a regular neural network shown in left, an example of a Convolutional Neural Network shown in right.

In Figure 3.5, an example structure of a deep CNN structure is illustrated. As we illustrated in Figure 3.5, a CNN is comprised of a sequence of layers where each layer uses an activation function, in particular to this case a Rectified Linear Unit (ReLU) is demonstrated, transform one volume of activations into another. In the general architecture of a CNN, we see three primary four of layers: the Convolutional Layer (Con), Batch Normalization (BN) layer the Pooling Layer (max pooling is demonstrated for this case), and the Fully-Connected (FC) Layer. These layers are stacked on top of each other to create a complete deep CNN structure.

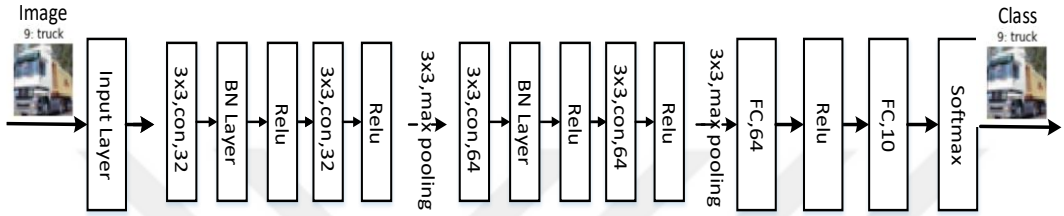


Figure 3.5 : An example of a deep CNN for a classification example.

For the example structure we shown in Figure 3.5, we go through more details below, yet the basic structure can be constructed as [Input Layer – BN-Layer – Con – ReLU – Max Pooling – FC]. The details about these layers can be summarized as [12]:

- INPUT layer [32x32x3] will hold the raw pixel values of the image, in this case, the image is composed of a 32px width, 32px height and 3 color channels R,G,B. Here, a 'zero-center' normalization is applied to the image.
- BN Layer helps to accelerate and stabilize the training of DNNs. It applies a normalization to the inputs of the layers through re-centering and re-scaling. It is also known as the standardization.
- The Con layer will calculate the outputs of neurons that are linked to localized regions in the input. Each neuron computes a dot product between its weights and a small region of the input volume to which it is connected. In this case of first stack convolution filter is selected as [3x3x32] which indicates a filter 3x3 kernel size and 32 filters in total.
- ReLU layer applies an element-wise activation function, such as the $max(0, x)$ function that sets all negative values to zero. This operation preserves the size of the volume, resulting in an output volume of the same

size as the input volume. For the case of the first stack the size of [3x3x32] is preserved between the Con layers after the ReLU layer.

- POOL layer will perform a downsampling operation along the width and height.
- FC layer computes the class scores, in this case it results with a size [1x1x10]. Each of the 10 numbers corresponds to a class score for one of the 10 categories. Similar to conventional neural nets, every neuron in this layer is connected to all other neurons in the previous layer.

In the Convolution layers the other process we should discuss about is the spatial arrangement [8]. The Convolutional Layers have three structural hyperparameters to be determined before the training which are the depth, stride and zero-padding.

- The depth of the output volume is a hyperparameter that determines the number of filters used in the Convolutional Layer. Each filter is trained to detect a unique feature in the input.
- Another hyperparameter is that the stride which determines how the filter move across the input image. When the stride is selected as 1 then the filter slides across the image one pixel at time. While if the stride is selected as 2 or more that causes the filter jumps 2 pixels at one time. Thus, using a larger stride will result in a smaller output dimension.
- In some cases, we may want to pad the input image with zeros along the edges, which is another hyperparameter. This zero-padding enables us to regulate the spatial size of the output image. The advantage of zero-padding is that it allows us to preserve the spatial dimensions of the input image while sliding the filter over it.

In a nut shell, we can compute the width output size of a convolutional layer using the following formulization [8]:

$$output\ size\ of\ the\ width = \frac{W - F + 2P}{S} + 1 \tag{3.1}$$

Here, W stands for the width of the input image to the convolutional layer, F is the kernel size of the convolution filter, P represents the amount of zero-padding and S is the number of the selected stride. Similarly, we can formulate the height output size of

the convolutional layer and they can be summarized for an input that has a dimension of $[W_i \text{ x } H_i \text{ x } D_i]$ then the output size $[W_o \text{ x } H_o \text{ x } D_o]$ of the convolutional layer can be calculated as:

$$W_o = \frac{W_i - F + 2P}{S} + 1$$
$$H_o = \frac{H_i - F + 2P}{S} + 1 \qquad (3.2)$$
$$D_o = K$$

Here K is the number of the convolutional filters.

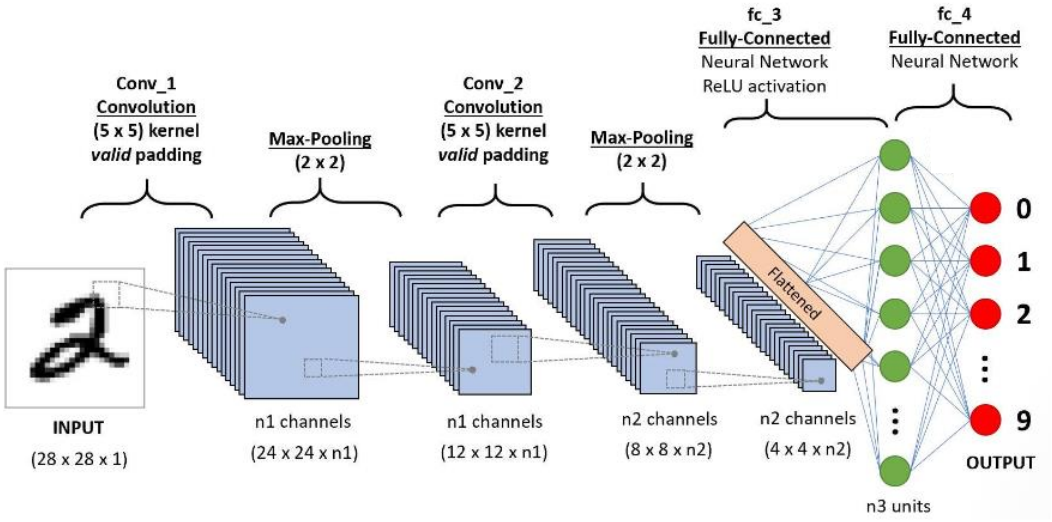


Figure 3.6 : An illustrative example of a CNN structure for handwritten digit classification [13].

Another layer that affects the dimension of the outputs images is the Pooling layers. The main purpose of the Pooling Layer in a CNN structure is to reduce the spatial size of the input image. In that way, it helps to decrease the number of parameters and computational resources required by the CNN. This can also help to prevent overfitting. The most common type of Pooling Layer uses filters of size 2x2 and applies a stride of 2 to downsample the input. In general, for an input that has a dimension of $[W_i \text{ x } H_i \text{ x } D_i]$ then the output size $[W_o \text{ x } H_o \text{ x } D_o]$ of the Pooling layer can be calculated as follows:

$$W_o = \frac{W_i - F}{S} + 1 \qquad (3.3)$$

$$H_o = \frac{H_i - F}{S} + 1$$

$$D_o = D_i$$

Where, F is the kernel size of the Pooling layer and S is the number of the selected stride for pooling layer.

In Figure 3.6, we illustrated a CNN structure for handwritten digit classification for understanding the dimensions of the resulting outputs after convolution and pooling operations. Here, we focus on Conv_1(5x5) layer and first Max-Pooling(2x2) layer.

- The input image has a size of $[W_i = 28 \text{ x } H_i = 28 \text{ x } D_i = 1]$ and we can calculate output of the Conv_1 layer using the formulation defined in equation 3.2. Here, kernel size is selected as $F = 5$, padding and stride are selected as $P = 0,\ S = 1$. The number of convolution filter is left with a symbolic number as n_1. Therefore, we can calculate the dimension of the outputs of Con_1 layer as: $W_o = \frac{28-5+2*0}{1} + 1 = 24,\ H_o = \frac{28-5+2*0}{1} + 1 = 24,\ D_o = n_1$.
- The input dimension $[W_i = 24 \text{ x } H_i = 24 \text{ x } D_i = n_1]$ of the first Max-Pooling layer. We can calculate the output dimension of the Max-Pooling layer using the formulations defined in equation 3.3. Here, the kernel size of the Max-Pooling layer is selected as $F = 2$ and the Stride is selected as $S = 2$. In this context, we can calculate the dimension of the outputs of the first Max-Pooling layer as: $W_o = \frac{24-2}{2} + 1 = 12,\ H_o = \frac{24-2}{2} + 1 = 12,\ D_o = D_i = n_1$.

3.2.1 Commonly used parameter learning methods in deep learning

Many deep learning algorithms aim to optimize an objective function. The derivative-based gradient descent method, which is well known and widely used in the literature, is the most popular one [14]. According to this algorithm, if objective proceeds in the negative slope direction of the function $L(\theta)$, that function can be minimized and the change in the current point can be expressed as follows:

$$\theta_{l+1} = \theta_l - \rho \nabla L(\theta_l) \tag{3.4}$$

Here $\rho > 0$ denotes the learning rate, and $\nabla J(\theta_l)$ denotes the change of the objective function. Many learning methods developed have emerged based on this idea. The commonly used learning methods in this literature are as follows; Stochastic Gradient Descent, Stochastic Gradient Descent with Momentum, Root mean square propagation, and adaptive moment estimation [14-16].

- *Stochastic Gradient Descent* [14]: In the standard gradient descent method, the changes of the objective function ($\nabla L(\theta)$) are calculated using the entire data set to be trained and it is used only once. The Stochastic Gradient Descent method, on the other hand, does not use the entire data set to be trained when calculating the change of the objective function, a smaller sample is selected from this data set, and the change of the objective function is calculated and the parameters are updated according to this change. This selected sample data set is called a mini-batch. At each iteration, the algorithm takes one step to minimize the objective function. The full transition of the entire training set using the mini-batch of the training series is an era. One step of training with the chosen small piece is called the epoch.

- *Stochastic Gradient Descent with Momentum* [15]: The Stochastic Gradient Descent method may experience some oscillations on the way to the optimal point. Adding a momentum term is one of the methods used to reduce oscillation in parameter updating. Probabilistic gradient descent method with momentum updates the parameters as follows:

$$\theta_{l+1} = \theta_l - \rho \nabla L(\theta_l) + \gamma(\theta_l - \theta_{l-1}) \tag{3.5}$$

  Here, γ dictates the extent to which the previous gradient step affects the current iteration. This parameter is also a hyperparameter to be tuned before the training is start.

- *Root mean square propagation* [16]: The stochastic gradient descent method with momentum tries to find all the parameters to be learned using only one learning rate. There are optimization algorithms that aim to enhance parameter training by utilizing learning rates that vary for each parameter and can automatically adjust to the optimized loss function. One such algorithm is

Root-Square Propagation, which computes a moving average of the squares of parameter modifications in the following manner:

$$v_l = \beta v_{l-1} + (1-\beta)[\nabla L(\theta_l)]^2) \tag{3.6}$$

Here β denotes the decay rate of the moving averages. The decay rate is a hyperparameter used in the optimization algorithm for updating the moving average of squared gradients during training in deep learning models. The decay rate is generally selected a value from the interval [0, 1], with common values being 0.9, 0.99, and 0.999. The decay rate determines the weight given to the previous moving average and the current gradient squared in updating the moving average. Specifically, the averaging length of the squared gradients is equal to $1/(1-\beta)$, which results in 10, 100, and 1000 parameter updates for decay rates of 0.9, 0.99, and 0.999, respectively. The root-square spread algorithm performs the parameter update as follows:

$$\theta_{l+1} = \theta_l - \frac{\rho \nabla L(\theta_l)}{\sqrt{v_l} + \varepsilon} \tag{3.7}$$

Here, the term ε is a small number close to zero added to avoid the problem of divided by zero, and the defined operation is performed element-wise. The most important advantage of the root-square propagation algorithm is that the learning rates of parameters with large rate of change are effectively reduced, and parameters with small rate of change can effectively increase the learning rates. In this way, the optimal point is approached faster or exceeding this point is prevented.

- *Adaptive Moment Estimation (Adam)* [95]: The Adaptive Momentum Estimation algorithm is a parameter update method similar to the Square Root Propagation algorithm. Similarly calculates a moving average of both the parameter changes itself and its squares

$$\begin{aligned} m_l &= \beta_1 m_{l-1} + (1-\beta_1)\nabla L(\theta_l) \\ v_l &= \beta_2 v_{l-1} + (1-\beta_2)[\nabla L(\theta_l)]^2 \end{aligned} \tag{3.8}$$

Here, we can determine the β_1 and β_2 similar to the root mean square propagation method. The Adam algorithm performs the parameter update as follows:

$$\theta_{l+1} = \theta_l - \frac{\rho m_l}{\sqrt{v_l} + \varepsilon} \tag{3.9}$$

If there is consistency in the parameter changes across multiple iterations, the Root-Square Propagation algorithm leverages the moving average of the parameter changes to introduce momentum to the parameter updates in a specific direction. Conversely, if the parameter changes are predominantly noise, the moving average of the parameter changes becomes smaller, which, in turn, leads to a reduction in the parameter updates.

3.3 Proposed Deep Learning Based T1/T2 Fuzzy Logic Systems

In this thesis study, we propose a novel approach to learn the parameters of the T1/T2-FLSs. In this approach, we use a mini-batch sampled deep learning parameter learning methods, we introduced these methods in previous section, to learn the parameters of the T1/T2-FLSs. In the novel approach we propose a mini-batch sampled deep learning parameter learning method since the conventional parameter learning methods for T1/T2-FLSs face difficulties in learning extensive datasets because of problems such as the curse of dimensionality and an increasing number of parameters that depend on the size of the rules.

Although T2-FLSs (IT2-FLS or GT2-FLS) are powerful tools for modeling uncertainties and nonlinear expressions in systems, they contain many design parameters since T2-FSs are defined as input membership function. These design parameters can be grouped into two parts as structural and learnable parameters. The Structural design parameters in T2-FLSs can be defined as follows:

- Total number of rules P, total number of inputs M, total number of outputs T
- The shapes of the T2-FSs defined in the antecedent which are illustrated in Figure 2.3 and Figure 2.4.
- In case the input membership functions are defined as GT2-FS, the form of the secondary membership function as shown in Figures 2.7 and 2.8.

- Using wheatear an IT2-FS or a GT2-FS in the antecedent as can be seen from Figure 2.3, 2.4, 2.7 and 2.8.
- In the antecedent, the number of membership functions to be used to define each input.
- In the concluding part, the number of membership functions to be used to define each output.
- Type-Reduction Method (SM, KM or NT) to be used to defuzzify the crisp output as defined in equations 2.26 to 2.31.

In this thesis study, these structural parameters are considered as hyperparameters and they must be predefined and set before the training.

Remark: It should be remarked that the FSs in the antecedent parts can be selected different from a triangle or Gaussian type, yet we only consider these two types of FSs in the scope of this thesis. Moreover, the type-reduction method can be also selected outside of the methods (SM, KM and NT) defined in section 2.2.1. We only consider these 3 type-reduction methods in the scope of this thesis.

The learnable parameters of the T1/T2-FLSs can be listed as follows:

- The learnable parameters in the antecedent part
    - For a triangle type T1/T2-FS, the most general form of the learnable parameters $\boldsymbol{\theta_A} = \{\underline{\boldsymbol{\theta}}, \overline{\boldsymbol{\theta}}\}$. Here, $\underline{\boldsymbol{\theta_A}}_{i,m} \rightarrow \{\underline{l}_{i,m}, \underline{c}_{i,m}, \underline{r}_{i,m}, \underline{h}_{i,m}\}$, $\overline{\boldsymbol{\theta_A}}_{i,m} \rightarrow \{\overline{l}_{i,m}, \overline{c}_{i,m}, \overline{r}_{i,m}, \overline{h}_{i,m}\}$ If we set the parameters as $\underline{h}_{i,m} = \overline{h}_{i,m} = 1$, $\underline{l}_{i,m} = \overline{l}_{i,m}$, $\underline{c}_{i,m} = \overline{c}_{i,m}$ and $\underline{r}_{i,m} = \overline{r}_{i,m}$, the triangle T2-FS reduces to a conventional T1-FS.
    - For a gaussian type T1/T2-FS, the most general form of the learnable parameters $\boldsymbol{\theta_A} = \{\underline{\boldsymbol{\theta}}, \overline{\boldsymbol{\theta}}\}$. Here, $\underline{\boldsymbol{\theta_A}}_{i,m} \rightarrow \{\underline{c}_{i,m}, \underline{\sigma}_{i,m}, \underline{h}_{i,m}\}$ and $\overline{\boldsymbol{\theta_A}}_{i,m} \rightarrow \{\overline{c}_{i,m}, \overline{\sigma}_{i,m}, \overline{h}_{i,m}\}$. If we set the parameters as $\underline{h}_{i,m} = \overline{h}_{i,m} = 1$, $\underline{c}_{i,m} = \overline{c}_{i,m}$, $\underline{\sigma}_{i,m} = \overline{\sigma}_{i,m}$, then the Gaussian type T2-FS reduces to a conventional T1-FS.
- In case the input membership functions are defined as GT2-FSs, the parameters of the secondary membership function and the total number α_k-planes.

- The learnable parameters in the consequent part which can be defined as $\boldsymbol{\theta_C} = \{\boldsymbol{a}, \boldsymbol{a_0}, \boldsymbol{b}, \boldsymbol{b_0}\}$. Here, $\boldsymbol{\theta}_{\boldsymbol{C}_{p,m}} \rightarrow \{a_{p,m}, a_{p,0}, b_{p,m}, b_{p,0}\}$.

As a result, if we try to learn T2-FLSs with T2-FSs (IT2-FS or GT2-FS) in their antecedents for M inputs and Toutputs with P rules have many design parameters. In this context, learning that much parameters naturally presents a major design challenge, although T2-FLSs are powerful tools for modeling uncertainties and nonlinear expressions in complex problems. In this thesis study, for the first time in the literature, the deep learning-based T2-FLS layer structure in Figure 3.7 will be developed.

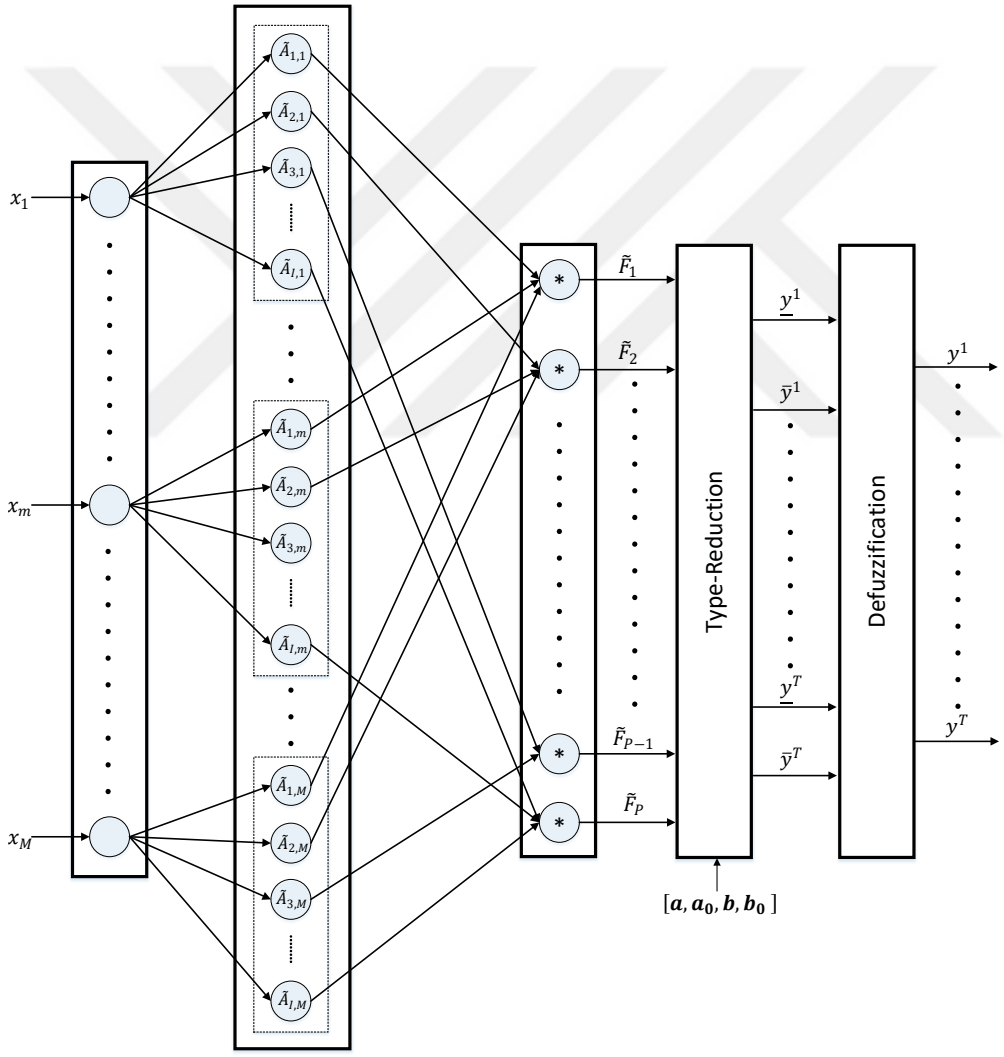


Figure 3.7 : An illustration of the T1/T2 Fuzzy Layer Structure.

The proposed T1/T2 Fuzzy Layer structure within the scope of this thesis study is considered to be composed of 5 layers. These layers are:

- *Layer 1:* It is the input and normalization layer.
- *Layer 2:* The membership values ($\overline{\mu}_{\tilde{A}_{i,m}}(x_m)$ and $\underline{\mu}_{\tilde{A}_{i,m}}(x_m)$), of each crisp value is calculated in this layer. If the antecedent membership functions are defined with GT2-FSs, this process is repeated for each defined α-plane levels $\alpha_k(k = 1 \dots K)$.

  Note that: this layer is also valid for T1-FLSs if we select $\overline{\mu}_{\tilde{A}_{i,m}}(x_m) = \underline{\mu}_{\tilde{A}_{i,m}}(x_m)$ then automatically antecedent MF reduced to a T1-FS.
- *Layer 3:* In this layer, Using the equations 2.10, 2.21 and 2.22 rule firings (f_p or $\tilde{F}_p = [\underline{f}_p, \overline{f}_p]$) are calculated. If the antecedent membership functions are defined with GT2-FSs, this process is repeated for each defined α-plane levels $\alpha_k(k = 1 \dots K)$.
- *Layer 4:* In this layer, using the calculated firings (f_p or $\tilde{F}_p = [\underline{f}_p, \overline{f}_p]$), we calculate the left and right endpoints ($\underline{y}$ and $\overline{y}$) of the type-reduced set using the equations 2.26-2.31. If the antecedent membership functions are defined with GT2-FSs, this process is repeated for each defined α-plane levels $\alpha_k(k = 1 \dots K)$.
- *Layer 5:* In this layer, the crisp outputs are calculated using the end points of the type reduced set ($\underline{y}$ and $\overline{y}$) as defined in equation 2.25 or 2.9 is used if antecedents defined with T1-FSs. If the antecedent membership functions are defined with GT2-FSs, the expression given in equation (2.39) is used by using the end-points of the α-plane outputs ($y_{GT2}^{\alpha_k}$).

In this context, it can be concluded that the proposed structure resembles a deep artificial boundary network (due to the number of neurons). There are learnable parameter sets ($\boldsymbol{\theta}$: parameter vector) in the 2nd and 4th layers of this proposed structure. Through the parameter learning methods commonly used in deep learning described in section 3.2.1, the tuning of these parameters can be provided by the one of the algorithms defined in section 3.2.1.

Remark: It must be noted that the Learnable parameter set ($\boldsymbol{\theta}$) of the proposed structure seen in Figure 3.7 is the parameters of MFs. Thus, in the learning of IT2-FLSs, we have to consider that the learnable parameters are the parameter of the fuzzy MFs. In

this context, the problem is that the parameter space $\boldsymbol{\theta}$ is constrained to $\boldsymbol{\mathcal{C}}$, which presents a problem because deep learning frameworks like Matlab PyTorch or TensorFlow, only provide unconstrained optimization. Thus, we will propose novel parameterization tricks for the parameter set ($\boldsymbol{\theta}$) and explain them in detail in the following sections such that we can use deep learning parameter learning methods with an unconstraint space.

The proposed structure can be implemented in various supervised and unsupervised problems. Also, the structure can be combined with the commonly used deep learning layers such as CNN. In the next sections we will go through in detail of the proposed new architectures, combinations of Fuzzy Layers with CNNs, proposed new composed loss functions for parameter learning of T2-FLSs and employments of these new structures and methods.

3.4 Learning with Type-2 Fuzzy Activation Functions to Improve the Performance of Deep Neural Networks

This subsection presents a detailed introduction to a novel activation function based on IT2-FLS and its architecture, which is designed to enhance the performance of DNNs. To achieve this, we propose a new layer called the IT2 Fuzzy Activation Layer (IT2-FAL), which is comprised of Single input IT2 (SIT2) Fuzzy Rectifying Units (FRUs). In the following subsection, we will provide a thorough explanation of the features, internal characteristics, and learning method for the new IT2-FAL. Additionally, we will present a mathematical formulation for SIT2-FRU and analyze how its parameters influence the input-output characteristics. Based on our analysis of these mappings, we suggest that they can be considered as either hyperparameters or parameters to be learned. Finally, we will evaluate the performance of the SIT2-FRU against other commonly used activation units such as ReLU, PReLU, and ELU, using various benchmark datasets to demonstrate the effectiveness of our proposed IT2-FAL.his subsection,

The major novelties of the proposed structure are:

- We construct the new SIT2-FRU activation function with three design parameters which determine not only the slope in the positive and negative quadrants but also the shape of the input-output mapping of the activation

function. This means that the SIT2-FRU can improve the learning ability of Deep Neural Networks by expressing both linear and complex input-output mappings through adjusting the size of its IT2-FSs.

- The SIT2-FRU can also handle negative input values, which can shift the mean activation of the layer towards zero, similar to ELU and PReLU, leading to an improvement in learning performance. This also helps to mitigate the vanishing gradient problem. Additionally, unlike ReLU, PReLU, and ELU, SIT2-FRU is capable of learning the contribution of each neuron in the positive quadrant. Therefore, using SIT2-FRU can potentially accelerate the convergence speed of Deep Learning models.
- We investigated the robustness of the IT2-FAL concerning the hyperparameters of Deep Learning. Our findings indicate that the learning performance of IT2-FAL is not affected by variations in parameter settings such as the learning rate and Mini Batch Size (MBS). Compared to ReLU, PReLU, and ELU, the DNN with IT2-FAL perform a more robust learning performance.
- We also analyzed the impact of implementing a Batch Normalization (BN) layer on the learning performance of DNN with IT2-FAL. Our comparative results revealed that unlike ReLU, PReLU, and ELU counterparts, SIT2-FRU can maintain satisfactory learning performance regardless of whether BN is used or not.

3.4.1 Brief overview on activation functions

Activation functions are critical components for the deep neural nets since it plays a crucial role in the learning process by since it determines the input-output characteristic of the DNN. Since the discovery of the Rectified Linear Unit (ReLU), which is currently the most widely used activation function, Deep Neural Networks have made significant progress. ReLUs have not only helped to alleviate the vanishing gradient problem, but also to improve the learning performance, as reported in several research studies [11], [96]. Several activation functions have been proposed as extensions of ReLU to enhance the learning performance of Deep Neural Networks. These rules of thumb activation functions can be classified as ReLU and its variations. For example, Leaky ReLU (LReLU) is an activation function that can handle negative

input values by assigning a predefined slope value, as described in previous studies [97]. On the other hand, in Parametric ReLU (PReLU), this slope parameter is learned during the training process of the Deep Neural Network [98]. In addition to LReLU and PReLU, ELU and PELU have been introduced as activation functions for DNNs [99], [100]. ELU and PELU are described as one for positive inputs and $a(e^x - 1)$ for negative ones. They mainly address the problems of bias shift and improve learning speed. Despite their good performance in DNNs, these activation functions have a common weakness which can be defined as their limited ability to generate sophisticated input-output mappings.

As mentioned earlier widely used Activation functions are ReLU and the ones developed based on ReLU. The ReLU can be defined as following [11]:

$$f(x) = \max(0, x) \tag{3.10}$$

Here, x represents the inputs for each neuron and $f(x)$ is the mapping characteristics of the function. The two enhancements of the ReLU is that LReLU [96] and PReLU [97] and their definitions can be made as:

$$f(x) = \max(0, x) + a \min(0, x) \tag{3.11}$$

In the above formulation, a should be selected as $a > 0$. LReLU and PReLU are very similar to each other the only difference is that the parameter a, namely leak parameter, is learned during training in PReLU, while it is predefined for LReLU. This leak parameter (a) helps to avoid PReLU and LReLU saturating negative input to zero. Another activation function which can be seen as a derivative of ReLU is that ELU [99] and can be defined as:

$$f(x) = \max(0, x) + a\ (\mathrm{e}^{\min(0,x)} - 1) \tag{3.12}$$

This activation function gives exponential weights to the inputs on the negative quadrant by using the its hyperparameter (a) and an exponential mapping characteristic defined in equation 3.12. In addition to that activation functions, there are also various other activation functions. However, they have similar characteristics to ReLU and can be seen variants of it. They can be listed as Adaptive Piecewise Linear [4] unit and Maxout Unit [5].

The Adaptive Piecewise Linear (APL) unit aims learning a weighted sum of S parametrized Hinge functions [4]. APL is defined with the following function:

$$f_i(x) = \max(0, x) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s) \tag{3.13}$$

Here, the hyperparameter S is the number of hinges, the variables a_i^s, b_i^s for $i \in 1, \dots, S$ are the parameters of linear functions and learned during training. Even though, APL has the flexibility to represent both convex and non-convex function, it explicitly forces the rightmost segment to have a unit slope and zero bias. This restriction may not be suitable in deep networks and could degrade the learning performance of the network. Moreover, another drawback of APL is that non-differentiable points are increasing linearly with the number of hinges S [4].

Another activation unit is Maxout unit [5] which takes as its inputs the outputs of multiple linear functions and returns the largest as follows:

$$f(x) = \max_{k \in \{1,\dots,K\}} w^k x + b^k \tag{3.14}$$

As stated in [5], Maxout function can approximate any convex function in theory. However, it has not able to learn non-convex functions. Moreover, Maxout units need large memory storage capacity and have considerable high training time since a large number of extra parameters added by the K linear functions for each hidden unit. Therefore, it diminishes the training efficiency of stacked DNNs. In other words, the main disadvantage of Maxout units is that it increases amount of weights to be learned by a factor of K in each layer. Hence, the Maxout units increase computational burden significantly for DNNs.

The adaptive piecewise linear unit can generate convex and non-convex mappings since it can be defined as hinge-shaped functions. This procedure sets a specific rule for the last segment in a series of data points or segments, which requires that it has a slope of 1 and a bias of 0. The constraint imposed by the adaptive piecewise linear unit on the rightmost segment may not be suitable for DNNs and could hinder their learning performance. Another disadvantage is that the number of non-differentiable points increases linearly with the number of hinges. Another variation called Maxout [5] uses the outputs of multiple linear functions and returns the maximum value, but it has

limitations when it comes to learning non-convex functions. Moreover, Maxout units require a large amount of memory storage and can lead to high training times, which may be problematic for stacked DNNs as it can decrease their training efficiency.

3.4.2 Proposed type-2 fuzzy logic based activation layer

In this subsection, we will introduce our proposed IT2 fuzzy logic based activation layer. We explain its internal structure, properties and the construction of the IT2-FAL. Moreover, we will also provide the necessary derivations to learn the parameters of the new layer.

In figure 3.8, a DNN structure with the proposed activation layer IT2-FAL is illustrated. In this structure, IT2-FAL contains C number of hidden units and these hidden units are constructed with IT2-FLSs. The inputs $\tilde{x}_c$ $(c = 1, \dots, C)$ of the IT2-FLSs placed in IT2-FAL is normalized as follows:

$$\sigma_c = K_f \tilde{x}_c \tag{3.15}$$

where σ_c are the normalized inputs and K_f is the scaling factor which is used to normalize the inputs into the $\sigma_c \in [-1,1]$. K_f generally selected as $K_f = 1/\max(\tilde{x}_c)$ and it should be mentioned that the K_f is calculated for each mini-batch which reveals that the value of the K_f is changed for each epoch in the learning process. We also denormalize the outputs of the IT2-FAL via the following formulation $\tilde{y}_c = K_f^{-1}\varphi_c$. Here the φ_c is the output of the IT2-FRU.

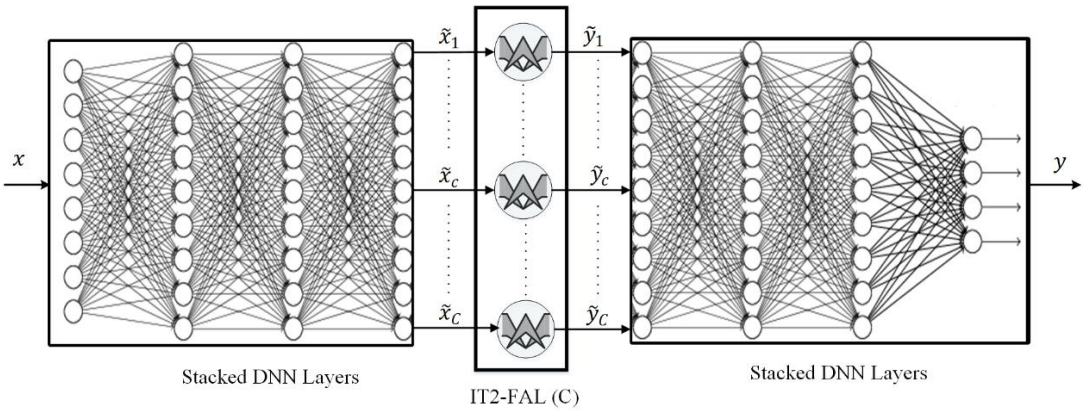


Figure 3.8 : An example of a DNN structure with the proposed activation function based on IT2-FLS.

3.4.2.1 The internal structure of SIT2-FRU

The rule structure of the proposed SIT2-FRU is defined as:

$$R_i: \text{IF } \sigma \text{ is } \tilde{A}_i \text{ THEN } \varphi \text{ is } \varphi_i = B_i, \qquad i = 1,2,3 \tag{3.16}$$

This rule structure is defined with three triangular IT2-FSs ($\tilde{A}_i$) which are constructed by extending the T1-FSs as demonstrated in Figure 3.9. While the consequent MFs are defined with crisp values (B_i). As we explained in section 2.2.1, an IT2-FS is a combination of an LMF ($\underline{\mu}_{\tilde{A}_i}$) and an UMF ($\overline{\mu}_{\tilde{A}_i}$). In this context, we can define the triangular type UMF as:

$$\overline{\mu}_{\tilde{A}_i} = \begin{cases} \dfrac{\sigma - c_{i+1}}{c_i - c_{i+1}} & \sigma \in [c_i, c_{i+1}] \\ \dfrac{c_{i-1} - \sigma}{c_{i-1} - c_i} & \sigma \in [c_{i-1}, c_i] \end{cases} \tag{3.17}$$

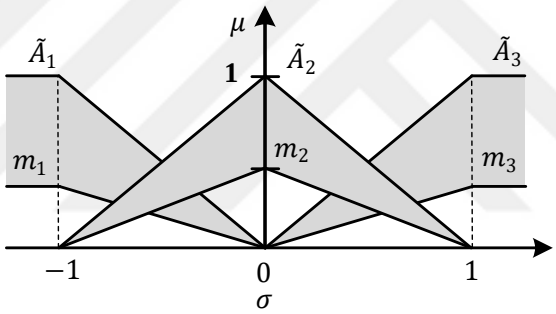


Figure 3.9 : Illustration of the antecedent MFs of SIT2-FRU.

Here, σ is the normalized input of the SIT2-FRU and c is the core of the triangle type UMF. In the illustration we give in Figure 3.9, the heights of the LMFs are defined with the parameter m. Therefore, we have three height parameters and they are reparametrized as $m_1 = h_1, m_2 = h_2$ and $m_3 = h_3$. The LMF can be calculated using the definitions we provided in equation 2.23 and 2.24 as follows:

$$\underline{\mu}_{\tilde{A}_i} = m_i \, \overline{\mu}_{\tilde{A}_i} \tag{3.18}$$

In the scope of this structure, the cores of the triangle type IT2-FS are predefined and selected as $c_1 = -1, c_2 = 0$ and $c_3 = 1$. As it can be seen from Figure 3.9, the only parameter that changes the amount of FOU is the height parameter of the LMFs (m_i). Furthermore, these height parameters are reorganized as we suggest in our previous work as in the following [45]:

$$m_2 = \alpha$$
$$m_1 = m_3 = 1 - \alpha \tag{3.19}$$

Here, α is the only parameter which can change the FOU of the IT2-FSs and it should satisfy the $\alpha \in [0,1]$. The consequent parameters of the rules of SIT2-FRU is defined with crisp numbers as seen from equation 3.16. We construct the consequent MFs with three linguistic parameters as following:

$$B_1 = N \quad B_2 = Z \quad B_3 = P \tag{3.20}$$

Here, N and P can be selected as learnable parameters or they can be hyperparameter to be predefined before training. On the other hand, we set $Z = 0$ in order to satisfy the equation $\varphi(\sigma = 0) = 0$.

The output of the SIT2-FRU can be calculated with respect to the type-reduction or center of set calculation method namely KM method introduced in [61] as follows:

$$\varphi = (\varphi_r + \varphi_l)/2 \tag{3.21}$$

In above equation, φ_r and φ_l are the type reduced set and they can be found the equations we provided in 2.27 and 2.28, also they can be reformulated for 3 rules as:

$$\varphi_r = \frac{\sum_{i=1}^{R} \underline{\mu}_{\tilde{A}_i} \varphi_i + \sum_{i=R+1}^{I=3} \overline{\mu}_{\tilde{A}_i} \varphi_i}{\sum_{i=1}^{R} \underline{\mu}_{\tilde{A}_i} + \sum_{i=R+1}^{I=3} \overline{\mu}_{\tilde{A}_i}}$$
$$\varphi_l = \frac{\sum_{i=1}^{L} \overline{\mu}_{\tilde{A}_i} \varphi_i + \sum_{i=L+1}^{I=3} \underline{\mu}_{\tilde{A}_i} \varphi_i}{\sum_{i=1}^{L} \overline{\mu}_{\tilde{A}_i} + \sum_{i=L+1}^{I=3} \underline{\mu}_{\tilde{A}_i}} \tag{3.22}$$

As we illustrated in Figure 3.9, we choose to employ fully overlapped triangle type IT2-FS for the SIT2-FRU. In this context, it can be commented that the normalized inputs of the FSs always be a member of two IT2-FSs ($\tilde{A}_i$ and $\tilde{A}_{i+1}$) at one time. Therefore, for one particular input at most two rules be activated and this situation results with $R = L = 1$ (left and right switching points defined in reference [61]) as shown in the reference paper [45]. Therefore, the mapping characteristic of the SIT2-FRU can be formulized as follows:

$$\varphi(\sigma) = \frac{1}{2}\left(\frac{\overline{\mu}_{\tilde{A}_i}\varphi_i + \underline{\mu}_{\tilde{A}_{i+1}}\varphi_{i+1}}{\overline{\mu}_{\tilde{A}_i} + \underline{\mu}_{\tilde{A}_i}} + \frac{\underline{\mu}_{\tilde{A}_i}\varphi_i + \overline{\mu}_{\tilde{A}_{i+1}}\varphi_{i+1}}{\underline{\mu}_{\tilde{A}_i} + \overline{\mu}_{\tilde{A}_{i+1}}}\right) \tag{3.23}$$

The above equation can be redefined by if we substitute the equations 3.17 and 3.20 into 3.23 and some algebraic manipulation, we can formulate the mapping function of the SIT2-FRU for the $\sigma \in [0, 1]$ as

$$\varphi(\sigma) = P\sigma k(\sigma) \tag{3.24}$$

Furthermore, we can formulate the $k(\sigma)$ as the below:

$$k(\sigma) = \frac{1}{2}\left(\frac{1}{\alpha + \sigma - \alpha\sigma} + \frac{-1 + \alpha}{-1 + \alpha\sigma}\right) \tag{3.25}$$

we can also formulate the mapping function of the SIT2-FRU for the $\sigma \in [-1, 0]$ as

$$\varphi(\sigma) = N\sigma k(-\sigma) \tag{3.26}$$

At that point it is possible to visualize the mapping characteristic of the SIT2-FRU according to the mapping residue which can be defined as $\varepsilon_0(\sigma) = |\varphi(\sigma)| - |\sigma|$. This can be also defined as residue which is left from the differences of the absolute values normalized input (σ) and the output of the SIT2-FRU ($\varphi(\sigma)$).

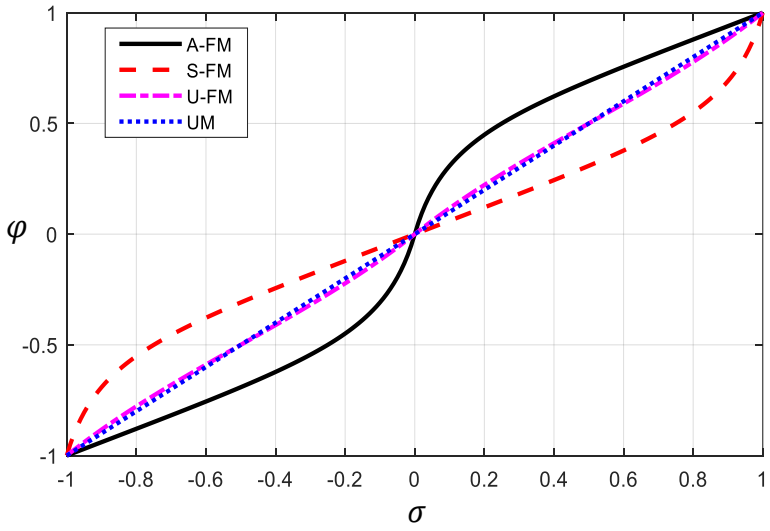


Figure 3.10 : Illustration of the mapping characteristics of SIT2-FRU.

The different mapping characteristic of the SIT2-FRU is visualized in Figure 3.10 with respect to the different values of α. The selection of α affects the mapping characteristics and they can be listed as also provided in reference [45]:

(i) If the α is set to a value which stay in the interval of $0 < \alpha \leq \alpha_A$ then it leads to an absolute output always bigger than the absolute input which makes $\varepsilon_0(\sigma) > 0$ always guranteed for $\forall\, \sigma \in [-1, 1]$. Thus, we defined the mapping characteristic of the SIT2-FRU as Aggressive Fuzzy Mapping (A-FM) for $0 < \alpha \leq \alpha_A$.

(ii) If the α is set to a value which stay in the interval of $\alpha_S \leq \alpha < 1$ then it leads to an absolute output always smaller than the absolute input which makes $\varepsilon_0(\sigma) < 0$ always guranteed for $\forall\, \sigma \in [-1, 1]$. Thus, we defined the mapping characteristic of the SIT2-FRU as Smooth Fuzzy Mapping (S-FM) for $\alpha_S \leq \alpha < 1$.

(iii) If the α is set to a value which stay in the interval of $\alpha_A < \alpha < \alpha_S$ then it leads to an absolute output almost equal to the absolute input which makes $\varepsilon_0(\sigma) \approx 0$ for $\forall\, \sigma \in [-1, 1]$. Thus, we defined the mapping characteristic of the SIT2-FRU as Unit Fuzzy Mapping (U-FM) for $\alpha_A < \alpha < \alpha_S$.

Here, the values of α_A and α_S are set as $\alpha_A = (3 - \sqrt{5})/2$ and $\alpha_S = (-1 + \sqrt{5})/2$. In Figure 3.10, we set the learnable consequent parameters as $P = N = 1$. On the other hand, we set the $\alpha = 0.1$ to obtain the A-FM and we select the $\alpha = 0.9$ to end up with a S-FM while we set the $\alpha = 0.5$ for U-FM. In this figure, it can be seen that the S-FM has a low sensitivity compared to A-FM especially focused on the $\sigma \approx 0$. Besides, it is possible to almost obtain a U-FM when α is selected from the range of $\alpha_A < \alpha < \alpha_S$ which is also can be seen from Figure 3.10.

3.4.2.2 Mapping characteristics of the type-2 fuzzy activation units

In this subsection, we provide the input-output characteristics of the proposed SIT2-FRU. Firstly, we can rearrange the formulations we derived in equations 3.24 and 3.26 which are related to mapping characteristics of the SIT2-FRU and they can be reorganized as follows:

$$f(\sigma; \boldsymbol{\theta}) = \begin{cases} P\sigma k(\sigma), & \text{if } \sigma > 0 \\ N\sigma k(-\sigma), & \text{if } \sigma \leq 0 \end{cases} \tag{3.27}$$

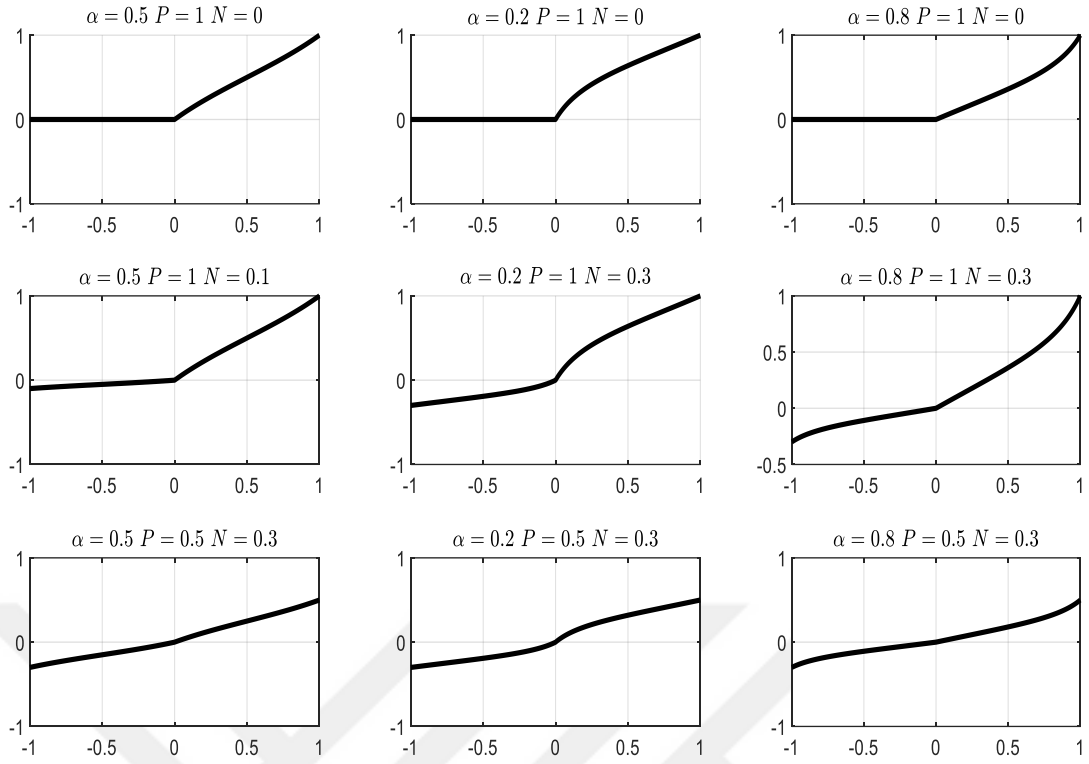


Figure 3.11 : Illustration of the mapping characteristics of SIT2-FRU for different selection of the parameter set $\boldsymbol{\theta} = [\alpha, P, N]$.

The selection of the learnable parameter set $\boldsymbol{\theta} = [\alpha, P, N]$ directly affects the input-output mapping characteristic of the SIT2-FRU. It is possible to construct various linear or sophisticated mapping characteristics with the different set of selection of the learnable parameter set $\boldsymbol{\theta}$ as it is demonstrated in Figure 3.11. As it is also noticeable in Figure 3.11, the parameter P actually defines the slope of the mapping function for the inputs in positive quadrant, N actually responsible for the slope of the mapping function for the inputs in negative quadrant. Moreover, when the α is set to a value which stay in the interval of $\alpha_A < \alpha < \alpha_S$ then the mapping is reduced to function which is close to a linear activation with the slopes of N for negative quadrant and P for positive quadrant. On the other hand, if the α is set to a value which stay in the interval of $0 < \alpha \leq \alpha_A$ or if it is set to a value which stay in the interval of $\alpha_S \leq \alpha < 1$, the mapping function defined in equation 3.27 can end up with complex nonlinear input-output mappings which have aggressive or smooth characteristics. In this context, the design parameter set $\boldsymbol{\theta} = [\alpha, P, N]$ can be defined as hyperparameters to be defined by user before training or they can be selected as learnable parameters which will be learned by the proposed algorithm to improve the performance of the DNNs. In this structure, the total number of learnable parameters will be $3C$ since the

IT2-FAL has the C hidden units. Thus, it can be said that IT2-FAL will not bring a huge computation burden to deep learning algorithm compared to a DNN with huge amount of weights.

The SIT2-FRU has several new features that can enhance the DNN performance. the SIT2-FRU activation function has several benefits that set it apart from other activation functions such as ReLU, ELU, and PReLU. Its ability to handle input values in the negative quadrant, learn each hidden neuron's contribution in the positive quadrant, and map complex input-output relationships in the negative quadrant through the FOU parameters N and α make it a versatile and valuable tool for enhancing DNN's DL performance. Additionally, SIT2-FRU's ability to mitigate the impact of small variations in certain hyperparameter settings makes it more resilient against parameter variations, which is crucial in real-world applications. The unique features of SIT2-FRU offer exciting possibilities for future research in this field and highlight the importance of continually exploring new activation functions to improve the performance of DNNs. Moreover, if necessary, SIT2-FRU can also alleviate the impact of small variations in certain hyperparameter settings. For instance, when $\sigma \approx 0$, SIT2-FRU has a relatively low input sensitivity when S-FM is used. This results in an activation unit that eliminates the problem of propagating noisy input values to the next layer, making the resulting DNN with IT2-FAL potentially more robust against parameter variations. Overall, the SIT2-FRU novel offers several unique features that can enhance the DL performance of DNN, making it a promising candidate for future research in this field.

3.4.2.3 Learning the parameter of type-2 fuzzy activation units

This subsection provides the necessary derivations to learn the design parameter set of the SIT2-FRU. As we mentioned in previous section, the design parameter set $\boldsymbol{\theta} = [\alpha, P, N]$ can be defined as hyperparameters to be defined by user before training or they can be selected as learnable parameters which will be learned by the proposed algorithm. In this context, if the user decides to use these parameters as hyperparameters then it can be tuned using the definitions we provided in section 3.4.2.2 and set them as desired. On the other hand, if the user left these parameters to be learned by the algorithm we provide the necessary mathematical derivation in the below.

The parameter update rule for the learnable parameter set $\boldsymbol{\theta} = [\alpha, P, N]$ can be derived using the chain rule as in the following:

$$\frac{\partial L}{\partial \theta_c} = \sum_j \frac{\partial L}{\partial f_c(\sigma_{cj})} \frac{\partial f_c(\sigma_{cj})}{\partial \theta_c} \tag{3.28}$$

Here, the channels in the network are indexed by the variable c which takes values from 1 to C, and the elements over height, width, and observations are represented by the index j. The objective function of the network is denoted by L. The term $\frac{\partial L}{\partial f_c(\sigma_{cj})}$ refers to the gradient that is propagated from the deeper layers after the IT2-FAL activation function is applied. The gradients of the SIT2-FRU can be derived as follows:

$$\frac{\partial f_c(\sigma_c)}{\partial \alpha_c} = \begin{cases} \frac{P_c \sigma_c}{2}\left(\frac{1}{\alpha_c\sigma_c - 1} + \frac{\sigma_c - 1}{(\alpha_c + \sigma_c - \alpha_c\sigma_c)^2} + \frac{\sigma_c(1-\alpha_c)}{(\alpha_c\sigma_c - 1)^2}\right), & \text{if } \sigma_c > 0 \\ -\frac{N_c \sigma_c}{2}\left(\frac{1}{\alpha_c\sigma_c + 1} + \frac{\sigma_c + 1}{(\alpha_c - \sigma_c + \alpha_c\sigma_c)^2} + \frac{\sigma_c(1-\alpha_c)}{(\alpha_c\sigma_c + 1)^2}\right), & \text{if } \sigma_c \leq 0 \end{cases} \tag{3.29}$$

$$\frac{\partial f_c(\sigma_c)}{\partial P_c} = \begin{cases} \sigma_c k_c(\sigma_c), & \text{if } \sigma_c > 0 \\ 0, & \text{if } \sigma_c \leq 0 \end{cases}$$

$$\frac{\partial f_c(\sigma_c)}{\partial N_c} = \begin{cases} 0, & \text{if } \sigma_c > 0 \\ \sigma_c k_c(-\sigma_c), & \text{if } \sigma_c \leq 0 \end{cases}$$

Here, the $k_c(.)$ in equation above is the same function as described in equation (3.25). The updates of the learnable parameter $\boldsymbol{\theta}$ can be achieved by the Stochastic Gradient Descent with Momentum method, which is introduced in section 3.2.1, as following:

$$\Delta\theta = \rho\Delta\theta + \gamma\frac{\partial L}{\partial \theta} \tag{3.30}$$

where ρ is the momentum and γ stands for the learning rate parameter.

Remark: it should be notified that we have not constrained the negative and positive slope parameters N, P and use them directly in the unconstrained optimizer namely Stochastic Gradient Descent with Momentum. On the other hand, we constrained the FOU parameter (α) as $\alpha \in [0, 1]$ because we need to take into account that this

parameter is the FOU parameter of an IT2-FS and $\overline{\mu}_{\tilde{A}_i} > \underline{\mu}_{\tilde{A}_i}$, $\forall f_c\ (c = 1, \ldots, C)$ must be always satisfied for IT2-FSs.

4. MORE THAN ACCURACY: A COMPOSITE LEARNING FRAMEWORK FOR INTERVAL TYPE-2 FUZZY LOGIC SYSTEMS

This section presents a novel composite learning framework for IT2-FLSs. In literature a majority of rule of thumb papers aim to design/tune/learn/optimize a T1/T2-FLS model for the handled problem that has a highly accurate mean prediction. In order to increase the accuracy of the IT2-FLS, the researchers generally use the FOU parameter which is in the antecedent parts. Therefore, the wide majority of the researchers treat the FOU parameter like another design parameter to increase the mean accuracy rather than a parameter which can be used to envelope the uncertainty. We see limited amount of paper, in literature, in which the type-reduced set of constructed IT2-FLS is used to cover the uncertainty or generate prediction intervals [49], [101, 102].

In this proposed work, we present a novel framework to learn the parameters of various types of TSK type IT2-FLSs with a motivation of "More than Accuracy". In this context, the aim of the proposed framework not only learn an IT2-FLS with a high prediction mean accuracy but also envelope the uncertainty at the same time exploiting their type reduced set. In this context, we have identified three primary obstacles that must be overcome to achieve this aforementioned objective: (i) the ability of IT2-FLSs to overcome unknowns or uncertain effects, (ii) the development of a suitable objective function, and (iii) the construction of an algorithm that can manage the complexity of learning while considering the constraints which brings by IT2-FLSs and also the handled dataset such as its feature and size.

i. The inference mechanism of an IT2-FLS determines how uncertain inputs are translated into uncertain outputs. It decides how the system makes decisions and how it deals with missing or incomplete information. A robust inference mechanism can help IT2-FLSs handle uncertainty more effectively. The structure of an IT2-FLS determines the way in which the components of the system are connected and how they interact with each other. A well-designed structure can help the system cope with uncertainty by allowing it to

incorporate multiple sources of information and multiple decision-making criteria. Therefore, we consider three different CSCMs since processing the uncertainty gets effected significantly by the CSCM, as we explained the details of these methods in section 2.2.1, they can be listed as; the first method is a SM which directly uses the upper and lower bounds of the IT2-FSs, the second one is NT method and the last one is the widely used KM method. After that, we construct different structured IT2-FLSs, to improve the handling capability of the FLSs against the uncertainties, leveraging the antecedent Membership Functions (MFs) and the rule consequents. Specifically, we consider Gaussian type IT2-FSs with different FOU structures for the antecedent MFs, and either crisp, linear or Interval Valued (IV) functions for the rule consequents. By varying the size of the FOU for the antecedent MFs and deploying IT2 and IV-FSs into the antecedents and consequents, respectively, we can construct IT2-FLSs that capture uncertainty in different ways. Some IT2-FLSs may rely solely on the uncertainty captured in their antecedents, while others may benefit from the use of IT2 and IV-FSs in both the antecedents and consequents. Overall, we define various IT2-FLSs along with their learnable parameters that represent and process uncertainty differently. By exploring different structures and parameters, we can design IT2-FLSs that can effectively handle uncertainty in various real-world applications.

ii. In the construction of our loss function, we aim to balance accuracy and the ability to cover uncertainty, which are both crucial for designing effective IT2-FLSs that can handle real-world uncertainties. In order to find an IT2-FLS has a high model accuracy, we use an empirical loss function, which is a common approach in machine learning. To capture uncertainty, we use a tilted loss function l_τ, which is often used in Quantile Regression (QR) to generate Prediction Intervals (PIs). This function defines an expected quantile range that we want our IT2-FLS to cover. By including this tilted loss function in our optimization problem, we ensure that our IT2-FLSs can handle uncertainty by producing PIs that cover the expected range. In order to simplify the learning process and avoid increasing complexity, we convert our multi-objective loss function to a composite loss via a hyperparameter which is widely done in DL approach where a single loss function is used to optimize the model. Typically,

the L_1 or L_2 norm is used to learn the prediction mean in DL. However, in our case, the empirical risk function must have similar properties to the tilted loss l_τ to prevent the degrading effects as the training continuous since there is a hard parameter sharing between the empirical risk and tilted loss. To address this, we proposed to select $L_{log-cosh}$ as the empirical risk function in our proposed novel loss function. By choosing this loss function, we give opportunity that the empirical risk works well with the tilted loss and $L_{log-cosh}$ also has the beneficial aspects of L_1 and L_2 [52]. Moreover, the need for setting a separate hyperparameter is eliminated in our proposed composite loss function due to its alignment with the tilted loss function which makes the learning process simpler and more efficient. Our proposed loss function explicitly leverages the reduced set ($[\underline{y}, \overline{y}]$) in the tilted loss function and the defuzzified output (y) in the $L_{log-cosh}$ function. By combining these two components, we can train IT2-FLSs that has ability of both enveloping an expected amount of uncertainty and achieving a high performance in prediction outputs.

iii. The parameter learning process of the IT2-FLS is not a trivial job since there are some challenges related to the big data problem such as size and dimensionality. On the other hand, we have some constraints over our optimization problem since the learnable parameters are the parameters of the IT2-FSs which they must satisfy some definitions coming from the FS theory. While DL can help overcome the former issue, yet directly implementing it is not feasible due to the constraints related to IT2-FSs must satisfy some definitions. To address this issue, we will present some new parameterization tricks that can handle the optimization problem as an unconstraint one and it allows us to apply DL algorithms for learning IT2-FLSs. By doing so, we can effectively deal with the complexities of the data while also satisfying the constraints imposed by the FSs. This approach allows us to learn the parameters of the IT2-FLS in an efficient and effective manner, without compromising on the accuracy or handling of uncertainties.

In the scope of this thesis, we will provide extensive experimental results to demonstrate the effectiveness of our new approach. we conduct a thorough analysis of the learning performance and its sensitivity to hyperparameters using a univariate

dataset. Through this analysis, we aim to show how our approach can effectively handle uncertainties and achieve high accuracy performance on real-world data. By examining the sensitivity of our approach to hyperparameters, we can identify the optimal values for these parameters and fine-tune the learning process to achieve even better results. Our comprehensive analysis provides valuable insights into the capabilities and limitations of the learned IT2-FLSs via the proposed approach, allowing us to optimize its performance and ensure its effectiveness in a wide range of applications. In the experiment section, section 6.2.2 and 6.2.3, we will present intra-model (different types of IT2-FLSs) and inter-model (compared to commonly used models) comparisons using multivariate datasets. Therefore, our results will provide a strong evidence for the effectiveness of our proposed approach, and demonstrate its potential for use in a wide range of applications where uncertainty is a significant concern.

The proposed composite learning approach for IT2-FLSs brings the following novelties:

- our approach provides a new and effective way to learn IT2-FLSs that can effectively handle uncertainty, making it suitable for use in a wide range of real-world applications where accurate predictions are critical.
- In proposed composite loss, by combining the empirical loss for accuracy and the tilted loss for capturing uncertainty, we enable the effective handling of both objectives within a single framework. This represents a significant improvement over traditional methods, which may focus solely on accuracy and neglect uncertainty.
- The proposed parametrization tricks for IT2-FLSs represents a significant improvement since the other approaches may struggle to effectively train IT2-FLSs within the context of DL frameworks.
- The results obtained from our study and presented in section 6.2 demonstrate that our proposed DL-based training approach effectively learn the parameters of IT2-FLSs with the capability to capture and process uncertainty while achieving high levels of accuracy. In fact, our approach outperformed various models, in some studies DNNs are used, in terms of accuracy. This highlights the effectiveness of our approach in addressing the challenges associated with handling uncertainty.

4.1 Preliminaries on Quantile Regression

The main idea of parameter estimation is that estimating the conditional mean $\mathbb{E}[y|\mathrm{X} = x]$ of the output (y) by using the inputs $\boldsymbol{x}$ and function $q(.)$ defined:

$$y = q(\boldsymbol{x}; \boldsymbol{\theta}) + \epsilon \tag{4.1}$$

here $\boldsymbol{\theta}$ is the learnable parameter set and $\epsilon \in \mathbb{R}$ stands for the estimation error.

Estimating only the conditional mean might be inadequate for the datasets that have heteroscedasticity behavior [103]. In the analytical sense, the QR approach may capture the exact representation of the whole conditional distribution of y using the input rather than only taking the conditional mean into consideration [49]. As a result, conditional quantities may be argued to be more efficient ways for real-world applications since they can better capture the conditional distribution of the modeling phenomena.

The conditional quantile for $\tau \in (0, 1)$ is defined as $q^{\tau}(\boldsymbol{x}) \in \mathbb{R} : P(y < q^{\tau}(\boldsymbol{x})|\mathrm{X}) = \tau$. Generally, more than one quantitative function is estimated to capture unknown characteristics of real-world data. A PI I^{φ} with the expected range $\varphi = [\underline{\tau}, \overline{\tau}]$ may therefore be generated using a specified lower $(q^{\underline{\tau}})$ and upper $(q^{\overline{\tau}})$ quantile function. The minimization problem of the τ^{th} quantile [49] is as:

$$\arg\min_{\boldsymbol{\theta}} \sum_{n=1}^{N} \ell_{\tau}(y_n - q^{\tau}(\boldsymbol{x}_n; \boldsymbol{\theta}))$$

In above equation, q^{τ} represent the conditional quantile function at τ while the ℓ_{τ} stands for the tilted/ pinball loss that is defined as:

$$\ell_{\tau} = \begin{cases} \tau\epsilon_n^{\tau}, & \text{if } \epsilon_n^{\tau} \geq 0 \\ (\tau - 1)\epsilon_n^{\tau}, & \text{if } \epsilon_n^{\tau} < 0 \end{cases} \tag{4.2}$$

In the equation 4.2, ϵ_n^{τ} represents the quantile residue ($\epsilon_n^{\tau} = y_n - q^{\tau}(\mathbf{x}; \boldsymbol{\theta})$) for the n-th sample, where the penalty for the predicted value exceeding the specified quantile level is more pronounced. ℓ_{τ} can be also defined in a compact form as:

$$\ell_{\tau} = \max(\tau\epsilon_n^{\tau}, (\tau - 1)\epsilon_n^{\tau}) \tag{4.3}$$

The compact form clearly shows a problem: the sloping tilted ℓ_τ equals the absolute value of ϵ_n^τ scaled by τ. Hence, it can be concluded that when the predicted value goes beyond the chosen quantile level, it is penalized heavily, resulting in automatic generation of the expected prediction interval (I^φ) within the chosen range φ due to the tilted loss. However, computing separate quantile levels for each sample can be computationally intensive, and the crossing problem of quantile curves can also occur. Therefore, a composite loss with multiple objectives is defined to address these issues.

4.2 IT2-FLSs for Uncertainty Representation

To capture uncertainty, it is possible to construct different IT2-FLSs concerning the type of the employed antecedent MF (i.e. different FOU size and characteristic) and consequent MFs (i.e. crisp or interval valued functions). For instance, an IT2-FLS can be constructed to model the uncertainty through only its antecedent MFs while another one could try to capture the uncertainty by benefiting from the deployment of IT2-FSs and interval-valued consequents. Also, the processing of the captured uncertainty depends on the deployed center of sets calculation methods. Therefore, their way of representing and processing uncertainty is quite different. In this section, we define various Takagi-Sugeno-Kang type IT2-FLSs alongside their learnable parameters to capture the uncertainty.

4.2.1 The internal structure of the TSK-type IT2-FLSs

The formulation of the rules for IT2-FLS for uncertainty capturing can be constructed for an input $\boldsymbol{x} = (x_1, x_2, \dots, x_M)^T$, as we provided preliminary theoretical background in section 2.2.1, as in following:

$$Rule_p : \text{If } x_1 \text{ is } \tilde{A}_{p,1} \text{ and ... } x_M \text{ is } \tilde{A}_{p,M} \text{ Then } y \text{ is } \tilde{Y}_p \tag{4.4}$$

In the above equation $\tilde{Y}_p$ is the consequents and we provided the definition of the $\tilde{Y}_p$ in equation 2.19. Here, if we select the number of outputs $t = 1$, the consequents can be redefined as:

$$\underline{y_p} = \sum_{m=1}^{M} a_{p,m} x_m + a_{p,0} - \sum_{m=1}^{M} b_{p,m} |x_m| - b_{p,0} \tag{4.5}$$

$$\overline{y}_p = \sum_{m=1}^{M} a_{p,m} x_m + a_{p,0} + \sum_{m=1}^{M} b_{p,m} |x_m| + b_{p,0}$$

For the antecedent MFs of the IT2-FLS we consider to use a gaussian type IT2-MF. In this context, we can formulate a gaussian type IT2-MF using the equations 2.7, 2.23 and 2.24 we provided in section 2 as follows:

$$\begin{aligned} \overline{\mu}_{\tilde{A}_{p,m}}(x_m) &= \exp\left(-\left(x_m - \overline{c}_{p,m}\right)^2 / 2\overline{\sigma}_{p,m}^2\right) \overline{y}_p \\ \underline{\mu}_{\tilde{A}_{p,m}}(x_m) &= h_{p,m} \exp(-\left(x_m - \underline{c}_{p,m}\right)^2 / 2\underline{\sigma}_{p,m}^2) \end{aligned} \tag{4.6}$$

where $\underline{c}_{p,m}$ and $\overline{c}_{p,m}$ are the centers, while $\underline{\sigma}_{p,m}$ and $\overline{\sigma}_{p,m}$ are the standard deviations of LMFs and UMFs.

In order to calculate the crisp output of the IT2-FLSs, we consider three CSCM methods as we mentioned earlier in section 2.2.1. It should be noted that there are various type reduction methods are available in literature. KM type reduction [61] and NT [51], which are summarized in section 2.2.1 in equations 2.27 to 2.31, type reduction methods are the widely used methods in literature. Therefore, we also picked these two methods in this thesis to be employ as the type reducer for the employed IT2-FLSs. We also use a method that directly uses the endpoints of the type-reduced set namely SM and it is formulized in equation 2.26. Consequently, we can calculate the crisp output of the IT2-FLS by using the equation 2.25.

4.2.2 Parametric IT2-FLSs models and learnable parameters of the IT2-FLSs

The IT2-FSs is usually constructed by blurring its T1 counterpart. In Figure 4.1, various IT2-FSs that could be constructed to represent the uncertainty are illustrated. In this subsection, we listed the various versions of IT2-FSs that we employed to represent uncertainty in this thesis study.

- *H type IT2-FSs:* In this type of antecedent MF, as illustrated in Figure 4.1a, the uncertainty is represented only with the height of the LMF ($h_{r,m}$) such that the parameters of IT2 gaussian FS, defined in equation 4.6, are set as $\underline{\sigma}_{p,m} = \overline{\sigma}_{p,m,} \forall\, p, m$ with condition of $h_{p,m}$ satisfying for $\forall\, h_{p,m} \in [0,1]$. It can be seen

as from Figure 4.1a, as the input ($x_{p,m}$) gets close to the center ($c_{p,m}$) of the gaussian FS the FOU area increases.

- *S type IT2-FS:* As demonstrated in Figure 4.1b, in this type of the IT2-FS, the IT2-FS is constructed by providing uncertainty at the end points of its T1 counterpart. In other word, the FOU is changed by the standard deviation parameters ($\underline{\sigma}_{p,m}$ and $\overline{\sigma}_{p,m}$) of the LMF and UMF. Therefore, we set the height of the LMF, defined in equation 2.24, as $h_{p,m} = 1$ and the centers as $\underline{c}_{p,m} = \overline{c}_{p,m}$ with condition of $\underline{\sigma}_{p,m} \leq \overline{\sigma}_{p,m}$, $\forall\, p, m$ is satisfied. It can be seen as from Figure 4.1b, as the input ($x_{p,m}$) gets close to the center ($c_{p,m}$) of the gaussian FS the FOU area decreases unlike the H type.
- *HS type IT2-FS:* The IT2-FSs illustrated in Figure 4.1c is the most general version where the both end points and height of the LMF is used to handle the uncertainty. In this context, HS type can be seen as the generalized version of the H and S types as it uses both $\underline{\sigma}_{p,m}$, $\overline{\sigma}_{p,m}$ and $h_{p,m}$ to arrange the FOU area of IT2-FS. Although, this version is the most flexible one, the parameters must satisfy some certain restrictions to satisfy the FS definitions. It must satisfy the condition of $h_{p,m} \in [0,1]$ for $\forall\, p, m$ as in H type and it also must fulfill the condition of $\underline{\sigma}_{p,m} \leq \overline{\sigma}_{p,m}$ for $\forall\, p, m$.

It should be pointed out that we arrange the centers of the IT2-FSs as $c_{p,m} = \underline{c}_{p,m} = \overline{c}_{p,m}$ for all versions of the IT2-FSs (H, S and HS).

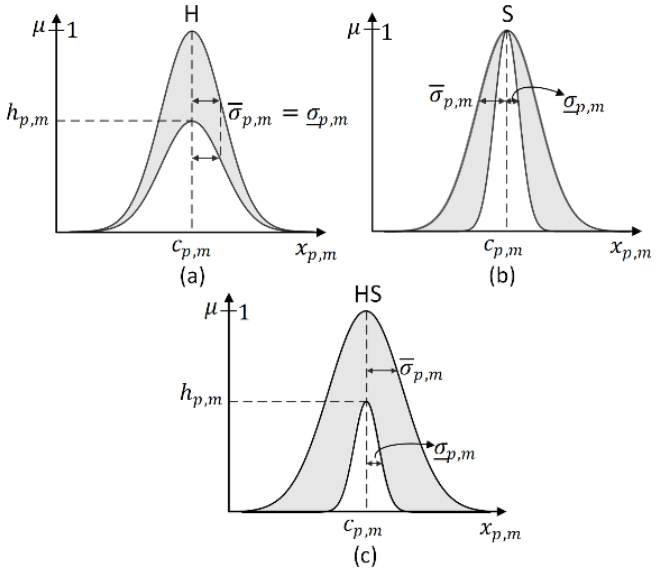


Figure 4.1 : Illustration of the various IT2-FS to capture the uncertainty with antecedent MFs (a) H type, (b) S type, (c) HS type.

In FLSs, another way of handling uncertainty is utilizing the consequent MFs. We considered to model the uncertainties using only antecedents for some FLSs while for some others we used antecedents and also consequents to handle uncertainties. The followings define the possible consequents.

- *Crisp (C) rule consequent:* In this type of rule consequent, we only consider to represent the uncertainty using only antecedents of IT2-FLSs and set the consequent parameters defined in equation 4.5 to obtain crisp MFs as $a_{p,m} = 0$, $b_{p,m} = 0$ and $b_{p,0} = 0$ for $\forall\, p, m$. Therefore, as it can be seen from equation 4.5, we obtained crisp consequents such that $\underline{y}_p = \overline{y}_p = a_{p,0}$.
- *Linear (L) rule consequent:* Similar to C rule consequents, we also only consider to use antecedents for uncertainty handling, but select linear consequents by setting parameters defined in equation 4.5 as $b_{p,m} = 0$ and $b_{p,0} = 0$

The following ones define the rule consequents that are capable of capturing the uncertainty as they are defined with IV-FSs.

The next two rule consequent type use IV-FSs for each rule to further increasing the expressiveness and accuracy of the IT2-FLS. By using these FSs, the IT2-FLS can produce output values that are associated with different levels of uncertainty, thereby providing a more comprehensive and accurate representation of the system's behavior.

- *IV rule consequent:* IT2-FLSs that uses IV consequents try to model the uncertainty not only their IT2 antecedents but also with their IV consequent FSs. In this context, the consequent parameters defined in 4.5 set as $a_{p,\mathrm{m}} = 0$, $b_{p,m} = 0$ for $\forall\, p, m$ to obtain IV-FSs as the consequents.
- *IV Linear (IVL) rule consequent:* The FLSs using the IV linear consequents try to handle the uncertainties with their both antecedent and consequent MFs in a similar manner to IV crisp ones. The IT2-FLSs with IVL consequents are the most flexible models as we do not any presetting on the parameters defined in equation 4.5 for $\forall\, p, m$. Although, this version can be seen as the most general one among the others, it brings more design parameters and restrictions on the them.

It should be noted that the rule structures of all parametric IT2-FLSs are the same as defined in (4).

In Table 4.1, all possible parametric interval models are tabulated. In total, we have 12 parametric models and some models use only antecedents to represents the uncertainty while some models use both of their antecedent and consequent MFs. The increase in model flexibility leads to an increase in the total number of learnable parameters. It is worth noting that due to the dependence of the uncertainty processing on the CSCM, there are 36 different IT2-FLSs in the proposed approach.

Table 4.1 : All possible Parametric IT2-FLS.

		Consequent			
		C	**L**	**IV**	**IVL**
Antecedents	**H**	H-C	H-L	H-IV	H-IVL
	S	S-C	S-L	S-IV	S-IVL
	HS	HS-C	HS-L	HS-IV	HS-IVL

The learnable parameters ($\boldsymbol{\theta}$) of the IT2-FLSs also the parameters of antecedent ($\boldsymbol{\theta_A}$) and consequents ($\boldsymbol{\theta_C}$) MFs. All of these parameters can be trained with a derivative-based optimization approach, Adam is used in this work, or the parameter is predefined by the user before the training. However, they must satisfy some constraints as we explained the parametric IT2-FLSs, thus in the following sections (section 4.4) we will define them.

The learnable parameter sets will be formed with respect to the selection of the parametric models. We group the parameters regarding to whether they come from antecedent or consequent for the sake of simplicity. In the followings, we will list the learnable parameter sets for the all parametric IT2-FLSs defined in Table 4.1.

The learnable parameter set for the rule antecedents are:

- *H type FS:* The set of learnable parameters for $\boldsymbol{\theta_{A-H}} = \{\boldsymbol{c}, \boldsymbol{\sigma}, \boldsymbol{h}\}$ where $\boldsymbol{c} = \left(c_{1,1}, \dots, c_{P,M}\right)^T \in \mathbb{R}^{P\times M}$, $\boldsymbol{\sigma} = \left(\sigma_{1,1}, \dots, \sigma_{P,M}\right)^T \in \mathbb{R}^{P\times M}$ $\left(\sigma_{p,m} = \underline{\sigma}_{p,m} = \overline{\sigma}_{p,m}\right)$ and $\boldsymbol{h} = \left(h_{1,1}, \dots, h_{P,M}\right)^T \in \mathbb{R}^{P\times M}$. Thus, $\boldsymbol{\theta_{A-H}}$ adds $3PM$ parameters coming from antecedents.

- *S type FS:* Learnable parameter set for S type IT2-FS is $\boldsymbol{\theta_{A-S}} = \{\boldsymbol{c}, \underline{\boldsymbol{\sigma}}, \overline{\boldsymbol{\sigma}}\}$ where $\underline{\boldsymbol{\sigma}} = \left(\underline{\sigma}_{1,1}, \dots, \underline{\sigma}_{P,M}\right)^T \in \mathbb{R}^{P\times M}, \overline{\boldsymbol{\sigma}} = \left(\overline{\sigma}_{1,1}, \dots, \overline{\sigma}_{P,M}\right)^T \in \mathbb{R}^{P\times M}$. Thus, $\boldsymbol{\theta_{A-S}}$ adds $3PM$ parameters in total which lies in the antecedents.

- *HS type FS:* the set is $\boldsymbol{\theta_{A-HS}} = \{\boldsymbol{c}, \underline{\boldsymbol{\sigma}}, \overline{\boldsymbol{\sigma}}, \boldsymbol{h}\}$. This is the most flexible version and $\boldsymbol{\theta_{A-HS}}$ brings $4PM$ learnable parameters.

Table 4.2 : Learnable parameter numbers of all possible Parametric IT2-FLS.

		Consequent ($\boldsymbol{\theta_C}$)			
		C	**L**	**IV**	**IVL**
Antecedent ($\boldsymbol{\theta_A}$)	**H**	3PM+P	3PM+ P(M+1)	3PM+ P(M+1)	3PM+ 2P(M+1)
	S	3PM+P	3PM+ P(M+1)	3PM+ P(M+1)	3PM+ 2P(M+1)
	HS	4PM+P	4PM+ P(M+1)	4PM+ P(M+1)	4PM+ 2P(M+1)

Now, we present the learnable parameter set for the consequents parts which are:

- *C rule consequent:* The set of learnable parameters for Crisp consequents is $\boldsymbol{\theta_{C-C}} = \{\boldsymbol{a_0}\}$ where $\boldsymbol{a_0} = \left(a_{1,0}, \dots, a_{P,0}\right)^T \in \mathbb{R}^{P\times 1}$. It has P parameters.

- *L rule consequent:* Learnable set for the IT2-FLSs with Linear consequents is $\boldsymbol{\theta_{C-L}} = \{\boldsymbol{a}, \boldsymbol{a_0}\}$ with $\boldsymbol{a} = \left(a_{1,1}, \dots, a_{P,M}\right)^T \in \mathbb{R}^{P\times \mathrm{M}}$. It has $P(M+1)$ parameters.

- *IV rule consequent:* the set is $\boldsymbol{\theta_{C-IV}} = \{\boldsymbol{a}, \boldsymbol{b_0}\}$ where $\boldsymbol{b_0} = \left(\, b_{P,0}\right)^T \in \mathbb{R}^{P\times 1}$. It has $P(M+1)$ parameters.

- *IVL rule consequent:* it is $\boldsymbol{\theta_{C-IVL}} = \{\boldsymbol{a}, \boldsymbol{a_0}, \boldsymbol{b}, \boldsymbol{b_0}\}$, with $\boldsymbol{b} = \left(b_{1,1}, \dots, b_{P,M}\right)^T \in \mathbb{R}^{P\times \mathrm{M}}$. It is the most general case and adds $2P(M+1)$ learnable parameters.

The total learnable parameter for an IT2-FLS is the summation of its antecedent and consequent parts $\boldsymbol{\theta} = \{\boldsymbol{\theta_A}, \boldsymbol{\theta_C}\}$. For instance, the parameter set for the most flexible version $\boldsymbol{\theta} = \{\boldsymbol{\theta_{A-HS}}, \boldsymbol{\theta_{C-IVL}}\}$ and it has $4PM$ (antecedent MF parameters) + $2P(M+$

1) (consequent MF parameters) $= 4PM + 2P(M + 1)$ parameters in total. In Table 4.2, the total number of learnable parameters is presented.

4.3 The Composite Loss Function for Learning IT2-FLSs

In this section we present the proposed composite loss function for IT2-FLSs. Here, our purpose is to train an IT2-FLS model that can accurately predict outputs while also handling uncertainty. To achieve this, we formulate a multi-objective optimization problem for a dataset of N samples $\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$ where x_n is an M-dimensional input vector and y_n is the corresponding output. The goal is to minimize two objectives: the first is the empirical risk, which measures the model's mean error in predicting y_n for each input x_n; the second is the tilted loss, which considers the model's ability to capture the uncertainty in the output. We aim to find the optimal values for the learnable parameters that minimize both objectives simultaneously. This is a challenging optimization problem because the total number of parameters increases with the model's flexibility, and the problem is further complicated by the 36 different IT2-FLSs required to handle uncertainty effectively. Thus, we can define the problem as:

$$P_1: \min_{\boldsymbol{\theta}\in\boldsymbol{C}} L = \left(\sum_{n=1}^{N} L_R\big(f(x_n)\big), \sum_{n=1}^{N} \ell_\tau\big(q^\tau(x_n)\big) \right) \tag{4.7}$$

The optimization problem P_1 aims to learn an IT2-FLS that achieves high accuracy and envelopes uncertainty. It consists of an empirical risk term $L_R(\cdot)$ and a tilted loss term $l_\tau\,(.)$ to capture uncertainty, subject to constraints defined by set C. In order to make the learning process easier, we convert P_1 into a constrained optimization problem with a single objective as follows:

$$P_2: \begin{cases} \min\limits_{\boldsymbol{\theta}\in\boldsymbol{C}} & L = \sum\limits_{n=1}^{N} L_R\big(f(x_n)\big) \\ \text{s.t.} & \sum\limits_{n=1}^{N} \ell_\tau\big(q^\tau(x_n)\big) \le t \end{cases} \tag{4.8}$$

Here, we aim to train an IT2-FLS that can achieve high accuracy while effectively handling uncertainty. This involves ensuring that the uncertainty coverage meets a

certain threshold value $t \geq 0$, which needs to be sufficiently large to ensure feasibility of the optimal solution. It is important to note that the feasibility of the optimum solution depends on the value of t; if it is not set high enough, the optimal solution may not be feasible.

Next, in line with common practice in deep learning, we reformulate P_2 as a composite loss function by using the Lagrange form with a threshold of $t = 0$. The composite loss function is defined as follows, and it combines the empirical risk for accuracy and the tilted loss for uncertainty coverage:

$$P_3: \min_{\boldsymbol{\theta} \in \boldsymbol{\mathcal{C}}} L = \sum_{n=1}^{N} \left[L_R\big(f(x_n)\big) + \lambda\, \ell_\tau\big(q^\tau(x_n)\big) \right] \tag{4.9}$$

The parameter $\lambda \geq 0$ is the Lagrange multiplier and controls the trade-off between the two objectives. The composite loss function allows us to optimize the model for both accuracy and uncertainty coverage simultaneously. Moving from the original learning problem P_1 to P_2 and then to P_3 results in a natural limitation of the space of potential solutions since

$$\bigcup_{\lambda \geq 0} \{\mathrm{Soln}(P_3)\} \subseteq \bigcup_{t \geq 0} \{\mathrm{Soln}(P_2)\} \subseteq \bigcup \{\mathrm{Soln}(P_1)\} \tag{4.10}$$

As a result, while the transformation of P_1 to P_3 narrows down the potential solutions, it also simplifies the learning process. However, it is important to note that this approach may not necessarily result in the global optimum of P_1. Despite this, we still prefer to work with P_3 due to its reduced complexity.

In the following part of the thesis, we will explain how we select our empirical risk function $L_R(\cdot)$ and the tilted loss $\ell_\tau(.)$ as they will construct the composite loss for learning the parameters of the parametric IT2-FLSs.

The most common two losses used as empirical risk funtions for accuracy are the L_1 and L_2 losses which can be defined:

$$L_1 = \frac{1}{N}\sum_{n=1}^{N}|\epsilon_n|, \quad L_2 = \frac{1}{N}\sum_{n=1}^{N}{\epsilon_n}^2 \tag{4.11}$$

Here, $\epsilon_n = y_n - y(\boldsymbol{x}_n)$. An alternative loss function that can be used instead is the log-cosh loss, which has been proposed in previous research [52]. The log-cosh loss is formulated as the logarithm of the hyperbolic cosine of the difference between the predicted output and the ground truth. This loss function has some advantages over other loss functions, such as being more robust to outliers and having a smoother gradient. Moreover, the log-cosh loss is convex and differentiable, which makes it easy to optimize using gradient-based methods. It is defined as:

$$L_{\log-\cosh} = \frac{1}{N}\sum_{n=1}^{N}\log(\cosh(\epsilon_n)) \tag{4.12}$$

In Figure 4.2, the illustration shows the relationship between the loss functions and the value of ϵ_n. When ϵ_n is larger than 1 or smaller than -1, L_2 loss function has a steep gradient, which can lead to faster convergence and higher accuracy. However, this advantage comes with a trade-off. The L_2 loss function is known to be sensitive to noise and outliers. In contrast, the L_1 loss function is less affected by outliers as it imposes a smaller penalty when ϵ_n exceeds 1. However, the rate of convergence of the L_1 loss function is slower compared to the L_2 loss function since it applies less penalty as ϵ_n gets closer to 0. A significant limitation of the L_1 loss function is that it lacks differentiability at $\epsilon_n = 0$, which can result in instability and oscillations in the training procedure. On the other hand, the log-cosh loss function offers a blend of benefits from both L_1 and L_2 loss functions, without inheriting their drawbacks. As illustrated in Figure 4.2, when ϵ_n is small, the behavior of the log-cosh loss function is similar to that of the L_2 loss function, while when ϵ_n is large, its behavior is similar to that of the L_1 loss function. The log-cosh loss function's ability to behave like L_1 loss function in the presence of larger ϵ_n values and like L_2 loss function when ϵ_n is small makes it more resilient to outliers, and at the same time, it offers a faster convergence rate. Moreover, the log-cosh loss function has the advantage of being differentiable at all points, unlike the L_1 loss function. This ensures that the training process is stable

and smooth, which is essential for achieving accurate and reliable results. In summary, the log-cosh loss function has unique properties that can make it a more desirable option than the L_1 or L_2 loss functions in certain situations. By balancing the benefits of both L_1 and L_2, the log-cosh loss function can be more robust to outliers, while still providing faster convergence and better differentiability than L_1. These qualities can make it an effective tool for handling uncertainty and noise in machine learning models.

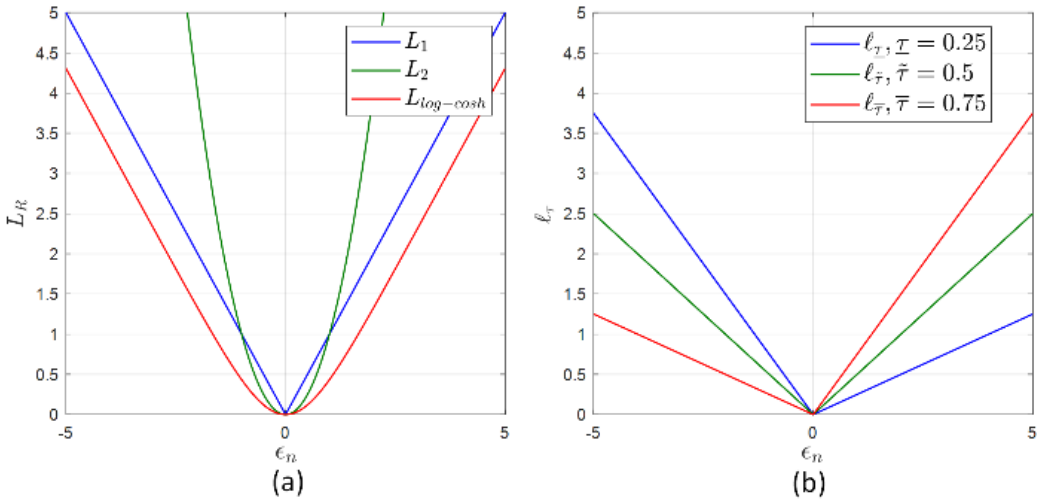


Figure 4.2 : Illustration of the (a) various empirical risk measures, (b) tilted losses for different levels (τ) of quantile.

The other part of our composite loss is the tilted loss which we use for generating PIs. The tilted loss function is utilized to create a boundary that encompasses the anticipated level of uncertainty $\varphi = [\underline{\tau}, \overline{\tau}]$ and is expressed as:

$$\ell_\tau\left(q^\tau(x_n)\right) = \underline{\ell}_\tau + \overline{\ell}_\tau \tag{4.13}$$

Here,

$$\begin{aligned} \underline{\ell}_\tau &= \sum_{n=1}^{N} \max(\underline{\tau}(y_n - \underline{y}(\boldsymbol{x}_n)), (\underline{\tau} - 1)(y_n - \underline{y}(\boldsymbol{x}_n))) \\ \overline{\ell}_\tau &= \sum_{n=1}^{N} \max(\overline{\tau}(y_n - \overline{y}(\boldsymbol{x}_n)), (\overline{\tau} - 1)(y_n - \overline{y}(\boldsymbol{x}_n))) \end{aligned} \tag{4.14}$$

Figure 4.2b depicts the graphs of three different tilted loss functions which can be listed as $\underline{\ell}_\tau$ has a quantile level as $\underline{\tau} = 0.25$, $\overline{\ell}_\tau$ is defined with a quantile level as $\overline{\tau} =$

0.75, and $\tilde{\ell}_\tau$ is defined with $\tau = 0.5$ (also known as the median quantile [49]), for a range of ϵ_n values from -5 to 5 with a step size of 0.1. It is important to note that, for the purpose of illustration, the assumption is made that ϵ_n is equal to both $\underline{\epsilon}_n$ and $\overline{\epsilon}_n$ ($\epsilon_n = \underline{\epsilon}_n = \overline{\epsilon}_n$). The tilted loss function is used to create a boundary that encompasses the expected amount of uncertainty, which is defined by the range $\varphi = [\underline{\tau}, \overline{\tau}]$.

The graphs show how the different tilted loss functions behave for varying values of ϵ_n. The $\underline{\ell}_\tau$ loss function has a steeper slope and penalizes more severely for $\epsilon_n < 0$ values outside the expected uncertainty range, whereas the $\overline{\ell}_\tau$ loss function has steeper slope and more penalty values for $\epsilon_n > 0$. The $\tilde{l}_\tau$ loss function, on the other hand, has a similar slope values for all ϵ_n values from -5 to 5. Overall, these graphs provide a useful visual representation of the behavior of tilted loss functions and how they can be used to effectively manage uncertainty in machine learning models.

The function $l_\tau(.)$ described in equation 4.13 incorporates the reduced set $(\underline{y}(\boldsymbol{x}), \overline{y}(\boldsymbol{x}))$ to train an IT2-FLS that can cover the uncertainty defined by $\varphi = [\underline{\tau}, \overline{\tau}]$. By setting $\underline{\tau} = 0.1$ and $\overline{\tau} = 0.9$, the training is conducted with a PI (possibility interval) of $I^{\varphi} = 80\%$. This means that the training is carried out in a way that ensures the IT2-FLS can handle uncertainty up to 80% confidence level, which is defined by the interval $\varphi = [0.1, 0.9]$. Figure 4.2b illustrates the functions $\underline{\ell}_\tau$, $\overline{\ell}_\tau$ and $\tilde{\ell}_\tau$ for $\epsilon_n \in [-5,5]$ for illustrative purposes. The function $l_\tau(.)$ is used to generate an envelope that covers the expected amount of uncertainty defined by the interval $\varphi = [\underline{\tau}, \overline{\tau}]$. Moreover, the median quantile is obtained when the quantile level τ is selected as $\tau = 0.5$.

When defining the loss function L in P_3, a hyperparameter λ needs to be defined to balance between accuracy and uncertainty representation. In order to weigh equally between the risk measure $L_R(\cdot)$ and the tilted loss function $l_\tau(.)$, we set $\lambda = 1$. To avoid negatively affecting the learning process due to hard-parameter sharing ($\boldsymbol{\theta}$), we need to use an $L_R(\cdot)$ that has similar characteristics to $l_\tau(.)$. In Figure 4.2, we can see that the log-cosh loss function shares similar characteristics with $l_\tau(.)$, making it a suitable choice as the empirical risk measure in P_3.

By employing the log-cosh loss function as the empirical risk measure, P_3 can construct a balance between accuracy and uncertainty representation in its learning process. This is achieved by equal weighting between $L_R(\cdot)$ and $l_\tau(.)$ through the hyperparameter λ. Since hard-parameter sharing can be problematic in deep learning models, it is crucial to use a suitable $L_R(\cdot)$ that is consistent with the tilted loss function $l_\tau(.)$ to avoid negative impacts on the learning process. The similarity between the characteristics of the log-cosh loss function and $l_\tau(.)$ makes it a good choice for the empirical risk measure in P_3, providing a stable and effective way of handling uncertainty in the learning process. In this context, we can define our novel composite loss as follows:

$$L = L_{\log-\cosh} + \ell_\tau\left(q^\tau(x_n)\right) \tag{4.15}$$

Remark: It should be noted that we suggest to use log-cosh as the empirical risk function in our composite loss. However, we will also analyze the performances of the L_1 and L_2 functions in case of their selected as empirical risk functions and provide detail result in the experimental studies section which is section 6.2.2.

From equations 4.12 to 4.15, it is easily seen that we use both the crisp output $y(\boldsymbol{x})$ and the type-reduced set $\tilde{Y} = [\underline{y}(\boldsymbol{x}), \overline{y}(\boldsymbol{x})]$ of IT2-FLS.

- Looking at the problem from a multi-task perspective, we can treat the loss function as a multi-objective one, consisting of both $L_{log-cosh}$ and $\ell_\tau\left(q^\tau(x_n)\right)$. Additionally, $\ell_\tau\left(q^\tau(x_n)\right)$ can be viewed as a form of regularization that helps prevent overfitting and promote the development of more robust prediction models. By incorporating both $L_{log-cosh}$ and $\ell_\tau\left(q^\tau(x_n)\right)$ into the loss function, we can set a balance between accuracy and uncertainty representation. Furthermore, using $\ell_\tau\left(q^\tau(x_n)\right)$ as a regularization term can help improve the model's ability to generalize to new data and perform well in real-world scenarios.
- Our objective is to train IT2-FLSs that provide accurate predictions while also accounting for uncertainty. To achieve this, we want the bounds of the IT2-FLS to cover an expected amount of uncertainty, which we define using the values of $\underline{\tau}$ and $\overline{\tau}$. These parameters are used to express the predefined level of

uncertainty that the antecedent and consequent MFs of the IT2-FLS must capture. By setting a predetermined level of uncertainty and ensuring that the IT2-FLS captures it, we can build more robust and reliable prediction models.

4.4 Composite Learning with Parametrization Tricks

In this section of this thesis, we will present the parametrization tricks for the parametric IT2-FLSs in that way we can use DL based frameworks as the parameter optimizers for these parametric models. Inspiring from the connection between FLSs and NNs described in literature [33], we approached the parameter learning task of IT2-FLS using optimization algorithms available in DL frameworks, as illustrated in Figure 4.3. Specifically, we utilized automatic differentiation and trained the IT2-FLS using a DL optimizer such as Adam. The training Algorithm, which also involves initializing parameters for the HS-IVL, is presented in Figure 4.4.

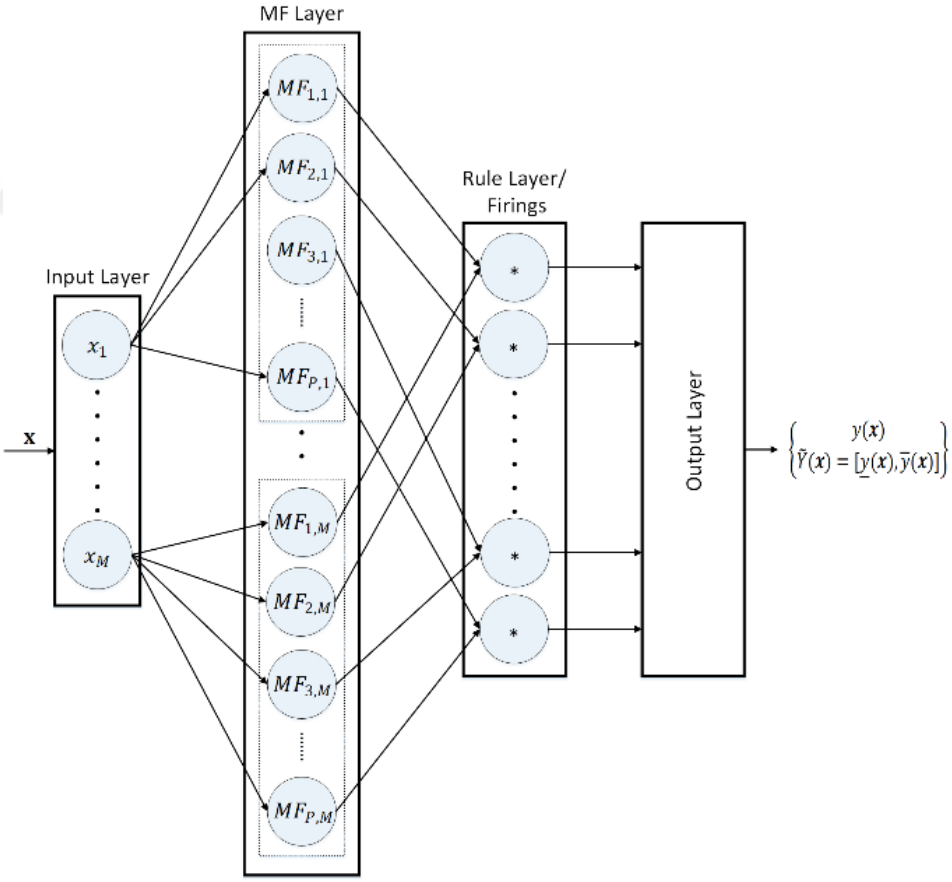


Figure 4.3 : Illustration of the NN representation of the proposed IT2-FLSs.

Input: N training samples $\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$ and the conditional quantile range $\varphi = [\underline{\tau}, \overline{\tau}]$
...P, the number of rules;
T, the maximum number of training iterations;
...N_{bs}, mini-batch size;
Set CSCM ∈ {KM, NT, SM}
Define t-norm operator ∩
Output: Learnable parameter set $\boldsymbol{\theta}$.
for $m = 1$ to M **do**
Calculate min and max values of $\{\boldsymbol{x}_{n,m}\}_{n=1}^{N}$;
Calculate standard deviation values of $\{\boldsymbol{x}_{n,m}\}_{n=1}^{N}$;
for $p = 1$ to P **do**
Initialize $c_{p,m}$ uniformly between the min. and max.;
Initialize $\underline{\sigma}_{p,m}, \overline{\sigma}_{p,m}$ with the same standard deviation values of $\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$;
Initialize $h_{p,m}$ uniformly between [0, 1];
end for
end for
Initialize $\widehat{\boldsymbol{\theta}}_{\boldsymbol{C}}^{0} = (a_{p,m}, a_{0_{p,m}}, b_{p,m}, b_{0_{p,m}})$ randomly around **0**;
Construct $\widehat{\boldsymbol{\theta}}_{\boldsymbol{A}}^{0} = (c_{p,m}, \underline{\sigma}_{p,m}, \overline{\sigma}_{p,m}, h_{p,m})$;
Construct $\widehat{\boldsymbol{\theta}}^{0} = \{\widehat{\boldsymbol{\theta}}_{\boldsymbol{A}}^{0}, \widehat{\boldsymbol{\theta}}_{\boldsymbol{C}}^{0}\}$;
for $t = 1$ to T **do**
Select randomly N_{bs} samples;
for $n = 1$ to N_{bs} **do**
for $p = 1$ to P **do**
Do the tricks for $\widehat{\boldsymbol{\theta}}_{\boldsymbol{A}}$ via (31) and $\widehat{\boldsymbol{\theta}}_{\boldsymbol{C}}$ via (32)
Find firing levels $\tilde{F}_p = [\underline{f}_p, \overline{f}_p]$, compute $\underline{f}_p$ via (10) and $\overline{f}_p$ via (11)
end for
Compute $y(x_n), \tilde{Y}(x_n) = [\underline{y}(x_n), \overline{y}(x_n)]$,
if CSCM == "KM"
Calculate $\underline{y}(x_n)$ via (12) and $\overline{y}(x_n)$ via (13)
else if CSCM == "NT"
Calculate $\underline{y}(x_n)$ by (15) and $\overline{y}(x_n)$ by (16)
else if CSCM == "SM"
Calculate $\underline{y}(x_n)$ and $\overline{y}(x_n)$ via (17)
end if
$y(x_n)$ via (18)
end for
Compute L via (28), calculate the gradient loss $\partial \boldsymbol{L}/\partial \widehat{\boldsymbol{\theta}}^{t}$
Update $\widehat{\boldsymbol{\theta}}^{t}$ via Adam optimizer
end for
$\widehat{\boldsymbol{\theta}}^{*} = \arg\min_{\boldsymbol{\theta}}(L)$
Transform $\widehat{\boldsymbol{\theta}}^{*}$ into $\boldsymbol{\theta}^{*}$by using (31), (32)
Return $\boldsymbol{\theta} = \boldsymbol{\theta}^{*}$

Figure 4.4 : Illustration of the composite learning algorithm for HS-IVL.

When training IT2-FLSs, it is important to keep in mind that the parameters that can be learned are the MFs that define the system. As a result, it is crucial that these MFs produce FSs that adhere to their respective definitions. For the antecedent IT2-FSs, we must guarantee $\overline{\mu}_{\tilde{A}_{p,m}}(x_m) \geq \underline{\mu}_{\tilde{A}_{p,m}}(x_m)$. Thus, we have the following constraints:

$$0 \leq h_{p,m} \leq 1 \quad \underline{\sigma}_{p,m} \leq \overline{\sigma}_{p,m} \tag{4.16}$$

For IV/IVL rule consequents, we must satisfy $\overline{y}_p \geq \underline{y}_p$ for $\forall\, p$. Therefore, $0 \leq b_{p,0}$ is satisfied for the IV consequents whereas for IVL ones, we must guarantee for $\forall\, p, m$:

$$0 \leq b_{p,m} \quad 0 \leq b_{p,0} \tag{4.17}$$

Note that there are no constraints on $a_{p,m}$ and $a_{p,0}$. The constraint parameter space $\boldsymbol{\theta} \in C$, as defined in equations (4.16) and (4.17), poses a challenge in the learning of IT2-FLSs as most DL frameworks such as PyTorch, TensorFlow, and Matlab only support unconstrained optimization. Hence, to address this problem, we suggest using the following parameterization techniques for $\boldsymbol{\theta_A}$:

$$\begin{gathered} h_{r,m} = sig(h'_{r,m}) \\ \overline{\sigma}_{p,m} = \sigma'_{p,m} + |\Delta_{\mathrm{A}}| \quad \underline{\sigma}_{p,m} = \sigma'_{p,m} - |\Delta_{\mathrm{A}}| \end{gathered} \tag{4.18}$$

where $sig(.)$ is the sigmoid function and $\{h'_{r,m}, \sigma'_{p,m}, \Delta_{\mathrm{A}}\} \in [-\infty, \infty]$ are the new and unbounded optimization variables. Similarly, we define the following parametrizations for $\boldsymbol{\theta_C}$:

$$\begin{aligned} b_{p,0} &= |b'_{p,0}| \\ b_{p,m} &= |b'_{p,m}| \end{aligned} \tag{4.19}$$

where $\{b'_{p,0}, b'_{p,m}\} \in [-\infty, \infty]$.

With the proposed parameterization tricks for $\boldsymbol{\theta_A}$ and $\boldsymbol{\theta_C}$, we were able to remove the constraint parameter space $\boldsymbol{\theta} \in C$, making it possible to apply DL methods to the learning problem. Specifically, we were able to use optimization algorithms in PyTorch, TensorFlow, and Matlab that only support unconstrained optimization. This allowed us to train the IT2-FLSs effectively and efficiently, achieving high accuracy performance while maintaining a desired level of uncertainty representation. By removing the constraints, we were able to take full advantage of the capabilities of DL frameworks for the learning of IT2-FLSs.

5. TOWARDS RELIABLE UNCERTAINTY QUANTIFICATION AND HIGH PRECISION WITH GENERAL TYPE-2 FUZZY SYSTEMS

One of the primary reason for utilizing T2- FLSs, also known as high-order FLS, is their ability to quantify uncertainty through the FOU embedded within their MFs. This additional degree of freedom allows for a more comprehensive representation of uncertainty [69]. However, similar to existing approaches for T1-FLS, the focus of learning methods has primarily been on improving precision [44, 45, 47, 104, 105]. Consequently, it can be argued that the FOU parameters in T2-FLS are often treated as additional trainable parameters, similar to increasing the number of neurons or layers in a deep neural network, with the only aim to enhance the accuracy rather than as an opportunity to quantify uncertainty.

However, in one of our recent study [49], we introduced a novel DL framework specifically designed for IT2-FLS. This framework employs a composite loss function that explicitly leverages the Type Reduced (TR) set and the system's output to generate pointwise estimates along with their Prediction Intervals (PI). Unlike previous approaches, this DL framework recognizes and capitalizes on the inherent uncertainty quantification capabilities of IT2-FLS, thereby offering a more comprehensive approach to learning. By explicitly considering the TR set and PI, this framework aims to provide both accurate predictions and meaningful uncertainty measures, bridging the gap between precision-focused learning methods and the true potential of IT2-FLS in quantifying uncertainty.

In the “Towards Reliable Uncertainty Quantification and High Precision with General Type-2 Fuzzy Systems”, we will introduce a novel learning approach for GT2-FLSs based on the α-plane representation which is introduced in section 2.2.2. Here, our approach aims to enhance the accuracy of pointwise predictions and generate reliable PIs by leveraging the shape and size of the SMFs of the GT2-FLS. We propose a novel composite loss function that combines an uncertainty quantification-focused term and an accuracy-focused term. The uncertainty-focused loss term explicitly utilizes the Type-Reduced (TR) set of the IT2-FLS associated with the $\alpha_0 = 0$ plane, which represents the FOU. By imposing constraints on the SMF size parameters of the GT2-FS, we effectively quantify the uncertainty and facilitate the learning of PIs. On the other hand, the accuracy-focused term offers two alternative loss functions. In one

approach, we utilize the aggregated output of the GT2-FLSs, while in the other, we only consider the output associated with the $\alpha_K = 1$ level. In both cases, we enforce the SMF shape parameters of the GT2-FS to enable precise pointwise predictions. By assigning different roles to the IT2-FLSs associated with different α-planes within the composite loss function, we allow for partially independent learning of the GT2-FSs. This approach enables the capture of uncertainty while maintaining high accuracy, as the output of the $\alpha_0 = 0$ plane does not contribute to the overall output calculation of the GT2-FLS.

To efficiently handle high-dimensional and complex data, we also propose a DL approach for GT2-FLSs. This approach formulates an unconstrained learning problem that adheres to the definitions of GT2-FSs through parameterization techniques. To demonstrate the effectiveness of our proposed learning approach, we conduct statistical comparative analyses using benchmark datasets. In the experiment section, the results highlight the potential of learning GT2-FLSs as a promising solution for reliable uncertainty quantification in real-world applications, offering both high precision in pointwise predictions and accurate estimation of uncertainty.

5.1 GT2-FLSs: Structure and Learning Parameters

In this section, we present the general structure of the deployed GT2-FLSs and define their learnable parameters.

5.1.1 Internal structure of the GT2-FLSs

The formulation of the rules for GT2-FLS for uncertainty capturing can be constructed for an input $\boldsymbol{x} = (x_1, x_2, \dots, x_M)^T$ and a single output y_{GT2}. , as we provided preliminary theoretical background in section 2.2.2, the rule base is composed of P rules ($\mathrm{p} = 1, 2, \dots, \mathrm{P}$) and defined in following:

$$Rule_p : \text{If } x_1 \text{ is } \tilde{A}_{p,1} \text{ and ... } x_M \text{ is } \tilde{A}_{p,M} \text{ Then } y_{GT2} \text{ is } s_p \tag{5.1}$$

where s_p represents the consequent MFs and are defined as:

$$\mathrm{s_p} = \sum_{m=1}^{M} a_{p,m} x_m + a_{p,0} \tag{5.2}$$

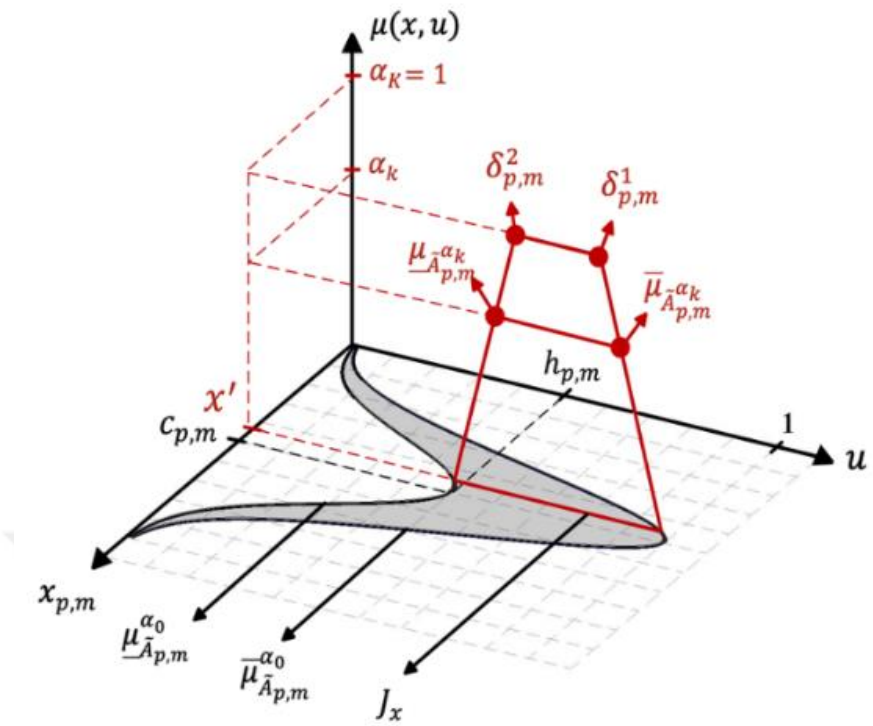


Figure 5.1 : Illustration of a GT2-FS with an α-plane.

As shown in Figure 5.1, the antecedent MFs are defined with GT2-FSs $\tilde{A}_{p,m}$ that can be defined as a collection of α-planes (α_k) as follows [50, 69]:

$$\tilde{A}_{p,m} = \bigcup_{\alpha_k \in [0,1]} \tilde{A}^{\alpha_k}_{p,m} \tag{5.3}$$

where $\tilde{A}^{\alpha_k}_{p,m}$ is the α-plane of $\tilde{A}_{p,m}$ associated with $\alpha_k \in [0,1]$. If α_k is uniformly distributed, then we define α_k as $\alpha_k = k/K$, $k = 0, 1, \ldots, K$, and thus the total number of α-planes is $K + 1$. This representation allows defining the Upper MF (UMF) and Lower MF (LMF) grades for an α_k-plane as:

$$\begin{aligned} \underline{\mu}_{\tilde{A}^{\alpha_k}_{p,m}} &= \underline{\mu}^{\alpha_0}_{\tilde{A}_{p,m}} + \alpha_k \left(\overline{\mu}^{\alpha_0}_{\tilde{A}_{p,m}} - \underline{\mu}^{\alpha_0}_{\tilde{A}_{p,m}} \right) \delta^1_{p,m} \\ \overline{\mu}_{\tilde{A}^{\alpha_k}_{p,m}} &= \overline{\mu}^{\alpha_0}_{\tilde{A}_{p,m}} - \alpha_k \left(\overline{\mu}^{\alpha_0}_{\tilde{A}_{p,m}} - \underline{\mu}^{\alpha_0}_{\tilde{A}_{p,m}} \right) \left(1 - \delta^2_{p,m}\right) \end{aligned} \tag{5.4}$$

where $\left\{\delta^1_{p,m}, \delta^2_{p,m} : \delta^1_{p,m} \geq \delta^2_{p,m}, \forall\, p, m\right\} \in [0,1]$ are the parameters that define the shape of the SMFs. In this study, for the sake of simplicity, we set them as $\delta^1_{p,m} = \delta^1_m$ and $\delta^2_{p,m} = \delta^2_m \; \forall\, p$. Besides, $\overline{\mu}^{\alpha_0}_{\tilde{A}_{p,m}}$ and $\underline{\mu}^{\alpha_0}_{\tilde{A}_{p,m}}$, presented in equation 5.4, are the UMF

and LMF of the base α-plane α_0, i.e. the MFs of the primary MF of the GT2-FSs. They are defined as:

$$\begin{aligned} \overline{\mu}_{\tilde{A}_{p,m}}^{\alpha_0}(x_m) &= \exp\left(-\left(x_m - c_{p,m}\right)^2 / 2\,\sigma_{p,m}^2\right) \\ \underline{\mu}_{\tilde{A}_{p,m}}^{\alpha_0}(x_m) &= h_{p,m} \exp\left(-\left(x_m - c_{p,m}\right)^2 / 2\,\sigma_{p,m}^2\right) \end{aligned} \tag{5.5}$$

Here, $c_{p,m}$ is the center and $\sigma_{p,m}$ is the standard deviation while $h_{p,m} \in [0,1]\ \forall p,m$ indicates the height of LMF which is the only parameter that defines the FOU of the GT2-FS on the primary MF (J_x) or the size of the SMF as illustrated in Figure 5.1.

The α-plane representation simplifies the output calculation of the GT2-FLSs as follows:

$$\mathrm{y}_{\mathrm{GT2}} = \frac{\sum_{k=0}^{K} \mathrm{y}^{\alpha_k}(\boldsymbol{x})\,\alpha_k}{\sum_{k=1}^{K} \alpha_k} \tag{5.6}$$

where $\mathrm{y}^{\alpha_k}(\boldsymbol{x})$ is the output of an IT2-FLS associated with an α-plane α_k (α_k-IT2-FLS) that is defined as:

$$y^{\alpha_k}(\boldsymbol{x}) = \frac{\underline{y}^{\alpha_k}(\boldsymbol{x}) + \overline{y}^{\alpha_k}(\boldsymbol{x})}{2} \tag{5.7}$$

Here, $\underline{y}^{\alpha_k}(\boldsymbol{x})$ and $\overline{y}^{\alpha_k}(\boldsymbol{x})$ are the bounds of the TR set which are obtained via a Center of Sets Calculation Method (CSCM) that is defined as*:*

$$\tilde{\mathrm{Y}}^{\alpha_k}(\mathrm{y}_1, \ldots, \mathrm{y}_\mathrm{P}, \tilde{\mathrm{F}}_1^{\alpha_k}, \ldots, \tilde{\mathrm{F}}_\mathrm{P}^{\alpha_k}) = \left[\underline{y}^{\alpha_k}(\boldsymbol{x}), \overline{y}^{\alpha_k}(\boldsymbol{x})\right] \tag{5.8}$$

where $\tilde{F}_p^{\alpha_k} = \left[\underline{f}_p^{\alpha_k}(\boldsymbol{x}), \overline{f}_p^{\alpha_k}(\boldsymbol{x})\right]$ represents the firing interval of the pth rule and is defined as:

$$\begin{aligned} \underline{f}_p^{\alpha_k}(\boldsymbol{x}) &= \underline{\mu}_{\tilde{A}_{p,1}^{\alpha_k}}(\mathrm{x}_1) \cap \underline{\mu}_{\tilde{A}_{p,2}^{\alpha_k}}(\mathrm{x}_2) \cap \ldots \cap \underline{\mu}_{\tilde{A}_{p,M}^{\alpha_k}}(\mathrm{x}_\mathrm{M}) \\ \overline{f}_p^{\alpha_k}(\boldsymbol{x}) &= \overline{\mu}_{\tilde{A}_{p,1}^{\alpha_k}}(\mathrm{x}_1) \cap \overline{\mu}_{\tilde{A}_{p,2}^{\alpha_k}}(\mathrm{x}_2) \cap \ldots \cap \overline{\mu}_{\tilde{A}_{p,M}^{\alpha_k}}(\mathrm{x}_\mathrm{M}) \end{aligned} \tag{5.9}$$

Here ∩ denotes the t-norm operator that can be the product or minimum operator. In order to calculate the crisp output of the GT2-FLSs, we consider three CSCM methods

as we mentioned earlier in section 2.2.1. It should be noted that there are various type reduction methods are available in literature. KM type reduction [61] and NT [51], which are summarized in section 2.2.1 in equations 2.27 to 2.31, type reduction methods are the widely used methods in literature. Therefore, we also picked these two methods in this thesis to be employ as the type reducer for the employed GT2-FLSs. The KM type-reduction and NT type reduction methods can be redifne, respectively, for GT2-FLSs as the following; the KM CSCM for GT2-FLS [61]:

$$\underline{y}^{\alpha_k}(\boldsymbol{x}) = \frac{\sum_{p=1}^{L} \underline{f}_p^{\alpha_k}(\boldsymbol{x})\, \mathrm{s_p} + \sum_{p=L+1}^{P} \overline{f}_p^{\alpha_k}(\boldsymbol{x}) \mathrm{s_p}}{\sum_{p=1}^{L} \underline{f}_p^{\alpha_k}(\boldsymbol{x}) + \sum_{p=L+1}^{P} \overline{f}_p^{\alpha_k}(\boldsymbol{x})} \tag{5.10}$$

$$\overline{y}^{\alpha_k}(\boldsymbol{x}) = \frac{\sum_{p=1}^{R} \underline{f}_p^{\alpha_k}(\boldsymbol{x})\, \mathrm{s_p} + \sum_{p=R+1}^{P} \overline{f}_p^{\alpha_k}(\boldsymbol{x}) \mathrm{s_p}}{\sum_{p=1}^{R} \underline{f}_p^{\alpha_k}(\boldsymbol{x}) + \sum_{p=R+1}^{P} \overline{f}_p^{\alpha_k}(\boldsymbol{x})} \tag{5.11}$$

The NT CSCM [51]:

$$y^{\alpha_k}(\boldsymbol{x}) = \frac{\sum_{p=1}^{P} \left[\underline{f}_p^{\alpha_k}(\boldsymbol{x}) \mathrm{s_p} + \overline{f}_p^{\alpha_k}(\boldsymbol{x}) \mathrm{s_p} \right]}{\sum_{r=1}^{R} \underline{f}_p^{\alpha_k}(\boldsymbol{x}) + \sum_{r=1}^{R} \overline{f}_p^{\alpha_k}(\boldsymbol{x})} \tag{5.12}$$

We can derive the following equivalent TR set [49]:

$$\underline{y}^{\alpha_k}(\boldsymbol{x}) = \frac{2\ \sum_{p=1}^{P} \underline{f}_p^{\alpha_k}(\boldsymbol{x}) \mathrm{s_p}}{\sum_{p=1}^{P} \underline{f}_p^{\alpha_k}(\boldsymbol{x}) + \sum_{p=1}^{P} \overline{f}_p^{\alpha_k}(\boldsymbol{x})} \tag{5.13}$$

$$\overline{y}^{\alpha_k}(\boldsymbol{x}) = \frac{2\ \sum_{p=1}^{P} \underline{f}_p^{\alpha_k}(\boldsymbol{x}) \mathrm{s_p}}{\sum_{p=1}^{P} \underline{f}_p^{\alpha_k}(\boldsymbol{x}) + \sum_{p=1}^{P} \overline{f}_p^{\alpha_k}(\boldsymbol{x})} \tag{5.14}$$

5.1.2 Learnable parameter sets

The learnable parameters ($\boldsymbol{\theta_{GT2}}$) of the GT2-FLSs also the parameters of antecedent ($\boldsymbol{\theta_{GT2-A}}$) and consequents ($\boldsymbol{\theta_{GT2-C}}$) MFs. All of these parameters can be trained with a derivative-based optimization approach, Adam is used for this work, or the parameter is predefined by the user before the training. However, they must satisfy some constraints similar to the parametric IT2-FLSs, thus in the following sections (section 5.2.1) we will define them.

The learnable parameter sets will be formed with respect to the selection of the parametric models as we explained in section 4.2.2. For the parametric GT-FLS, we use a H type IT2-FS for its primary MFs. In this context, the learnable parameters of the GT2-FLS ($\boldsymbol{\theta_{GT2}}$) are defined with two subsets, which are learnable parameters of antecedents $\boldsymbol{\theta_{GT2-A}}$ and consequents $\boldsymbol{\theta_{GT2-A}}$.

The antecedent parameters set is $\boldsymbol{\theta_{GT2-A}} = \{\boldsymbol{c}, \boldsymbol{\sigma}, \boldsymbol{h}, \boldsymbol{\delta^1}, \boldsymbol{\delta^2}\}$ where $\boldsymbol{c} = \left(c_{1,1}, \dots, c_{P,M}\right)^T \in \mathbb{R}^{P\times M}$, $\boldsymbol{\sigma} = \left(\sigma_{1,1}, \dots, \sigma_{P,M}\right)^T \in \mathbb{R}^{P\times M}$, $\boldsymbol{h} = \left(h_{1,1}, \dots, h_{P,M}\right)^T \in \mathbb{R}^{P\times M}$,$\boldsymbol{\delta^1} = (\delta_1^1, \dots, \delta_M^1)^T \in \mathbb{R}^{1\times M}$, and $\boldsymbol{\delta^2} = (\delta_1^2, \dots, \delta_M^2)^T \in \mathbb{R}^{1\times M}$. On the other hand, the consequent parameters set is $\boldsymbol{\theta_C} = \{\boldsymbol{a}, \boldsymbol{a_0}\}$ where $\boldsymbol{a} = (\boldsymbol{a}_{1,1}, \dots, \boldsymbol{a}_{P,M})^T \in \mathbb{R}^{P\times M}$ and $\boldsymbol{a_0} = \left(a_{1,0}, \dots, a_{P,0}\right)^T \in \mathbb{R}^{P\times 1}$. Thus, $\boldsymbol{\theta_{GT2}}$ has in total $(3P+2)M + P(M+1)$ learnable parameters.

5.2 The Proposed Learning Approach for the GT2-FLSs

Here, we first present a novel composite loss function to learn GT2-FLS that results in a high accuracy performance and reliable PIs then present the proposed learning framework based on DL optimizers. The proposed approach is presented for a dataset $\{\boldsymbol{x_n}, y_n\}_n^N$ defined with $\boldsymbol{x_n} = (x_{n,1}, x_{n,2}, \dots, x_{n,M})^T$.

5.2.1 The composite loss function for learning GT2-FLSs

In this section we present the proposed composite loss function for the proposed learning approach for GT2-FLSs. Here, our purpose is to train a GT2-FLS model that can accurately predict outputs while also handling uncertainty. To achieve this, we formulate a multi-objective optimization problem for a dataset of N samples $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$ where x_n is an M-dimensional input vector and y_n is the corresponding output. The goal is to minimize two objectives: the first is the empirical risk, which measures the model's mean error in predicting y_n for each input x_n; the second is the tilted loss, which considers the model's ability to capture the uncertainty in the output. We aim to find the optimal values for the learnable parameters that minimize both objectives simultaneously. This is a challenging optimization problem because the total number of parameters increases with the model's flexibility, and the problem is further complicated by the GT2-FLSs required to handle uncertainty effectively. In our previous section 4, we proposed the following loss function to learn the parameters

of an IT2-FLS ($\boldsymbol{\theta_{IT2}}$) by explicitly utilizing its defuzzified output $y_{IT2}(x_n)$ and TR set $[\underline{y}(x_n), \overline{y}(x_n)]$:

$$\min_{\boldsymbol{\theta}\in\boldsymbol{\mathcal{C}}} L = \sum_{n=1}^{N} \left[L_R\left(f(\mathrm{x_n})\right) + \ell_\tau\left(q^\tau(\mathrm{x_n})\right)\right] \tag{5.15}$$

where $\boldsymbol{\mathcal{C}}$ is the constraint set that is defined to end up with convex IT2-FSs ($\boldsymbol{\mathcal{C}_{IT2}}$), $L_R(\cdot)$ is the empirical risk function defined for accuracy purposes that is defined as:

$$L_R = \frac{1}{N}\sum_{n=1}^{N} \log\left(\cosh(\epsilon)\right) \tag{5.16}$$

with the prediction error ϵ for the IT2-FLS (ϵ_{IT2})

$$\epsilon_{IT2} = \mathrm{y_n} - \mathrm{y_{IT2}(x_n)} \tag{5.17}$$

and $\ell_\tau(\cdot)$ is the tilted loss defined for learning an envelope to cover the expected amount of uncertainty which is defined with a quantile interval as $\varphi = [\underline{\tau}, \overline{\tau}]$. $\ell_\tau(\cdot)$ is defined with lower ($\underline{\tau}$) and upper ($\overline{\tau}$) quantile level as follows:

$$\ell_\tau\left(q^\tau(\mathrm{x_n})\right) = \underline{\ell}_\tau + \overline{\ell}_\tau$$

$$\underline{\ell}_\tau = \sum_{n=1}^{N} \max(\underline{\tau}(y_n - \underline{y}(x_n), (\underline{\tau} - 1)(y_n - \underline{y}(x_n)) \tag{5.18}$$

$$\overline{\ell}_\tau = \sum_{n=1}^{N} \max(\overline{\tau}(\mathrm{y_n} - \overline{\mathrm{y}}(\mathrm{x_n}), (\overline{\tau} - 1)(\mathrm{y_n} - \overline{\mathrm{y}}(\mathrm{x_n}))$$

If we set $\underline{\tau} = 0.1$ and $\overline{\tau} = 0.9$, then as seen in equation 5.18 enforces the learning to result in a PI of $I^{\varphi} = \%80$.

Remark: The derivations of $y_{IT2}(x_n)$ and $[\underline{y}(x_n), \overline{y}(x_n)]$ alongside $\boldsymbol{\theta_{IT2}}$ is presented in previous section. Yet, they can be also derived via the definitions in Section 5.1. by setting $\delta_m^1 = 0$ and $\delta_m^2 = 1\ \forall m$ since the GT2-FSs reduce to IT2-FSs.

Considering the loss function in equation 5.15, it is evident that even though $y_{IT2}(x_n)$ and $[\underline{y}(x_n), \overline{y}(x_n)]$ are explicitly incorporated, the learning performance of IT2-FLS may not be satisfactory or reliable. This is primarily due to the direct influence of θ_{IT2} on both the accuracy and uncertainty coverage, leading to significant variation since the learnable initiate with different random values. To address this

limitation and enhance the reliability of pointwise accuracy and uncertainty quantification, our new approach capitalizes on the additional design flexibility provided by GT2-FLS. Specifically, we focus on leveraging the SMF parameters $\boldsymbol{\delta^1}$ and $\boldsymbol{\delta^2}$ to increase the dependability of the results. By harnessing the potential of these parameters, we can fine-tune the GT2-FLS to achieve better accuracy and more accurate quantification of uncertainty.

In the new approach our study, we introduce distinct roles for the α_k -IT2-FLS within the proposed loss function. This approach allows us to optimize the GT2-FLS by assigning specific tasks to different parts of the IT2-FLS. By doing so, we can effectively optimize the SMF parameters $\boldsymbol{\delta^1}$ and $\boldsymbol{\delta^2}$, enabling us to strike a balance between accuracy and uncertainty coverage. Through this strategy, we aim to mitigate the limitations observed in IT2-FLS and improve the overall reliability of pointwise accuracy and uncertainty estimation. By capitalizing on the flexibility offered by GT2-FLS and tailoring the loss function accordingly, our objective is to provide a more robust and dependable framework for achieving accurate pointwise predictions and quantifying uncertainty.

- For the uncertainty quantification-focused part, we utilize the TR set of $\alpha_0 -$ IT2-FLS as follows:

$$\ell_\tau^{\alpha_0}\left(q^\tau(\mathrm{x_n})\right) = \underline{\ell}_\tau^{\alpha_0} + \overline{\ell}_\tau^{\alpha_0} \tag{5.19}$$

$$\underline{\ell}_\tau^{\alpha_0} = \sum_{n=1}^{N} \max(\underline{\tau}(\underline{\epsilon}^{\alpha_0}), (\underline{\tau}-1)\underline{\epsilon}^{\alpha_0}) \tag{5.20}$$

Where $\underline{\epsilon}^{\alpha_0} = y_n - \underline{y}^{\alpha_0}(x_n)$

$$\overline{\ell}_\tau^{\alpha_0} = \sum_{n=1}^{N} \max(\overline{\tau}(\overline{\epsilon}^{\alpha_0}), (\overline{\tau}-1)\overline{\epsilon}^{\alpha_0}) \tag{5.21}$$

$$\overline{\epsilon}^{\alpha_0} = y_n - \overline{y}^{\alpha_0}(x_n) \tag{5.22}$$

Thus, as the output of $\alpha_0 - \text{IT2-FLS}$ has no contribution to $y_{GT2}(x_n)$ since $\alpha_0 = 0$, the main role of $\alpha_0 - \text{IT2-FLS}$ is to quantify the uncertainty via the size of the SMFs.

- For the accuracy-focused loss part (L_R), a natural choice would be the direct usage of $y_{GT2}(x_n)$ within the error term in (19) as follows:

$$\epsilon_{GT2-1} = y_n - y_{GT2}(x_n) \tag{5.23}$$

On the other hand, instead of using $y_{GT2}(x_n)$that is a collection of all aggregation of α-plane outputs, we suggest only using the defuzzified output of α-IT2-FLS associated with the level $\alpha_K = 1$ ($y^{\alpha_K}(x_n)$). Thus, we define:

$$\epsilon_{GT2-2} = y_n - y^{\alpha_K}(x_n) \tag{5.24}$$

Therefore, using only the $\alpha_K = 1$ plane output of the GT2-FLS in empirical risk gives the flexibility to learn the SMF parameters $\boldsymbol{\delta^1}$ and $\boldsymbol{\delta^2}$ to enhance solely the accuracy performance within $J_x = [\underline{\mu}_{\tilde{A}_{p,m}}^{\alpha_0}, \overline{\mu}_{\tilde{A}_{p,m}}^{\alpha_0}]$.

Thus, we can state that the parameters determining the size of the SMFs (i.e. the ones that define $\alpha_0 = 0$ plane) are trained to account for ambiguity, while the parameters defining the shape of the SMFs (i.e., those that define $\alpha_K = 1$ plane) are trained to achieve a high level of precision. To sum up, we propose the following two composite loss functions to learn GT2-FLS:

$$L_1 = L_R(\epsilon_{GT2-1}) + (\underline{\ell}_\tau^{\alpha_0} + \overline{\ell}_\tau^{\alpha_0}) \tag{5.25}$$

$$L_2 = L_R(\epsilon_{GT2-2}) + (\underline{\ell}_\tau^{\alpha_0} + \overline{\ell}_\tau^{\alpha_0}) \tag{5.26}$$

We believe that, through the proposed loss functions, an (partially) independent tuning of the $\boldsymbol{\theta_{GT2}}$ is possible for capturing uncertainty while ending up with high accuracy. It is worth mentioning that, although $y^{\alpha_0}(x_n)$ defined with $\alpha_0 = 0$ is not affecting the calculation of $y_{GT2}(x_n)$ or $y^{\alpha_K}(x_n)$, $J_x = [\underline{\mu}_{\tilde{A}_{p,m}}^{\alpha_0}, \overline{\mu}_{\tilde{A}_{p,m}}^{\alpha_0}]$ might affect the accuracy performance since the interval $[\underline{\mu}_{\tilde{A}_{p,m}^{\alpha_k}}, \overline{\mu}_{\tilde{A}_{p,m}^{\alpha_k}}]$ depends on J_x as given in equation 5.5. Yet, this effect can be seen as a regularization term within the proposed loss functions.

Input: N training samples $\{x_n, y_n\}_{n=1}^{N}$, $\varphi = [\underline{\tau}, \overline{\tau}]$

P, number of rules;
CSCM $\in \{KM, NT\}$; CSCM type;
T, the total number of iterations;
N_{bs}, mini-batch size;
lr, learning rate;
$L \in \{L_1, L_2\}$; Loss type

Output: Learnable parameter set $\boldsymbol{\theta_{GT2}}$.

for $m = 1$ to M **do**
  Find the min and max values of $\{x_{n,m}\}_{n=1}^{N}$;
  Calculate the standard deviation values of $\{x_{n,m}\}_{n=1}^{N}$;
    for $p = 1$ to P **do**
      Initialize $c_{p,m}$ uniformly between min and max values;
      Initialize $\sigma_{p,m}$ with the standard deviation of $\{x_n, y_n\}_{n=1}^{N}$;
      Initialize $h_{p,m}$ uniformly between [0, 1]
      Initialize $\boldsymbol{\delta^1} = \boldsymbol{\delta^2} = 0.5$;
    end for
end for
Initialize $\widehat{\boldsymbol{\theta}}_{\boldsymbol{C}}^{\boldsymbol{0}} = (a_{p,m}, a_{p,0})$ randomly around 0;
Construct $\widehat{\boldsymbol{\theta}}_{\boldsymbol{A}}^{\boldsymbol{0}} = (c_{p,m}, \sigma_{p,m}, h_{p,m}, \delta_m^1, \delta_m^2)$;
Define $\widehat{\boldsymbol{\theta}}^{\boldsymbol{0}} = \{\widehat{\boldsymbol{\theta}}_{\boldsymbol{A}}^{\boldsymbol{0}}, \widehat{\boldsymbol{\theta}}_{\boldsymbol{C}}^{\boldsymbol{0}}\}$;
for $t = 1$ to T **do**
  Select random N_{bs} samples
  for $n = 1$ to N_{bs} **do**
    for $k = 0$ to K **do**
      for $p = 1$ to P **do**
        Apply the tricks for $\widehat{\boldsymbol{\theta}}_{\boldsymbol{A}}$ via (31) and (32);
        Find firing strengths for each α-plane as
        Compute $\underline{f}_p^{\alpha_k}$via (11) and $\overline{f}_p^{\alpha_k}$ via (12);
      end for
      Compute $y_p^{\alpha_k}(x_n) = [\underline{y}^{\alpha_k}(x_n), \overline{y}^{\alpha_k}(x_n)]$;
      if CSCM = "KM"
        Calculate $\underline{y}^{\alpha_k}(x_n)$ via (13) and $\overline{y}^{\alpha_k}(x_n)$ via (14);
      else if CSCM == "NT"
        Calculate $\underline{y}^{\alpha_k}(x_n)$ via (16) and $\overline{y}^{\alpha_k}(x_n)$ via (17);
      endif
      Obtain $y^{\alpha_k}(x_n)$ via (9);
    end for
    Calculate $y_{GT2}(x_n)$ via (8);
  end for
  Compute L (L_1 via (29), L_2 via (30))
  Calculate the gradient as $\partial \boldsymbol{L}/\partial \widehat{\boldsymbol{\theta}}^{\boldsymbol{t}}$;
  Update $\widehat{\boldsymbol{\theta}}^{\boldsymbol{t}}$ via Adam optimizer;
end for
$\widehat{\boldsymbol{\theta}}^{*} = \arg\min_{\boldsymbol{\theta_{GT2}}}(L)$
Transform $\widehat{\boldsymbol{\theta}}^{*}$ into $\boldsymbol{\theta}^{*}$ by using (31) and (32);
Return $\boldsymbol{\theta_{GT2}} = \boldsymbol{\theta}^{*}$

Figure 5.2 : The learning algorithm for GT2-FLSs.

5.2.2 Composite learning framework with parametrization tricks

In this section of this thesis, we will present the parametrization tricks for the parametric GT2-FLSs in that way we can use DL based frameworks as the parameter optimizers for these parametric models. Inspiring from the connection between FLSs and NNs described in literature [33], we approached the parameter learning task of GT2-FLS using optimization algorithms available in DL frameworks, as illustrated in

Figure 4.3. Specifically, we utilized automatic differentiation and trained the GT2-FLS using a DL optimizer such as Adam.

In Figure 5.2, we present the proposed learning algorithm for GT2-FLSs which can be implemented in various DL frameworks. Here, we considered the fact the proposed loss function has to be trained by bearing in mind the constraints $\boldsymbol{\theta_{GT2}} \in \boldsymbol{\mathcal{C}}$. Thus, since DL optimizers require unbounded search spaces of the learnable parameters, we proposed parametrization tricks for $\boldsymbol{h}, \boldsymbol{\delta^1}, \boldsymbol{\delta^2} \in \boldsymbol{\theta_{GT2-A}}$ to guarantee that the learned $\tilde{A}_{p,m}^{\alpha_k}$ satisfy the conditions of FS, $\overline{\mu}_{\tilde{A}_{p,m}^{\alpha_k}}(x_m) \geq \underline{\mu}_{\tilde{A}_{p,1}^{\alpha_k}}(x_m)\ \forall p, m$.

Note that $\boldsymbol{c}, \boldsymbol{\sigma}, \boldsymbol{h}$, and $\boldsymbol{a}$ are defined within unbounded search spaces and thus parametrization tricks are not required.

- For $\boldsymbol{h}$, the constraint $0 \leq\ h_{p,m} \leq 1$ has to be satisfied which can be eliminated via the following trick:

$$h_{p,m} = sig(h'_{p,m}) \tag{5.27}$$

  where $sig(\cdot)$ is the sigmoid function that provides the generation of unbounded optimization variables.

- For $\boldsymbol{\delta^1}$ and $\boldsymbol{\delta^2}$, to guarantee a convex SMF, the constraint is $0 \leq \delta_m^2 \leq\ \delta_m^1 \leq 1$. To always satisfy this condition, we proposed the following parameterization tricks:

$$\delta_m^1 = sig\left(\delta_m^{1'}\right) \quad \delta_m^2 = \delta_m^1\, sig(\delta_m^{1'}) \tag{5.28}$$

Hence, we defined an unbounded set of learnable parameters $\left\{h'_{p,m}, \delta_m^{1'}, \delta_m^{2'}\right\}$ and thus we can now perform the training within DL frameworks such as Matlab and PyTorch.

6. EXPERIMENTAL STUDIES

In this section, we will provide a comprehensive experimental analysis for the proposed deep learning based T1/T2-FLSs. Firstly, we start presenting the experimental studies for the "Learning with Type-2 Fuzzy Activation Functions to Improve the Performance of Deep Neural Networks" which is explained in detail in section 3.4. After that, we will provide a detailed analysis for the composite deep learning approach for IT2-FLSs which is introduced in section 4. Moreover, we will present the learning performance analysis of the composite loss for the GT2-FLSs in the section 6.3.

6.1 Comparative Experimental Performance of the SIT2-FRU

In this section, we focus on evaluating the learning performance of DNN structures integrated with IT2-FRUs in supervised learning tasks. Specifically, we compare the learning performance of the newly proposed SIT2-FRU with existing activation functions such as ReLU, PReLU, and ELU. To assess and compare their performances, we conducted experiments using several benchmark datasets, namely MNIST handwritten digits [6], Quickdraw Pictionary [107], and CIFAR-10 [106].

In the case of classification problems, we employed the cross entropy loss function along with L2 regularization. This loss function is widely used and is defined as follows:

$$L(y_q, \hat{y}_q) = -\frac{1}{Q}\sum_{q=1}^{Q}\left[y_q \log \hat{y}_q + (1 - y_q)\log(1 - \hat{y}_q)\right] + \lambda \frac{w^T w}{2} \tag{6.1}$$

In regression problems, we utilized the Root Mean Square Error (RMSE) as the evaluation metric, along with L2 regularization. The RMSE is a commonly used metric to measure the accuracy of regression models, and it is defined as the square root of the mean of the squared differences between the predicted values and the actual values. The formulation of the RMSE with L2 regularization is given by equation 6.2:

$$L(y_j, \hat{y}_j) = \sqrt{\sum_{j=1}^{J} \frac{(\hat{y}_j - y_j)^2}{J}} + \lambda \frac{w^T w}{2} \quad (6.2)$$

In the given equation, J represents the total number of samples in the dataset. The regularization factor, denoted as λ, is a hyperparameter that controls the amount of regularization applied to the model. In this study, λ is set and fixed to a value of 0.004, indicating the strength of regularization in the L2 regularization term.

All experiments were conducted on a PC equipped with an Intel Core i7 3.3GHz CPU, 32GB RAM, and an NVIDIA GTX 1080 TI GPU. The MATLAB and CUDA environments were utilized for running the experiments. To ensure robustness and reliability of the results, each experiment was repeated 5 times with different weight initializations. The performance measures, including median, mean, and standard deviation (std), were calculated based on the obtained results. This approach provides a comprehensive evaluation of the performance by considering both the central tendency (median and mean) and the variability (standard deviation) of the performance measures across multiple runs.

6.1.1 Performance evaluations on the MNIST dataset

In order to assess the classification and regression capabilities of the IT2-FAL, we utilized the widely used MNIST dataset. This dataset consists of grayscale images with dimensions of 28×28 pixels, representing handwritten digits ranging from 0 to 9. The dataset comprises a total of 10,000 images. To conduct our experiments, we partitioned the dataset into two subsets: 7,000 images for training and 3,000 images for testing.

6.1.1.1 MNIST handwritten digit classification performance

In this subsection, we aim to demonstrate the superior learning behavior of the SIT2-FRU activation function compared to other activation functions. To highlight the advantages of incorporating the IT2-FAL into DNNs, we constructed a simple DNN architecture shown in Figure 6.1. The DNN consists of an image input layer with 'zero-center' normalization, followed by a convolution layer with 8 filters. Each kernel size in the convolution layer is 3x3, with padding of 3 pixels on each side. This layer is then followed by an activation layer and a fully connected (FC) layer with a size of 10.

The output layer utilizes the softmax activation function. To train the DNN, we set the hyperparameters as follows: learning rate (ξ) = 0.001, momentum (ρ) = 0.9, mini-batch size (MBS) = 128, and number of epochs = 15.

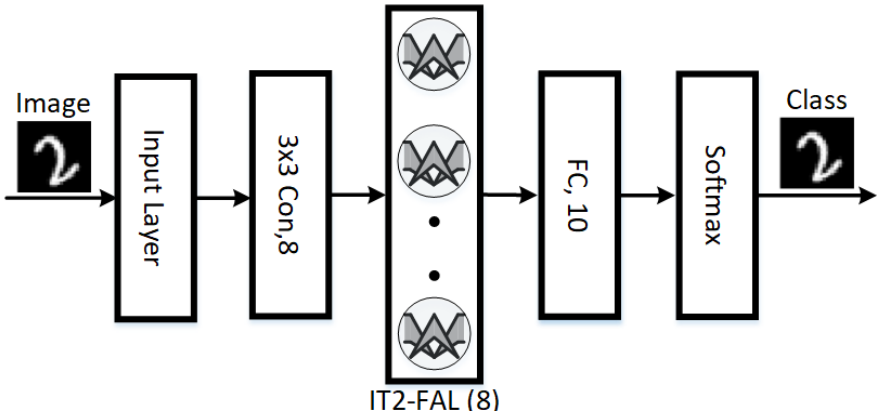


Figure 6.1 : The developed DNN architecture for MNIST digit classification

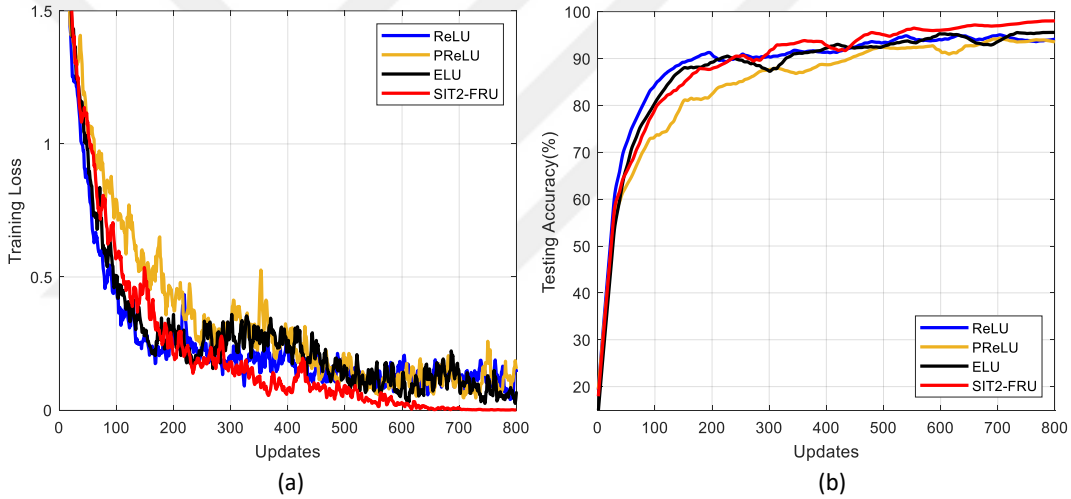


Figure 6.2 : MNIST Classification results with a simple DNN (a) Medians of training loss values over five trials (b) Medians of testing accuracy values over five trials.

Figure 6.2 shows the progression of the median training and testing accuracy of the DNN across five experiments conducted on the MNIST dataset. The results clearly demonstrate that the SIT2-FRU activation function outperforms ReLU, PReLU, and ELU in terms of both convergence speed and testing accuracy. Furthermore, Table 6.1 provides the final testing accuracy values of the DNNs. It is observed that SIT2-FRUs achieve a 4.4% increase in median accuracy compared to ReLUs and exhibit the highest mean testing accuracy rate of 98.23% (with a standard deviation of 0.005) among all the tested activation functions.

Table 6.1 : Classification accuracy result of the testing data over 5 experiments for MNIST.

	ReLU	PReLU	ELU	SIT2-FRU
Median (%)	93.82	93.30	95.63	**98.22**
Mean (%)	93.72	92.23	95.01	**98.23**
std	0.011	0.036	0.029	**0.005**

Figure 6.3 provides a comparison of the initial and learned mappings of the SIT2-FRUs for a particular classification experiment. It is evident from the visualization that the SIT2-FRUs exhibit diverse behaviors during initialization, ranging from mappings resembling ReLU or LReLU to those resembling S-FM and A-FM. As the training progresses, the negative quadrant (N) slope values are adjusted to map negative inputs closer to zero or to positive values. Meanwhile, the slope parameter P remains largely unchanged, and the FOU parameter (α) is learned to smooth the input-output mapping (S-FM). This analysis suggests that the SIT2-FRU, with its ability to generate various membership functions including those of ReLU and LReLU, enables the DNN structure to achieve a more stable and superior learning performance compared to ReLU, PReLU, and ELU counterparts.

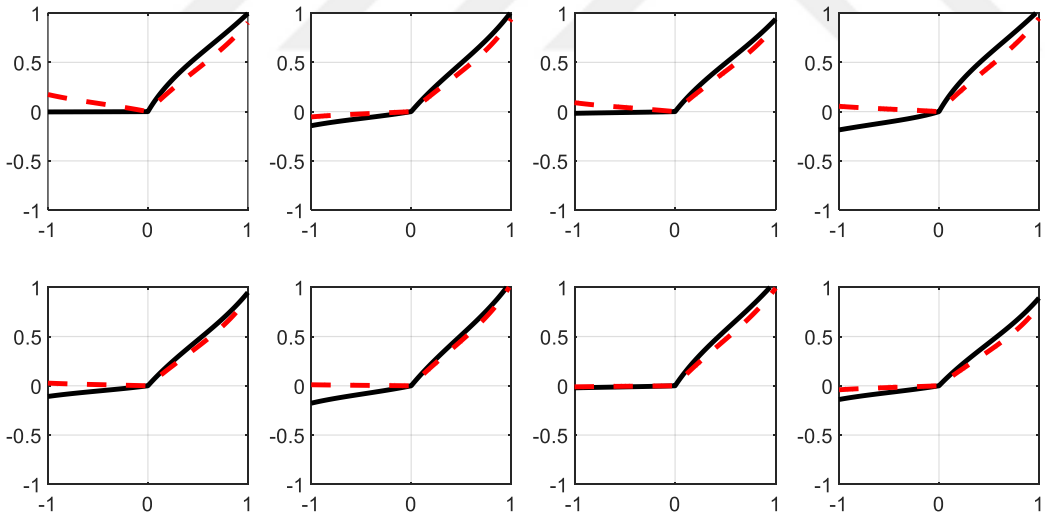


Figure 6.3 : Initial solid black ones and learned dashed red ones represent the mappings of the SIT2-FRUs for a particular classification example using MNIST dataset.

6.1.1.2 MNIST handwritten digit regression performance

In this subsection, we explore the performance of DNNs in regression tasks, specifically in predicting the rotation angles of handwritten digits. Our aim is to assess

the effectiveness of SIT2-FRU compared to other activation functions in predicting the rotation angles of handwritten digits.

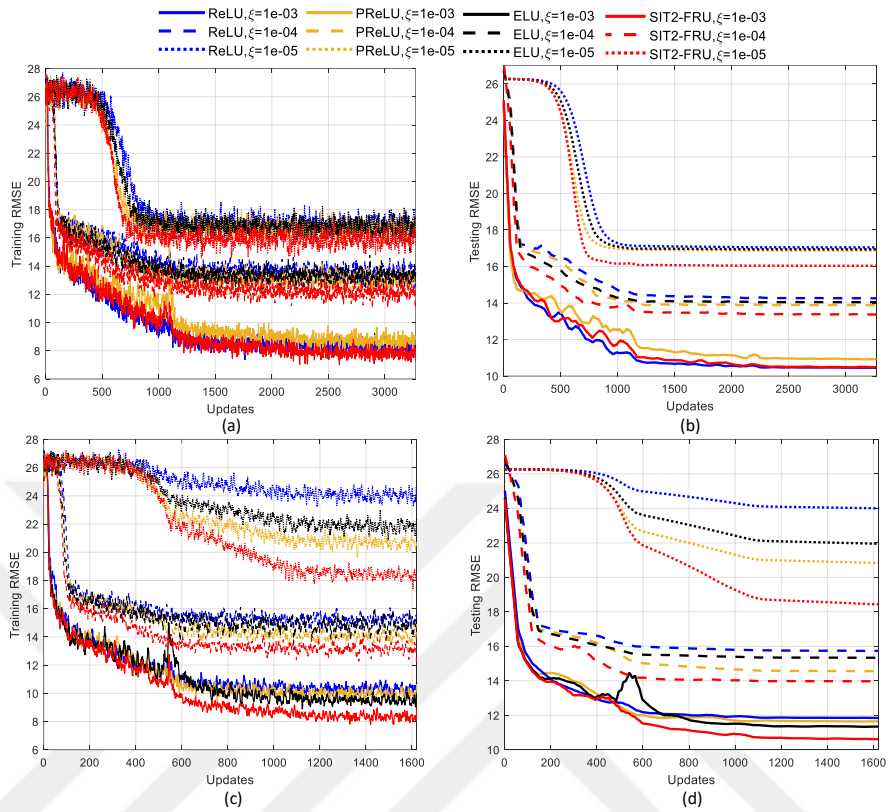


Figure 6.4 : MNIST regression results of DNN, (a) Training RMSE values for MBS = 64 (b) Testing RMSE values for MBS = 64 (c) Training RMSE values for MBS = 128 (d) Testing RMSE values for MBS = 128.

Our objective is to demonstrate the superior and robust learning behavior of SIT2-FRU across various hyperparameter settings, including the presence or absence of a Batch Normalization (BN) layer. To conduct a fair comparison, we carefully construct a DNN architecture that maintains a reasonable level of complexity while being suitable for the task at hand. This ensures that any observed differences in performance can be attributed to the choice of activation function rather than the network's overall architecture. By examining the performance of SIT2-FRU in relation to ReLU, PReLU, and ELU, we can gain insights into how different activation functions impact the learning process. This analysis allows us to draw conclusions about the advantages of deploying SIT2-FRU, particularly its ability to achieve superior and consistent learning outcomes in regression tasks, even under varying hyperparameter settings.

The DNN architecture used in our experiments consists of an image input layer with 'zero-center' normalization.

Table 6.2 : Regression test accuracy result of DNN over 5 experiments for MNIST.

Activation	Criteria	**MSB: 64**			**MSB: 128**		
		$\xi: 10^{-3}$	$\xi: 10^{-4}$	$\xi: 10^{-5}$	$\xi: 10^{-3}$	$\xi: 10^{-4}$	$\xi: 10^{-5}$
ReLU	Median (%)	68.92	50.85	41.71	60.85	45.72	24.57
	Mean (%)	68.83	50.32	41.20	60.63	45.51	24.23
	std	0.009	0.012	0.011	**0.004**	0.006	0.012
PReLU	Median (%)	66.46	52.14	42.98	61.35	49.35	30.21
	Mean (%)	65.91	51.87	42.24	60.78	49.24	29.93
	std	0.027	0.007	0.013	0.016	0.011	0.031
ELU	Median (%)	-	51.51	42.35	64.47	46.38	27.72
	Mean (%)	-	51.02	41.93	64.16	46.04	27.58
	std	-	0.009	0.008	0.011	0.009	**0.007**
SIT2-FRU	Median (%)	**69.81**	**54.82**	**45.93**	**68.64**	**51.62**	**35.78**
	Mean (%)	**69.74**	**54.75**	**45.81**	**68.51**	**51.46**	**35.61**
	std	**0.002**	**0.001**	**0.005**	0.005	**0.004**	**0.007**

**Empty cells represents non converging training performances*

This is followed by a convolutional layer with 32 filters, where each kernel has a size of 3×3 and is padded by 3 pixels on each side. Two variations of the network are constructed: one includes a Batch Normalization (BN) layer (referred to as BN-DNN), while the other does not include a BN layer (referred to as DNN). Following the convolutional layer, an activation layer is added, with options including ReLU, PReLU, ELU, or SIT2-FRU. Finally, a fully connected layer with a size of 1 is appended to the network. During the training process, we employ the stochastic gradient descent method with mini-batch sizes of 64 and 128. Three fixed learning rates are utilized: $\xi = 10\text{^}(-3)$, $\xi = 10\text{^}(-4)$, and $\xi = 10\text{^}(-5)$. These different configurations enable us to investigate the performance of the DNNs under varying hyperparameter settings and activation functions. By comparing the results of BN-DNN and DNN with different activation functions, we aim to assess the impact of the presence of a BN layer and the choice of activation function on the learning behavior and predictive accuracy of the DNNs for regression tasks.

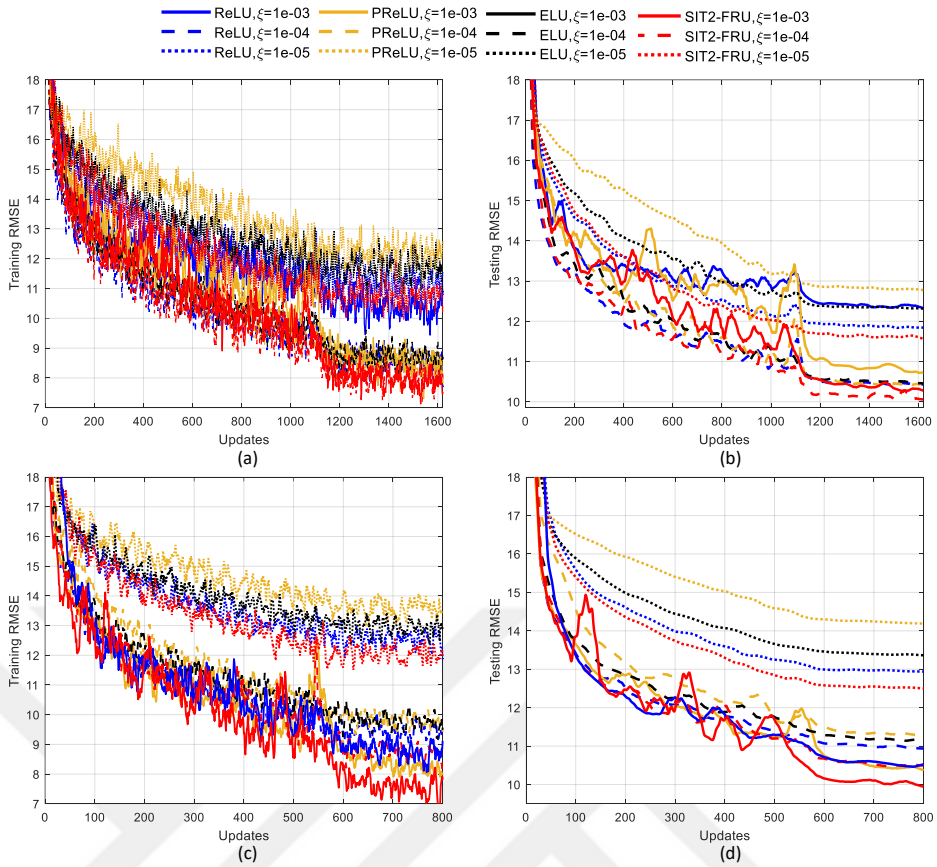


Figure 6.5 : MNIST regression results of BN-DNN, (a) Training RMSE values for MBS = 64 (b) Testing RMSE values for MBS = 64 (c) Training RMSE values for MBS = 128 (d) Testing RMSE values for MBS = 128.

Figure 6.4 and Figure 6.5 display the median training and testing Root Mean Square Error (RMSE) values, respectively, for the DNNs and BN-DNNs. The tables, namely Table 6.2 and Table 6.3, present the corresponding testing performance measures for the different experiments conducted. Analyzing the testing performance, we observe that in most experiments, the SIT2-FRU activation function consistently achieves the highest median accuracy rate compared to ReLU, PReLU, and ELU. When compared to its ELU counterpart, the deployment of SIT2-FRU in the DNN leads to a significant increase of 4.17% in the median accuracy rate. Specifically, in the case of MBS = 128 and $\xi = 10\text{^}(-3)$, the SIT2-FRU achieves the maximum median accuracy rate of 68.64%. Furthermore, the BN-DNN with SIT2-FRU outperforms its ReLU counterpart by 1.95% in terms of accuracy rate for MBS = 64 and $\xi = 10\text{^}(-4)$ (as shown in Table 6.3). Notably, the incorporation of IT2-FRU into the DNN consistently yields the lowest standard deviation value across different hyperparameter settings, demonstrating its robust performance.

Table 6.3 : Regression accuracy result of the BN-DNN over 5 experiments for MNIST.

Activation	Criteria	**MBS: 64**			**MBS: 128**		
		$\xi: 10^{-3}$	$\xi: 10^{-4}$	$\xi: 10^{-5}$	$\xi: 10^{-3}$	$\xi: 10^{-4}$	$\xi: 10^{-5}$
ReLU	Median (%)	58.80	68.87	61.78	68.23	65.23	56.65
	Mean (%)	58.49	68.77	61.66	69.56	65.00	56.44
	std	0.005	0.008	**0.005**	0.025	0.009	0.004
PReLU	Median (%)	66.48	67.90	57.35	69.27	62.53	50.82
	Mean (%)	66.37	67.67	57.23	68.98	62.42	50.57
	std	0.004	0.005	0.012	0.008	0.018	0.005
ELU	Median (%)	-	68.22	59.47	-	64.22	54.38
	Mean (%)	-	68.17	59.41	-	63.98	54.33
	std	-	**0.003**	0.008	-	0.007	**0.003**
SIT2-FRU	Median (%)	**70.08**	**70.82**	**62.37**	**71.57**	**67.62**	**58.27**
	Mean (%)	**70.02**	**70.70**	**62.30**	**71.26**	**67.48**	**58.12**
	std	**0.002**	0.007	0.007	**0.002**	**0.004**	0.008

**Empty cells represents non converging training performances*

The robustness of the SIT2-FRU activation function to hyperparameter selection is evident from the results presented in Figure 6.4 and Figure 6.5. Across various hyperparameter settings, the SIT2-FRUs consistently achieve the highest median test accuracy rates, as indicated in Table 6.2. Notably, even in cases where the BN-DNN with ELUs fails to learn the dataset (MBS = 64 and 128), the BN-DNN with SIT2-FRUs maintains its high test accuracy performance, as shown in Table 6.3. While the inclusion of the BN layer generally improves the learning performance of the DNNs, the DNN with SIT2-FRU surpasses the BN-DNN with other activation functions in terms of performance. This is evident for the specific hyperparameter setting of MBS = 64 and $\xi = 10\text{^}(-3)$. These findings clearly demonstrate that incorporating the IT2-FAL into the DNN structure leads to superior and robust learning behavior. The SIT2-FRU activation function not only outperforms other activation functions across various hyperparameter settings but also maintains high accuracy even when combined with the BN layer, highlighting its effectiveness in enhancing the learning performance of DNNs.

6.1.2 Performance evaluations on the quickdraw pictionary dataset

In this subsection, we analyze the learning behavior and performance of the SIT2-FRU activation function within a more complex DNN structure using the quickdraw pictionary dataset. This dataset consists of grayscale hand-drawn images with dimensions of 256×256 pixels. We specifically focus on the 10 classes of images, namely "ant, cake, wristwatch, umbrella, wine glass, zebra, teapot, pliers, saw, castle," as shown in Figure 10. The quickdraw pictionary dataset comprises 50,000 training images and 5,000 testing images.

To assess the learning behavior and performance of different activation units, we utilized a Stacked DNN (S-DNN) structure. The S-DNN consists of several layers, starting with an image input layer that undergoes 'zero-center' normalization. The structure is then composed of three consecutive convolutional layers with configurations of [1 × 16 × 3], [1 × 32 × 3], and [1 × 64 × 3], denoting the number of filters and kernel sizes. Each convolutional layer is padded with one pixel on each side and is followed by an activation layer and a max-pooling layer with a 2×2 kernel size and stride of 2. Finally, a fully connected (FC) layer with a size of 10 and a softmax classification layer are included in the S-DNN. During training, we employed stochastic gradient descent for five epochs with a mini-batch size of 100, a momentum factor (ρ) of 0.9, and a fixed learning rate (ξ) of 10^{-3}.

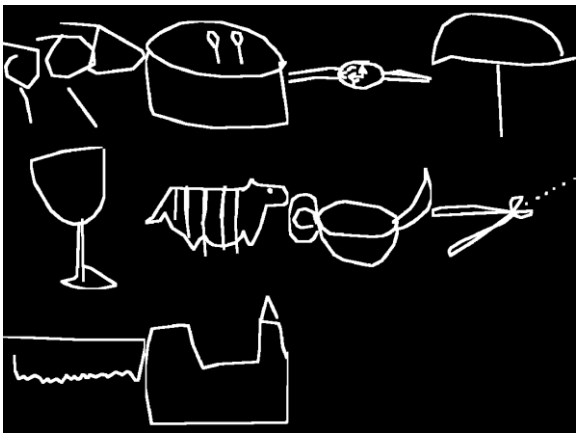


Figure 6.6 : Selected 10 samples images of pictionary data.

In Figure 6.7, the progression of training and testing losses for the Stacked DNNs (S-DNNs) with different activation functions is presented. It is evident that the S-DNNs using ReLUs, PReLUs, and ELUs experience a noticeable divergence between their training and testing losses after approximately 1k updates, suggesting the presence of

overfitting. However, the S-DNN with IT2-FALs demonstrates a more stable learning behavior, as indicated by the consistent and relatively low training and testing losses throughout the training process. This highlights the robustness of the IT2-FAL in preventing overfitting and maintaining a high learning performance.

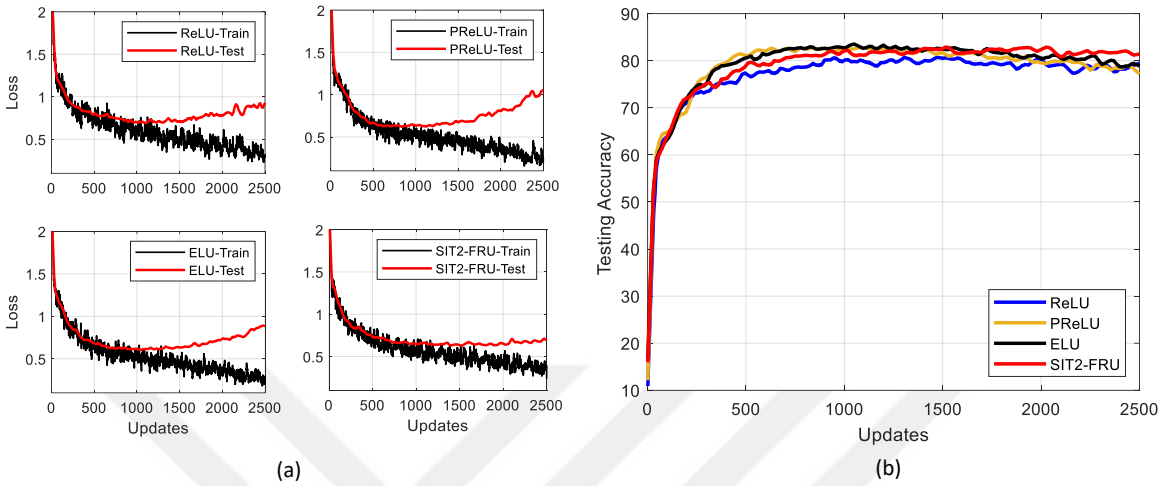


Figure 6.7 : Quickdraw pictionary classification results (a) Medians of training and testing loss values over five trials (b) Medians of testing accuracy values over five trials.

6.1.3 Performance evaluations on the CIFAR-10 dataset

In this section, we conducted experiments using two different configurations of stacked DNNs to evaluate the learning behavior and generalization capabilities of the proposed SIT2-FRU. The experiments were conducted on the CIFAR-10 dataset, which comprises 50k training images and 10k testing images belonging to 10 different classes. The objective was to assess how well the SIT2-FRU performs in classifying the images accurately across different classes in the CIFAR-10 dataset.

6.1.3.1 Stacked DNNs with single convolutional layer

In the S-DNN structures we designed, the first layer is an image input layer that applies 'zero-center' normalization to the input data. This is followed by a convolutional layer with 32 filters, where each filter has a kernel size of 5x5 and is padded with 2 pixels on each side. One of the structures includes a BN layer, referred to as the BN-DNN, while the other structure does not include a BN layer and is simply called the DNN. After the convolutional or BN layer, an activation unit is applied, followed by a fully connected layer with a size of 10. The final layer is a softmax classification layer for classifying the input images into one of the 10 classes.

Table 6.4 : Regression accuracy result of the over 5 experiments for CIFAR-10.

Activation	Criteria	DNN-Type		DNN-Type		DNN-Type
		BN-DNN	DNN	BN-S-DNN-1	S-DNN-1	S-DNN-2
ReLU	Median (%)	43.84	48.86	76.07	56.24	82.60
	Mean (%)	43.51	48.90	76.27	56.13	82.48
	std	0.033	0.008	0.004	0.009	0.006
PReLU	Median (%)	55.94	47.65	76.54	-	82.50
	Mean (%)	55.78	47.07	76.52	-	82.40
	std	**0.005**	0.022	0.003	-	**0.005**
ELU	Median (%)	57.29	48.70	76.63	59.77	82.64
	Mean (%)	57.60	48.45	**76.70**	58.60	82.49
	std	0.016	0.009	0.001	0.023	**0.005**
SIT2-FRU	Median (%)	**61.23**	**50.90**	**76.73**	**74.35**	**83.84**
	Mean (%)	**61.02**	**50.76**	76.54	**74.40**	**83.56**
	std	0.009	**0.005**	0.003	**0.007**	0.006

**Empty cells represents non converging training performances*

The second DNN structure consists of three stacked convolutional layers with different filter sizes: [1 × 32 × 5], [1 × 32 × 5], [1 × 64 × 5]. Each convolutional layer is padded with 2 pixels on each side to maintain the spatial dimensions of the input. In one structure, a BN layer is incorporated after each convolutional layer, resulting in the BN-S-DNN-1 model. In contrast, the S-DNN-1 model does not include any BN layers. Following the convolutional and BN layers, an activation layer is added, followed by max-pooling layers with a 3×3 kernel size and a stride of 2. Two fully connected layers with sizes of 64 and 10, respectively, are then added, with an activation unit between them. Finally, a softmax classification layer is appended to the network for the final output.

During the training phase of the DNN, BN-DNN, S-DNN-1, and BN-S-DNN-1 architectures, the learning rate is initially set to $\xi = 10^{-2}$ and it is reduced by a factor of 10 every 2 epochs. Stochastic gradient descent is employed as the optimization algorithm with a mini-batch size of 64 samples and a momentum value of ρ=0.9 for a total of 5 epochs. The resulting testing accuracies of these models are summarized in Table 6.4. Notably, SIT2-FRU achieved a higher median accuracy rate compared to ReLU, PReLU, and ELU activations. The DNN with SIT2-FRUs exhibited a median accuracy improvement of 2.04% over its ReLU counterpart, reaching a maximum

median accuracy of 50.90%. Moreover, the utilization of SIT2-FRU within the BN-DNN structure led to a substantial mean accuracy enhancement of 3.42% compared to its ELU counterpart, as indicated in Table 6.4. Additionally, the standard deviation values demonstrate that the inclusion of SIT2-FRU in both the BN-S-DNN-1 and S-DNN-1 structures resulted in consistently high learning performance, irrespective of the presence of a BN layer.

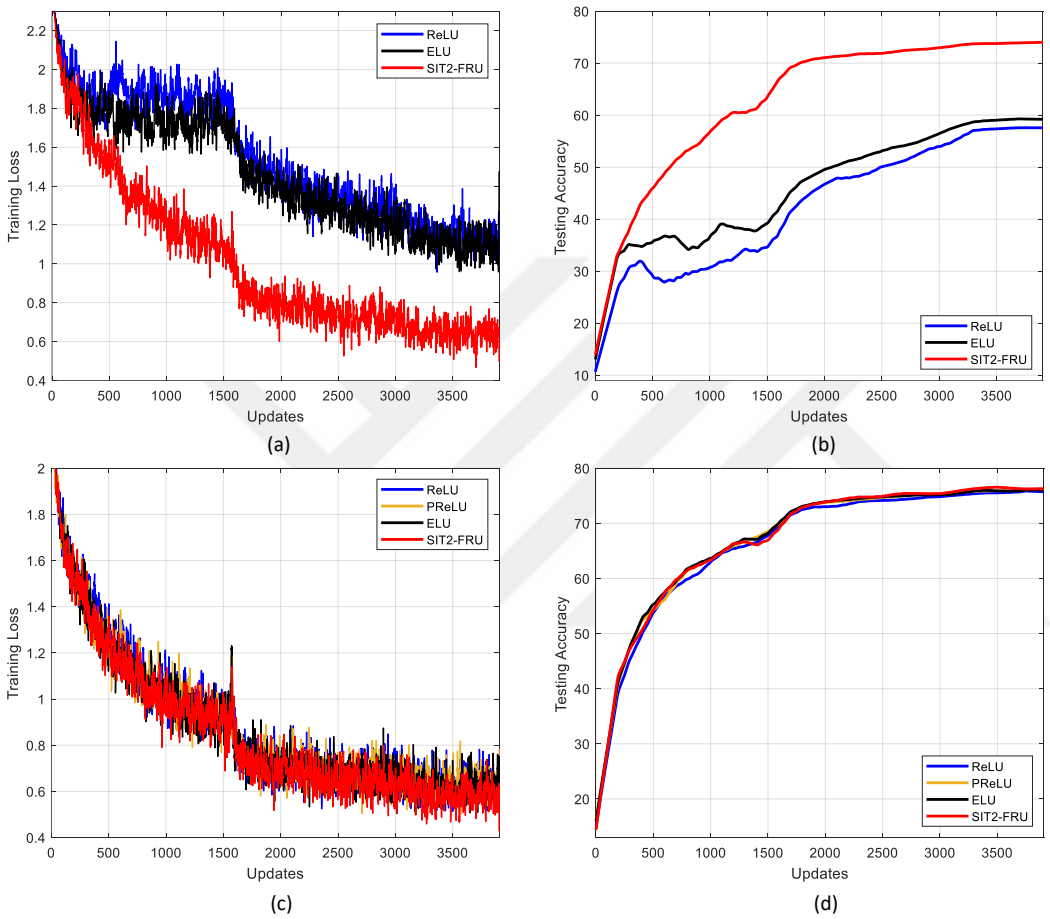


Figure 6.8 : CIFAR-10 dataset, (a) Training Loss values of S-DNN-1 (b) Testing Accuracy values of S-DNN-1 (c) Training Loss values of BN-S-DNN-1 (d) Testing.

Figure 6.8 provides a visual representation of the median training loss and testing accuracy results for S-DNN-1 and BN-S-DNN-1 architectures across five experiments. Notably, the results for PReLU are not shown in the figures due to its inability to learn the dataset under the specified hyperparameter setting. The outcomes clearly demonstrate that the proposed SIT2-FRU consistently outperforms ReLU, PReLU, and ELU in terms of lower training loss and higher testing accuracy, as evident in Figure 6.8. The remarkable performance of SIT2-FRU is particularly pronounced within the S-DNN-1 structure, as illustrated in Figure 6.8. Referring to

Table 6.4, SIT2-FRU exhibits a significant improvement in median accuracy rate, surpassing ReLU by 18.11% and achieving a maximum median accuracy rate of 74.35% in the S-DNN-1 configuration. It is important to highlight that the inclusion of a BN layer substantially enhances the training performance of ReLU, PReLU, and ELU within the BN-S-DNN-1 and S-DNN-1 structures. Additionally, while the BN layer slightly boosts the training performance of SIT2-FRU, as indicated in Figure 6.8. These findings underscore the crucial role of the BN layer in the performance of ReLU, PReLU, and ELU, while also highlighting the ability of SIT2-FRU to deliver satisfactory results with or without a BN layer, as evidenced by the findings in Table 6.4.

6.1.3.2 Stacked DNNs with multi convolutional layers

To further validate the results obtained from the S-DNN-1 structures, we designed a more intricate S-DNN configuration, named S-DNN-2, with multiple convolutional layers. The S-DNN-2, as depicted in Figure 6.9, is composed of 8 convolutional layers organized in stacks of [2 × 32 × 3], [2 × 64 × 3], and [4 × 128 × 3]. Each convolutional layer is padded by 2 pixels, and the max-pooling layers have a stride of 2. For training the S-DNN-2, we employed the stochastic gradient descent algorithm for 20 epochs, utilizing a mini-batch size of 100 samples and a momentum value of ρ=0.9. The initial learning rate was set to $\xi = 10^{-2}$ and it was decreased by a factor of 10 after 10 epochs. The objective of employing the S-DNN-2 is to investigate whether the favorable outcomes observed in previous section can be generalized to more complex DNN structures.

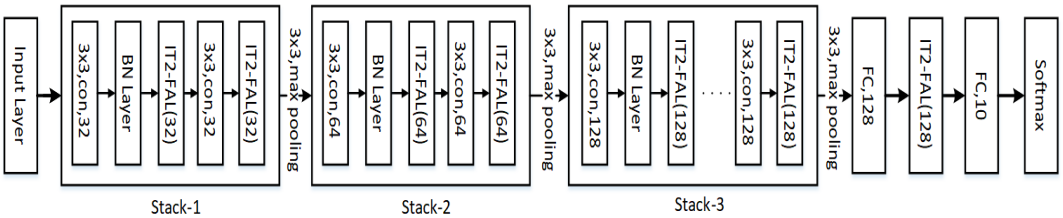


Figure 6.9 : The employed S-DNN-2 structure for CIFAR-10 classification.

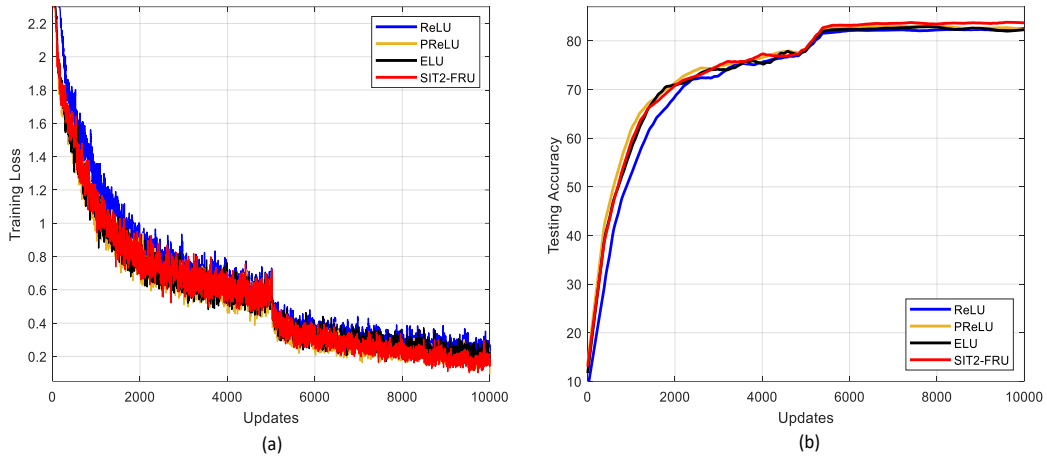


Figure 6.10 : S-DNN-2: (a) Training Loss values (b) Testing Accuracy values.

Figure 6.10 illustrates the medians of training loss and testing accuracy obtained from five experiments. It is evident that the SIT2-FRU exhibits superior convergence behavior compared to the other activation units. Table 6.4 confirms that PReLU, ELU, and ReLU yield similar performance measures, whereas the S-DNN-2 with SIT2-FRU achieves a median test accuracy rate of 83.84%, surpassing its ELU counterpart by 1.2%. These results indicate that DNNs incorporating IT2-FALs possess robust generalization capabilities, enabling them to achieve high learning performance even in complex S-DNN structures with multiple convolutional layers.

6.2 Comparative Results of the Composite Learning Approach for IT2-FLSs

In this section, we will present the results of the composite learning approach for IT2-FLSs which we introduced in section 4. Our aim in the experiments is to focus on analyzing the performance of the proposed composite learning approach for IT2-FLSs. Specifically, we present comparative results based on the KM type CSCM, as we have observed that employing KM tends to yield better learning performance.

We conducted experiments on eleven preprocessed datasets, as indicated in Table 6.5. To ensure fairness in the comparison, we initialized the parameters $\boldsymbol{\theta}$ randomly following the algorithm illustrated in Figure 4.4. All parametric IT2-FLSs used the same initial parameter set $\widehat{\boldsymbol{\theta}}^{\mathbf{0}}$. We set the expected coverage at 80% ($\varphi = [0.1,0.9]$) and performed five iterations (P). The training process spanned 100 epochs. The optimal values for the minibatch size (N_{bs}) and learning rate (lr) were determined through a grid search using the H-C model. N_{bs} was explored within the range [8,256],

divided into grids of size 4, while lr ranged from [0.001,0.2] with grids of size 5. The resulting N_{bs} and lr values were employed for training all IT2-FLSs. The experiments were conducted using Matlab®, and the performance evaluation involved measuring the Root Mean Squared Error (RMSE) for accuracy and the PI Coverage Probability (PICP) for uncertainty coverage.

Remark: It is important to note that in IT2-FLSs, the product operator is commonly used as the t-norm (∩) for most learning problems. However, we observed the occurrence of the vanishing gradient phenomena in datasets with ten or more input feature dimensions and more than 500 data sample points for each dimension. To address this issue, IT2-FLSs in such cases employ the minimum operator as the t-norm (∩).

Table 6.5 : Summary of the Datasets to train IT2-FLSs with the proposed composite approach.

Dataset	N_{Train}	N_{Test}	Normalization	N_{bs}	lr
Mcycle[1] (133×1)	89	44	Standard	32	0.01
Diabetes[2] (43×2)	35	8	Standard	16	0.01
Servo[3] (167×4)	134	33	Min-Max	16	0.002
MCPU[2] (209×6)	168	41	Max-Min	16	0.005
Bodyfat[4](252×14)	202	50	Max-Min	64	0.001
NO2[4] (500×7)	350	150	Standard	64	0.02
PM10[4](500×7)	350	150	Standard	64	0.02
Housing[3] (506×13)	405	101	Max-Min	64	0.01
Concrete[3](1030×8)	824	206	Max-Min	64	0.01
Wine[3](4898×11)	3429	1469	Standard	128	0.005
PowerPlant[3](9568×4)	6698	2870	Standard	128	0.02

[1]https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/mcycle.html.
[2]https://www.dcc.fc.up.pt/∼ltorgo /Regression/DataSets.html.
[3]https://archive.ics.uci.edu/ml/datasets.html.
[4]https://lib.stat.cmu.edu/datasets/.

6.2.1 Composite learning performance and sensitivity analysis

This subsection focuses on evaluating the performance of the proposed composite learning method and its sensitivity to hyperparameters for the parametric IT2-FLSs. To facilitate the analysis, we selected the univariate Mcycle dataset as the experimental dataset.

6.2.1.1 Composite learning performance analysis

Figure 6.11 displays the testing loss performances of the IT2-FLSs. It can be observed that IT2-FLSs employing H type antecedent membership functions tend to converge faster compared to other variants. Conversely, IT2-FLSs with IV/IVL rule consequents exhibit similar testing loss performances, which can be attributed to the higher number of learnable parameters in these models. However, it is important to note that all models achieved rapid convergence with low testing losses, indicating successful learning.

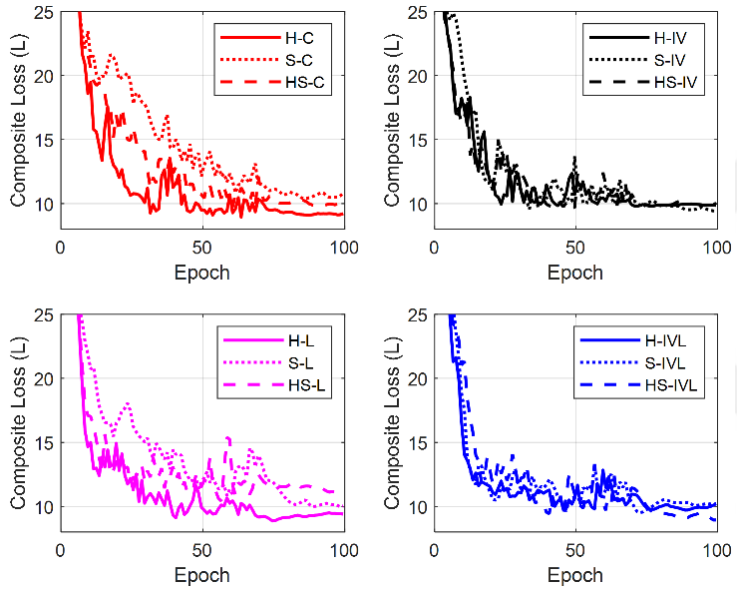


Figure 6.11 : Testing composite losses of the IT2-FLSs on the Mcycle dataset.

Table 6.6 : Testing performance results for Mcycle Dataset.

		C	L	IV	IVL
RMSE*	**H**	43.1	44.3	44.8	45.4
	S	44.3	45.3	44.0	45.6
	HS	44.5	45.6	45.1	43.2
PICP	**H**	75.0	93.1	72.7	75.0
	S	79.5	79.5	81.8	81.8
	HS	63.6	70.4	86.3	75.0

* RMSE values are scaled by 100.

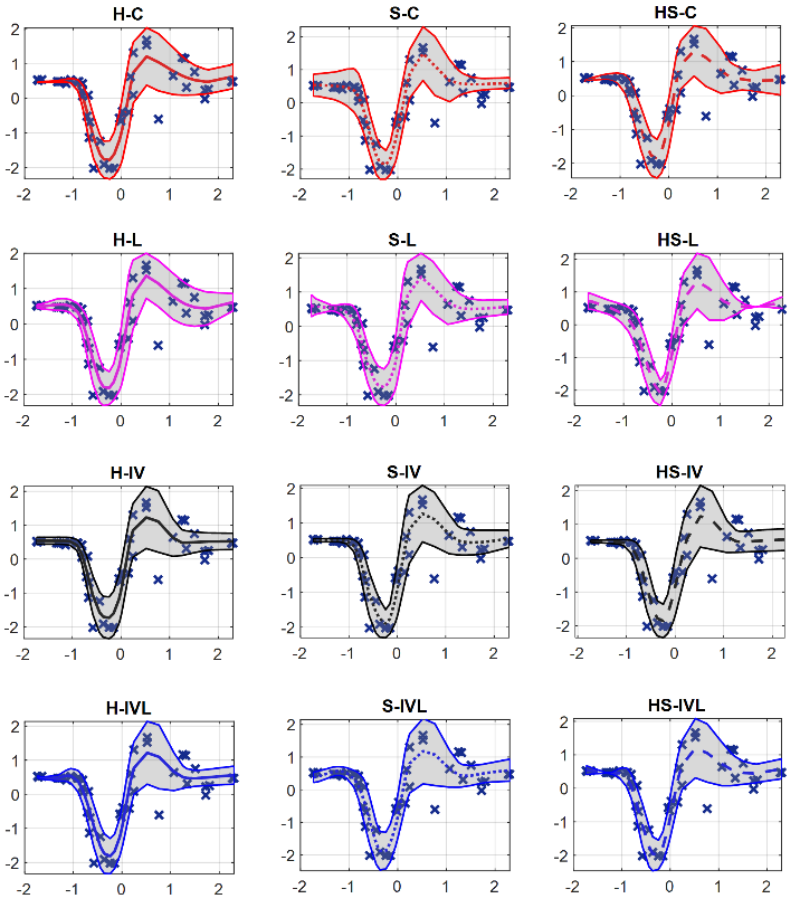


Figure 6.12 : Prediction results for the testing dataset.

The predictions for $y(\boldsymbol{x})$ and $\tilde{Y}(x) = [\underline{y}(\boldsymbol{x}), \overline{y}(\boldsymbol{x})]$ are given in Figure 6.12, while the corresponding evaluation measures are presented in Table 6.6. It is notable that the IT2-FLSs employing S-type antecedent IT2-FSs demonstrate superior performance in terms of expected coverage (80% ($\varphi = [0.1,0.9]$)) across all models. Moreover, IT2-FLSs with H type IT2-FSs exhibit better performance for C/L type consequents. Interestingly, the IT2-FLSs with IV consequents exhibit enhanced coverage of expected uncertainty compared to their crisp counterparts. On a different note, the IT2-FLS with C rule consequents achieves the lowest RMSE values, indicating more accurate predictions. These findings highlight the inherent trade-off within the composite loss function, as it optimizes both the expected coverage and prediction mean in a multi-objective manner.

In conclusion, the results demonstrate that the proposed composite loss function is effective in learning IT2-FLSs. The low RMSE values indicate high accuracy in prediction, while the resulting PICP values align well with the expected coverage of 80%, indicating successful representation of uncertainty. Therefore, the composite loss

proves to be an efficient approach for learning IT2-FLSs, achieving both accurate predictions and reliable uncertainty representation.

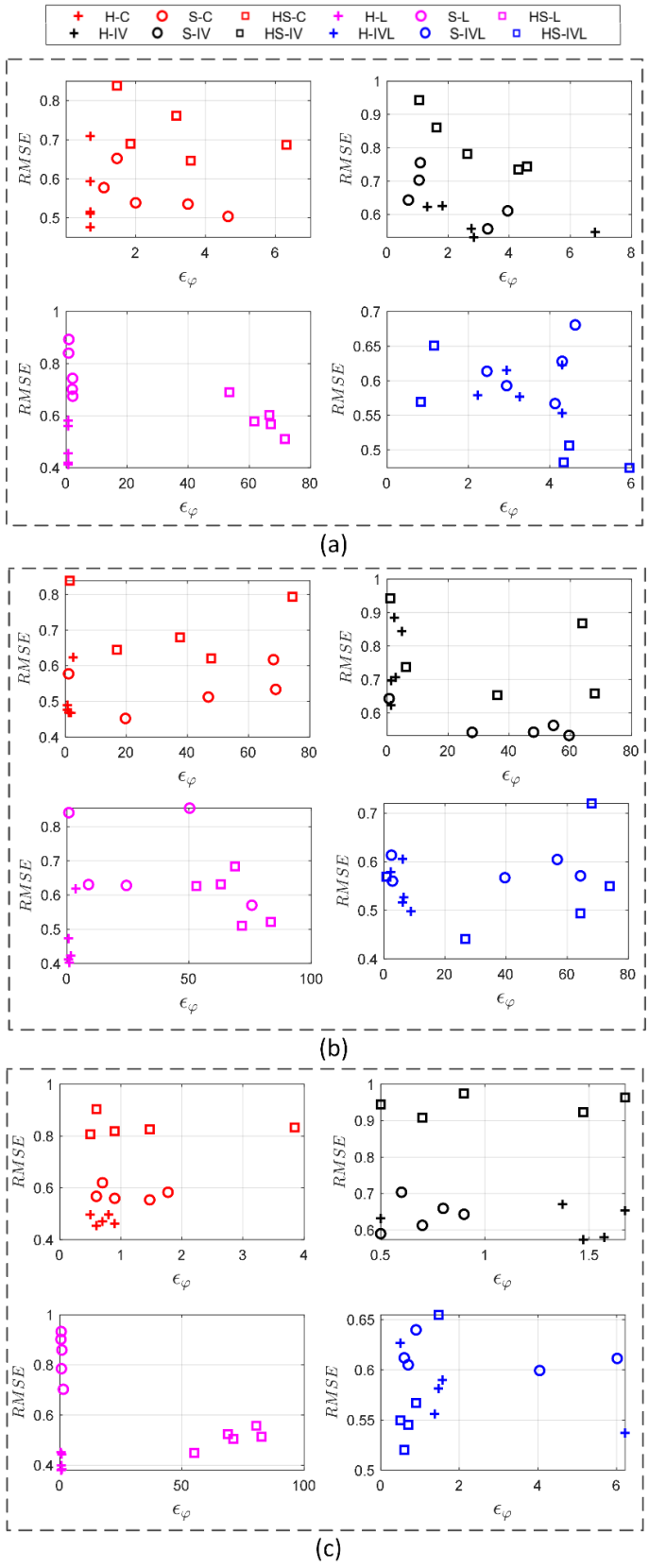


Figure 6.13 : Sensitivity analysis for (a) N_{bs}, (b) lr, (c) I^{φ}.

6.2.1.2 Parameter sensitivity analysis

In this analysis, we investigated the sensitivity of the composite learning approach for IT2-FLSs with respect to three hyperparameters: N_{bs}, lr, and φ. We performed a grid search to determine the optimal values for these hyperparameters, excluding the one we aimed to test for sensitivity. The experiments were repeated five times, and we calculated the average RMSE and $\epsilon_{\varphi} = |\varphi_d - \varphi|$, where φ_d represents the set expected coverage I^{φ}. To assess the robustness of a model to a particular hyperparameter, we examined the variance of the results. If the results cluster tightly with low variance, we consider the model robust to that hyperparameter. Conversely, if the results scatter widely with high variance, it indicates sensitivity to that hyperparameter. The sensitivity plots, shown in Figure 6.13, provide visual representations of the sensitivity analysis for each hyperparameter. By examining these plots, we can determine the impact of each hyperparameter on the model's performance and identify the robustness or sensitivity of the model to each hyperparameter.

Figure 6.13a demonstrates the sensitivity analysis of N_{bs} for IT2-FLSs, where N_{bs} takes values from the set {8, 16, 32, 64, 128}. It can be observed that IT2-FLSs with C/L rule consequents exhibit a more robust performance compared to those with IV/IVL rule consequents. This suggests that the choice of N_{bs} has a relatively smaller impact on the accuracy and uncertainty representation of IT2-FLSs with C/L rule consequents. On the other hand, IT2-FLSs with HS type IT2-FSs are more sensitive to variations in N_{bs}. This sensitivity can be attributed to the fact that HS type IT2-FSs have a higher number of design parameters, making them more susceptible to changes in N_{bs}. Consequently, careful selection of N_{bs} is crucial for achieving optimal performance with IT2-FLSs utilizing HS type IT2-FSs.

Figure 6.13b displays the sensitivity analysis of lr (learning rate) for IT2-FLSs, with values ranging from {0.001, 0.002, 0.01, 0.1, 0.2}. Notably, the H-C and H-L models exhibit a more stable performance, as evidenced by their clustered results with minimal variance. This suggests that the choice of lr has a relatively smaller impact on the robustness and performance of these models, as their results remain centered around a consistent value. On the other hand, IT2-FLSs with IV/IVL rule consequents (H-IV and H-IVL) demonstrate a higher level of robustness compared to their counterparts. This can be attributed to the fact that IT2-FLSs with H-type IT2-FSs have fewer

learnable parameters and constraints. Consequently, the impact of lr on these models is less pronounced, resulting in more consistent and stable performance across different lr values. These findings align with our expectations, as IT2-FLSs utilizing H-type IT2-FSs inherently have a simpler and more constrained structure, which leads to reduced sensitivity to changes in lr. Conversely, IT2-FLSs with S and HS type IT2-FSs have a higher number of learnable parameters and less constrained structures, making them more sensitive to variations in lr.

Figure 6.13c illustrates the sensitivity analysis of the expected coverage I^{φ} for IT2-FLSs, with φ values ranging from 50% to 90% in increments of 10%. The sensitivity test allows us to examine how the models respond to variations in the expected coverage parameter. In the analysis, we observe similar trends as in the previous sensitivity analyses. The H-C and H-L models exhibit a more robust performance, with their results clustered together and minimal scattering. This indicates that these models are less sensitive to changes in the expected coverage parameter, as their performance remains relatively stable across different values of φ. On the other hand, the IT2-FLSs with IV/IVL rule consequents show slightly higher sensitivity to changes in the expected coverage parameter. This is expected, as these models have more complex structures and more learnable parameters compared to the H-C and H-L models. As a result, their performance may vary to a greater extent when the expected coverage is adjusted.

6.2.2 Comparative results for intra-model comparison

In Figure 6.14, we present the testing performances of IT2-FLSs for the remaining multivariate datasets, focusing on the last 30 epochs to better visualize the performance differences. This intra-model comparison allows us to analyze how the different IT2-FLSs perform on various datasets with varying characteristics. Our observations indicate that the performance of IT2-FLSs with C/L consequents is generally better for small-sized datasets such as Servo and MCPU. These datasets likely have simpler patterns and lower dimensionality, allowing the C/L consequents to capture the underlying relationships effectively. However, as the dataset's feature dimension and sample size increase, such as in the case of White-Wine and PowerPlant, the performance of IT2-FLSs with C/L consequents tends to decrease. This suggests that the complexity of these datasets requires a more expressive modeling approach.

In contrast, IT2-FLSs with IV/IVL consequents struggled to learn medium-sized datasets with moderate feature dimensions like PM10, BH, and Concrete. These datasets likely have a higher level of complexity and may exhibit nonlinear relationships that are challenging to capture with the IV/IVL consequents. However, IT2-FLSs with IV/IVL consequents performed better with high-sampled datasets like PowerPlant and White-Wine. These datasets may have larger amounts of training data, enabling the IV/IVL consequents to capture more intricate patterns. Additionally, we observed that IT2-FLSs with IVL consequents exhibited signs of overfitting on small-sampled datasets and underfitting on medium-scale datasets with high-dimensional feature spaces. This behavior suggests that the IVL consequents may be prone to over-parameterization when the dataset size is limited, leading to overfitting. Conversely, on medium-sized datasets with high-dimensional feature spaces, the IVL consequents may not have enough capacity to capture the underlying relationships accurately, resulting in underfitting.

Overall, this intra-model comparison provides valuable insights into the performance variations of IT2-FLSs across different datasets. Understanding these performance differences helps in selecting the appropriate IT2-FLS variant based on the dataset characteristics, ensuring better modeling and prediction accuracy.

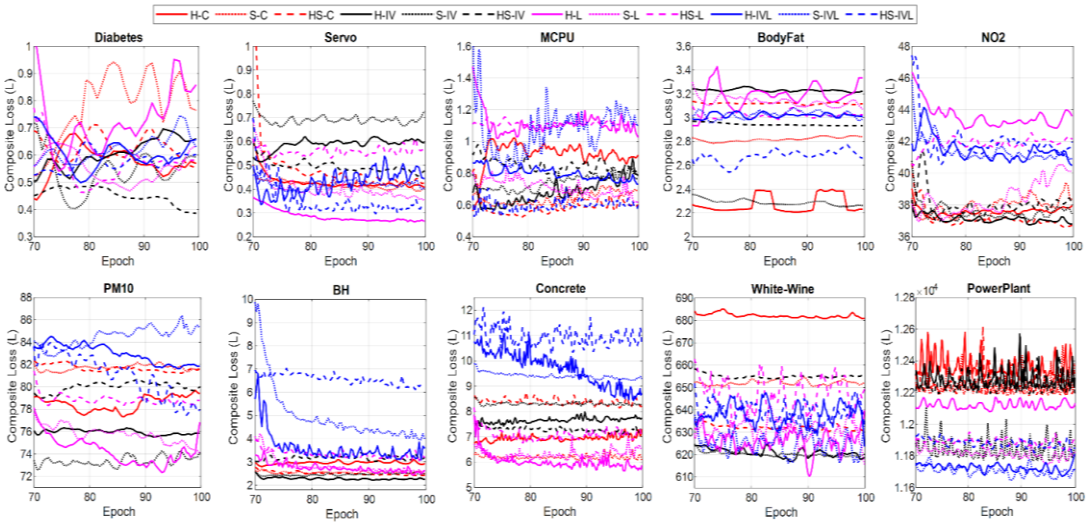


Figure 6.14 : Testing composite loss values of the proposed composite loss for all datasets.

Table 6.7 provides the performance measures for each dataset, allowing us to compare the performance of different IT2-FLS variants. Based on the observations, we can draw the following conclusions: For small-scale problems like Diabetes, Servo, and

Bodyfat, IT2-FLSs that incorporate uncertainty in their antecedents (such as S or HS types) demonstrate satisfactory performance.

Table 6.7 : Testing Performance Measures: Multivariate Datasets.

		Diabetes				Servo			
		C	**L**	**IV**	**IVL**	**C**	**L**	**IV**	**IVL**
RMSE*	**H**	13.7	16.8	13.6	15.5	3.7	1.9	4.7	4.4
	S	15.8	12.7	14.4	15.1	3.9	3.5	5.7	4.4
	HS	14.4	13.0	9.9	15.4	3.5	5.4	3.8	3.1
PICP	**H**	66.7	44.4	55.6	66.7	87.9	90.9	63.6	66.7
	S	55.6	55.6	55.6	66.7	72.7	72.7	60.6	66.7
	HS	77.8	66.7	100.0	66.7	75.8	69.7	63.6	78.8
		MCPU				**Bodyfat**			
		C	**L**	**IV**	**IVL**	**C**	**L**	**IV**	**IVL**
RMSE*	**H**	7.0	8.1	6.0	6.5	11.0	16.4	16.3	16.1
	S	4.9	5.2	6.6	8.9	14.9	16.2	12.2	16.2
	HS	4.9	10.4	10.5	5.0	15.7	16.1	12.7	12.3
PICP	**H**	81.0	76.2	83.3	81.0	82.0	72.0	72.0	80.0
	S	71.4	83.3	76.2	69.0	74.0	76.0	74.0	70.0
	HS	88.1	92.9	95.2	83.3	78.0	76.0	72.0	64.0
		NO2				**PM10**			
		C	**L**	**IV**	**IVL**	**C**	**L**	**IV**	**IVL**
RMSE*	**H**	45.6	49.6	45.2	48.1	77.9	73.6	75.4	78.4
	S	45.9	46.4	44.2	49.0	78.1	72.6	73.5	81.4
	HS	43.5	45.2	44.7	47.4	78.4	75.8	77.2	77.4
PICP	**H**	82.7	81.3	79.3	76.0	75.3	74.0	76.7	76.7
	S	79.3	81.3	87.3	80.0	76.0	76.7	73.3	72.7
	HS	85.3	82.0	87.3	82.7	83.3	73.3	76.0	72.0
		Housing				**Concrete**			
		C	**L**	**IV**	**IVL**	**C**	**L**	**IV**	**IVL**
RMSE*	**H**	7.3	6.5	5.5	8.0	8.7	7.7	9.9	9.8
	S	7.0	7.1	6.6	10.9	8.1	8.1	10.4	11.1
	HS	6.7	8.7	8.8	13.4	10.2	9.0	9.0	13.3
PICP	**H**	83.2	85.1	85.1	84.2	69.4	76.7	78.2	80.6
	S	82.2	80.2	76.2	84.2	83.0	70.9	76.7	80.1
	HS	83.2	79.2	74.3	78.2	81.1	79.1	81.6	86.9
		White-Wine				**PowerPlant**			
		C	**L**	**IV**	**IVL**	**C**	**L**	**IV**	**IVL**
RMSE*	**H**	71.6	69.2	68.9	69.9	434.1	427.2	437.1	418.7
	S	70.8	69.3	68.3	68.5	433.8	422.8	423.1	418.1
	HS	69.0	70.1	69.5	70.1	435.0	421.9	432.7	421.4
PICP	**H**	76.2	79.8	80.2	76.2	77.4	78.9	78.2	78.8
	S	73.7	78.7	79.9	79.8	78.7	78.6	78.6	79.9
	HS	77.9	74.9	79.0	79.0	80.9	78.4	79.2	78.5

* RMSE values are scaled by 100.

This suggests that capturing the uncertainty in the antecedents is beneficial for these datasets, potentially allowing the models to handle the inherent noise and variability

in the data. Consequently, constructing IT2-FLSs with crisp consequents seems to be a suitable choice for small-scale problems. However, for datasets with high feature dimensions like White-Wine and Housing, IT2-FLSs with IV consequents perform better than their crisp counterparts. The increased dimensionality of these datasets may introduce more complex and nonlinear relationships that can be better captured by the flexibility of IV consequents. By allowing a range of fuzzy membership values for the consequents, the models can adapt to the intricate patterns present in high-dimensional datasets.

On the other hand, IT2-FLSs with IVL consequents exhibit subpar performance when the dataset has a high feature dimension and a small number of samples, as observed in Bodyfat and Housing. This situation is likely due to the limited number of samples available for training the high number of learnable parameters in the IVL consequents. The models may suffer from overfitting, where they become too specific to the training data and struggle to generalize to unseen samples. Therefore, caution should be exercised when using IT2-FLSs with IVL consequents on datasets with limited samples and high feature dimensions. Interestingly, the performance of IT2-FLSs with IVL consequents improves as the sample size increases, as demonstrated in the PowerPlant dataset. With a larger number of samples, the models have more data points to learn from and can effectively utilize the additional parameters in the IVL consequents. This highlights the importance of having an adequate sample size when employing IT2-FLSs with IVL consequents, particularly for datasets with high feature dimensions.

In summary, the choice of IT2-FLS variant depends on the characteristics of the dataset. For small-scale problems, IT2-FLSs with crisp consequents can provide satisfactory performance. In contrast, for datasets with high feature dimensions, IT2-FLSs with IV consequents tend to perform better. However, caution should be exercised when using IT2-FLSs with IVL consequents on datasets with limited samples and high feature dimensions, as they may be prone to overfitting.

6.2.3 Comparative results for inter-model comparison

Table 6.8 provides comparative results to evaluate the performances of IT2-FLSs in comparison with various state-of-the-art approaches, including SVR, SVM, DNN-based QR, NN-based QR, RWN, RL, ELM, and GS. The best performing IT2-FLS

variant with KM is selected based on both PICP and RMSE criteria for each dataset. The results in Table VI demonstrate the performance of IT2-FLSs relative to other approaches across different datasets. By considering both PICP and RMSE, the selected IT2-FLS-KM consistently achieves competitive results compared to the state-of-the-art methods. This indicates that the proposed composite learning approach for IT2-FLSs, utilizing the KM CSCM, is effective in achieving accurate predictions while capturing uncertainty. Moreover, the comparison allows us to analyze the impact of the CSCM on performance. The results show that IT2-FLSs with KM generally outperform IT2-FLSs with NT and SM, indicating that the KM CSCM is more effective in modeling the uncertainty and achieving better prediction accuracy. This highlights the importance of the proposed composite learning approach and the utilization of the KM CSCM in IT2-FLSs.

The observed results indicate that IT2-FLSs with KM consistently achieve the best expected coverage, as indicated by their high PCIP values, across most datasets. Although they may not always yield the best RMSE values, the IT2-FLSs with KM outperform their counterparts when considering multiple objectives. In terms of accuracy alone, the IT2-FLSs learned through the proposed composite learning approach generally outperform non-fuzzy approaches for most datasets. However, it is important to note that this outcome is a natural consequence of the trade-off between coverage and accuracy embedded in our multi-objective loss function, as compared to the single-objective minimization employed by the non-fuzzy methods. Overall, the comparative analysis highlights the effectiveness of IT2-FLSs learned using the proposed approach, particularly in achieving a satisfactory balance between accuracy and uncertainty representation.

In summary, the proposed learning approach for IT2-FLSs enables them to effectively represent uncertainty while surpassing the performance of the compared approaches, including DNNs, in terms of accuracy measures.

Table 6.8 : Comparative Testing Performance Measures: Inter-Model Comparison.

		RMSE*	PICP			RMSE*	PICP			RMSE*	PICP
Mcycle	S-IV-IT2-FLS-KM	44.0	81.81	**Diabetes**	HS-C-IT2-FLS-KM	14.4	77.8	**Servo**	HS-IVL-IT2-FLS-KM	**3.1**	78.7
	HS-L-IT2-FLS-NT	44.0	79.64		HS-C-IT2-FLS-NT	11.7	66.7		HS-L-IT2-FLS-NT	3.2	78.7
	H-L-IT2-FLS-SM	**43.5**	79.54		H-IV-IT2-FLS-SM	12.0	66.7		H-IVL-IT2-FLS-SM	3.6	75.7
	DEEPJMQR[108]	51.5	**		LAD-SVR[110]	10.9	**		LAD-SVR[110]	9.7	**
	Linear QR[108]	108.6	**		RWN_{AFt}[111]	**8.4**	**		RWN_{AFt} [111]	9.5	**
	Indep. DL [108]	53.0	**		RC-ELM[112]	13.0	**		Res-TSVR[114]	11.8	**
	CQRNN[109]	68.5	**		NSLVReg[113]	16.9	**		RC-ELM[112]	7.0	**
MCPU	HS-IVL-IT2-FLS-KM	5.0	83.3	**Bodyfat**	H-C-IT2-FLS-KM	11.0	82.0	**NO2**	H-IV-IT2-FLS-KM	45.2	79.3
	S-IV-IT2-FLS-NT	5.2	83.3		S-IVL-IT2-FLS-NT	**8.3**	72.0		S-IV-IT2-FLS-NT	**44.2**	84.6
	H-IV-IT2-FLS-SM	4.4	76.1		S-L-IT2-FLS-SM	11.0	68.0		HS-C-IT2-FLS-SM	45.1	80.6
	LAD-SVR[110]	9.1	**		LAD-SVR[110]	14.8	**		IGS[117]	87.0	**
	RC-ELM[112]	**2.0**	**		Res-TSVR[114]	8.3	**		MixRL[118]	49.3	**
	NSLVReg[112]	6.0	**		kNNGNN[115]	12.0	**		RMR-ℓ_2[119]	47.6	**
	SNLR-MoG[116]	8.5	**		LS-SVR[110]	16.9	**		ε-FHTSVR[120]	50.0	**
Housing	H-IV-IT2-FLS-KM	**5.5**	85.1	**Concrete**	S-C-IT2-FLS-KM	**8.1**	83.0	**Wihte-Wine**	S-IVL-IT2-FLS-KM	68.5	79.8
	H-IV-IT2-FLS-NT	6.0	76.2		S-C-IT2-FLS-NT	8.5	78.1		HS-IV-IT2-FLS-NT	**68.3**	79.3
	HS-IV-IT2-FLS-SM	5.8	73.2		H-IV-IT2-FLS-SM	8.3	79.1		S-IV-IT2-FLS-SM	68.5	80.2
	LAD-SVR[110]	7.4	**		LAD-SVR[110]	8.6	**		IGS[117]	84.0	**
	Res-TSVR[114]	9.4	**		Res-TSVR[114]	9.3	**		IQR Fit [121]	69.1	**
	RC-ELM[112]	12.0	**		SNLR-MoG[116]	10.2	**		RBF-kernel SVM[122]	77.5	**
	kNNGNN[115]	21.0	**		WLS-SVR-L[110]	8.9	**		SDE-MKLSSVM 123]	87.8	**

* RMSE values are scaled by 100

** Not provided in the paper

The IT2-FLSs trained with the proposed approach exhibit a favorable trade-off between accuracy and uncertainty representation, making them a powerful tool for modeling and prediction tasks where both aspects are crucial.

6.3 Comparative Performance Analysis for the GT2-FLSs

Here, we present the performances of the towards reliable uncertainty quantification and high precision for GT2-FLS trained with L_1 (GT2-FLS-1) and L_2 (GT2-FLS-2), which are presented in section 5, in comparison with the IT2-FLS, introduced in section 4, on the benchmark datasets that are given in Table 6.9. We constructed the GT2-FLSs with $K = 2$, i.e. the total number of α-planes is 3. and defined all T2-FLSs with $P = 5$ rules. We used the product operator as the t-norm for the MCPU and Concrete datasets while the minimum operator for the BH. We set the desired uncertainty coverage as 80% ($\varphi = [0.1, 0.9]$).

Table 6.9 : Datasets Details and Hyperparameters for the analysis of the trained GT2-FLSs using composite loss.

Dataset	M	N	N_{train}	N_{test}	N_{bs}	lr
Machine CPU (MCPU)	6	209	168	41	16	0.05
Boston Housing (BH)	13	506	405	101	64	0.1
Concrete	8	1030	824	206	64	0.1

We normalized the datasets with the max-min normalization method and then trained all models for 100 epochs in Matlab ®. All experiments were repeated with five different initial seeds for statistical analysis. The performances are evaluated via the Root Mean Square Error (RMSE), PI Coverage Probability (PICP), and PI Normalized Averaged Width (PINAW) [124]. It is worth stating that the aim is to end with a model that has a low RMSE (high point-wise estimation) while having a high PICP and low PINAW value (a PI with narrow bands). As we set $\mathrm{I}^{\varphi} = 80\%$, we analyzed the uncertainty quantification performance via the following metric:

$$\Delta\mathrm{PICP} = 80 - \mathrm{PICP} \tag{6.3}$$

If $\Delta\mathrm{PICP} < 0$, then the uncertainty quantification performance is higher than expected, and vice versa. We refer to $\Delta\mathrm{PICP} < 0$ as overbounding and $\Delta\mathrm{PICP} > 0$ as underbounding.

The performance statistics of the T2-FLSs are analyzed via the box-and-whisker plots visualizing the median, max, min, and Inter Quartile Range (IQR) which is a measure of spread defined as the distance between the 75th and 25th percentiles of the data. The results of the T2-FLSs employing KM CSCM and NT CSCM are given in Figure 6.15 and Figure 6.16, respectively.

From the results of the T2-FLSs employing KM-CSCM presented in Fig. 2, we can observe that:

- The reliability of the uncertainty quantification of the GT2-FLSs is better than the IT2-FLS since the spreads of ΔPICP values (i.e. the size of the IQR box) of the GT2-FLSs are significantly smaller than their IT2 counterpart. For the MCPU and BH datasets, the GT2-FLSs resulted also in better median ΔPICP values, excluding the Concrete dataset in which the IT2-FLS is better. Yet, the IT2-FLS not only resulted in a large IQR for ΔPICP but also a large PINAW distribution. Thus, one might claim that the reliability of the GT2-FLSs is better than the IT2-FLS.
- The point-wise estimation performance of the GT2-FLSs is almost always better than its IT2 counterpart. Especially from the results of the BH dataset (high dimensional data) in Figure 6.15b. we can observe that GT2-FLS-2 has resulted in a smaller median value for RMSE than GT2-FLS-1 and IT2-FLS (both in testing and training) while also resulting more robust as the IQRs of the GT2-FLS-2 are smaller in all metrics.

We can make similar observations from the results of the T2-FLSs with NT CSCM given in Figure 6.16. Firstly, it is obvious that through the deployment of NT CSCM, the widths of IQRs have increased for all metrics when compared to the results of the KM ones. We can also state that:

- The spread size of ΔPICP for IT2-FLS is relatively smaller when compared to the GT2-FLSs. Yet, IT2-FLS underbounds as $\Delta\text{PICP} > 0$ while the ones of GT2-FLS-2 are distributed around $\Delta\text{PICP} = 0$. Also, the median ΔPICP of the GT2-FLS-2 is generally better than the IT2-FLS. Thus, one might prefer GT2-FLS-2 for its uncertainty quantification performance.

- The GT2-FLS-2 has resulted in low RMSE median values while having a small IQR size when compared to the other T2-FLSs. Hence, it can be stated that it has a more robust accuracy performance than GT2-FLS-1 and IT2-FLS.

To sum up, although the IT2-FLS resulted in acceptable performances in training, it struggled in constructing PIs with expected I^{φ} for all the datasets seen in testing regardless of the applied CSCM. Thus, we can conclude that the reliability of IT2-FLSs is relatively low for the handled datasets, obviously depending on the IT2-FLS structure as different antecedent and/or consequent MFs might perform better. On the other hand, we observed the GT2-FLSs performed better than the IT2-FLSs in general, regardless of the deployed CSCM method. More specifically, we can also state that the GT2-FLS-2 performed better than the other T2-FLSs (GT2-FLS-1 and IT2-FLS). The results indicate that it is possible to learn reliable GT2-FLS via the loss L_2 in which different roles for the α_k-IT2-FLS are assigned that are defined via the SMF shape and size parameters.

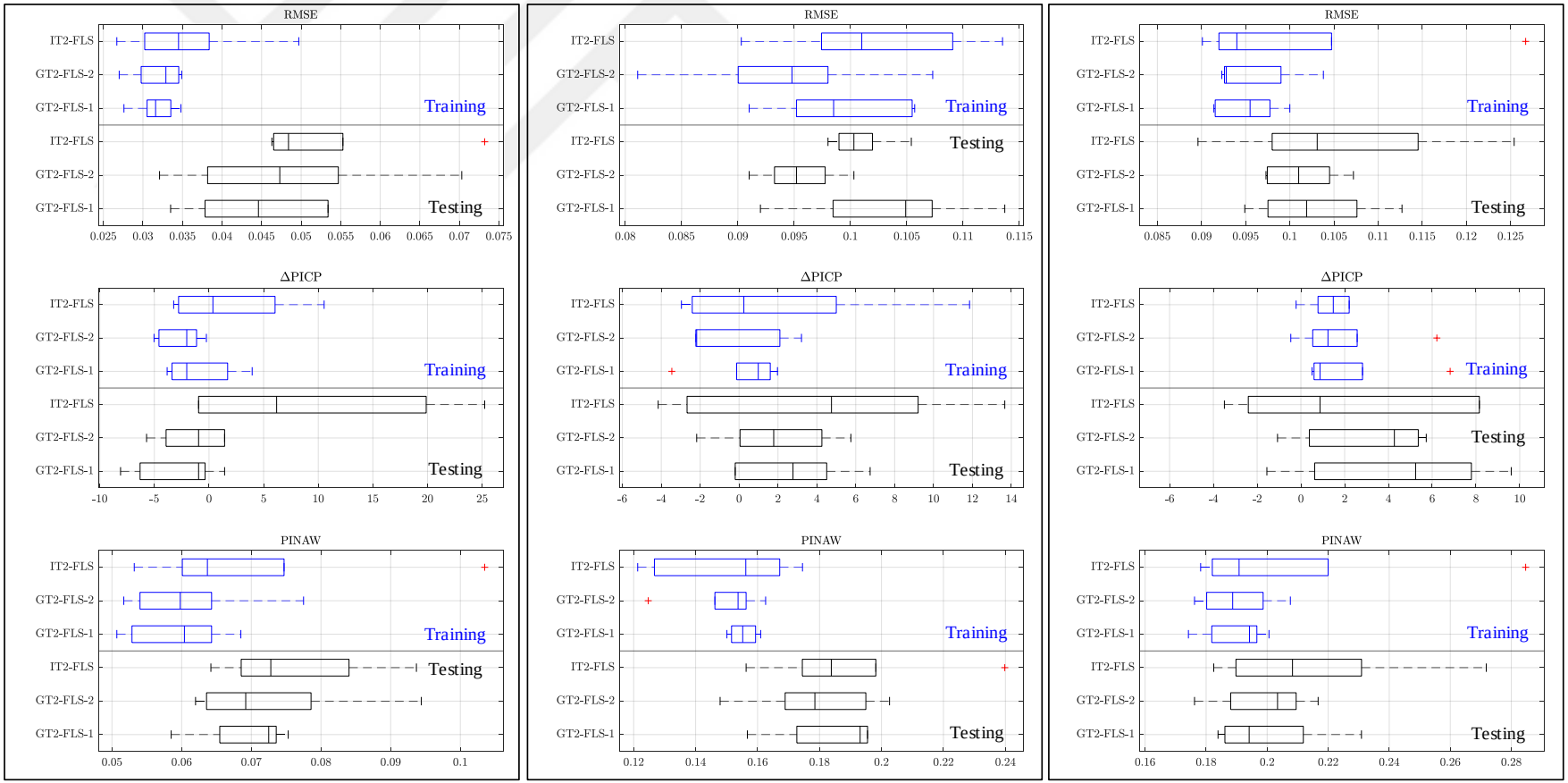


Figure 6.15 : Box-and-whisker plots of T2-FLSs with KM CSCM showing median (central mark), 25th/75th percentiles (left and right edges of box), whiskers (dashed line), and outliers (red plus sign): (a) MCPU (b) BH (c) Concrete datasets.

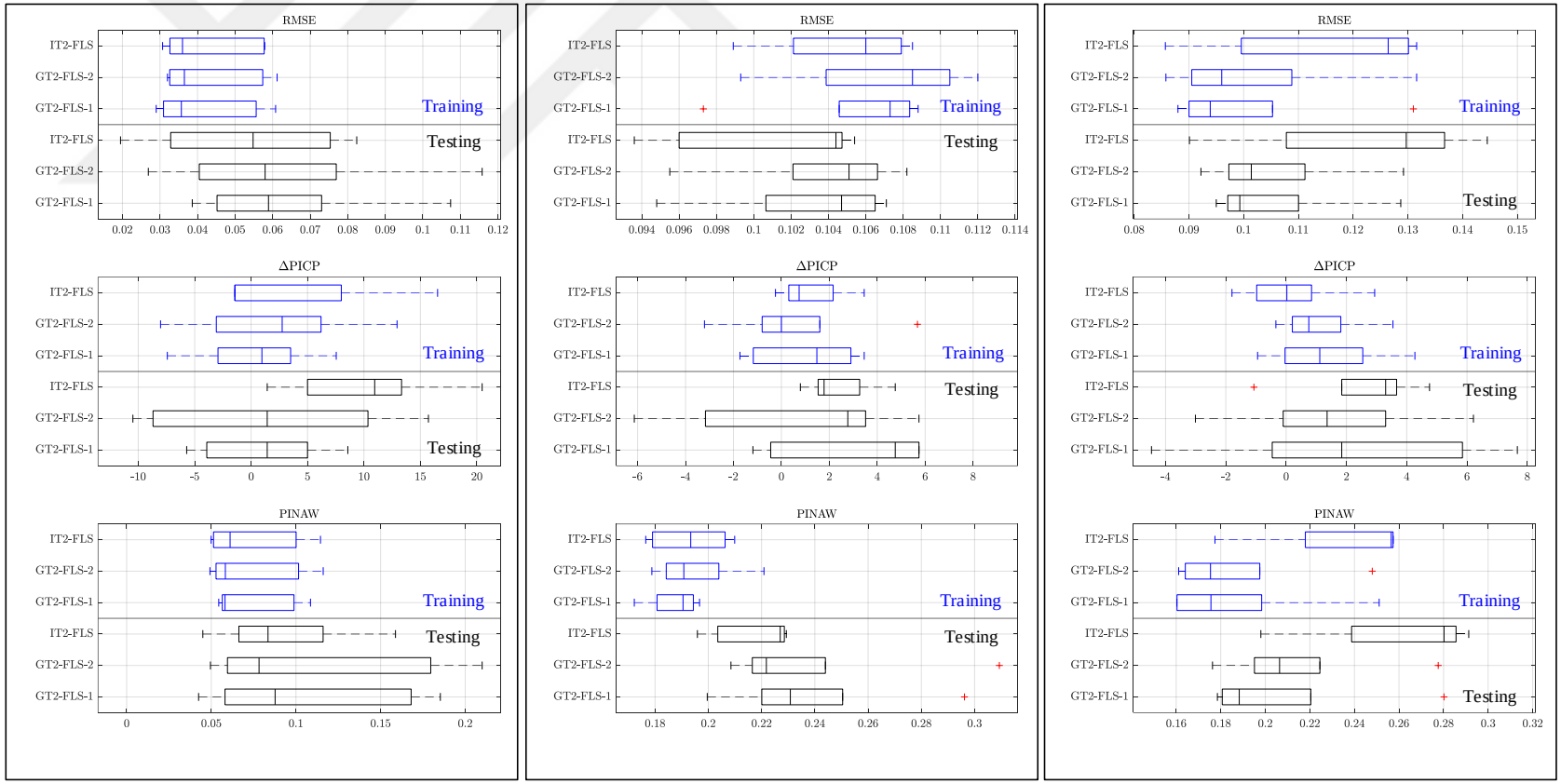


Figure 6.16 : Box-and-whisker plots of T2-FLSs with NT CSCM showing median (central mark), 25th/75th percentiles (left and right edges of box), whiskers (dashed line), and outliers (red plus sign): (a) MCPU (b) BH (c) Concrete datasets.

7. CONCLUSION

This thesis presents an innovative approach to learning the parameters of T2-FLSs, which include IT2-FLSs and GT2-FLSs. The proposed method leverages popular deep learning parameter learning techniques, as described in Section 3 of the thesis. To address the challenges faced by conventional parameter learning methods for T1/T2-FLSs, such as the curse of dimensionality and the increasing number of parameters associated with rule size, a mini-batch sampled deep learning parameter learning method is introduced.

The proposed approach employs a flexible structure that can be utilized in various ways to achieve models with both high prediction accuracy and the ability to handle uncertainties. The thesis comprises three distinct studies. The first study, titled "Learning with Type-2 Fuzzy Activation Functions to Improve the Performance of Deep Neural Networks," suggests utilizing Single input IT2-FLS as activation functions in DNNs to enhance their learning performance.

In the second study, titled "More than Accuracy: A Composite Learning Framework for Interval Type-2 Fuzzy Logic Systems," a novel composite learning framework is introduced. This framework is specifically designed for training regression models using IT2-FLSs, with the primary objective of achieving high accuracy performance while effectively representing and handling uncertainties.

The final work, titled "Towards Reliable Uncertainty Quantification and High Precision with General Type-2 Fuzzy Systems," takes a significant step towards learning reliable uncertainty quantification and achieving high precision performance. This is accomplished through the utilization of α-plane based General Type-2 GT2-FLSs.

Overall, this thesis presents a novel perspective on learning the parameters of T2-FLSs by combining deep learning methods with fuzzy logic systems. The studies conducted in this thesis aim to enhance learning performance, address uncertainties, and achieve high precision in a variety of applications.

The introduced activation function in “Learning with Type-2 Fuzzy Activation Functions to Improve the Performance of Deep Neural Networks”, SIT2-FRU, presents a unique design that uses the parameters of membership functions to define slopes of the in both positive and negative quadrants. These parameters not only shape the input-output mapping but also play a crucial role in generating negative valued outputs. By allowing negative outputs, SIT2-FRU addresses the vanishing gradient problem and assists in shifting the mean activation of the layer towards zero, similar to other popular activation functions like ELU and PReLU. Moreover, SIT2-FRU offers an additional advantage by enabling the learning of each neuron's contribution in the positive quadrant. This feature allows DNNs to improve their learning behavior and increase convergence speed. By accurately capturing the importance of individual neuron outputs in the positive region, SIT2-FRU enhances the overall performance of DNNs. In summary, the SIT2-FRU activation function provides benefits such as generating negative outputs, alleviating the vanishing gradient problem, and enhancing learning behavior and convergence speed in DNNs. These characteristics make SIT2-FRU a valuable tool for improving the performance of neural networks in various applications. The conducted experiments also validated our analyses and showed that:

- In the case of the MNIST dataset, it was observed that DNNs utilizing ELUs and ReLUs in combination with Batch Normalization (BN-DNNs) faced difficulties in effectively learning the dataset. The performance of BN-DNNs with ELUs was particularly inadequate, and DNNs with ReLUs also exhibited a significant decline in performance across different hyperparameter configurations. Conversely, DNNs, both with and without Batch Normalization (BN-DNNs and DNNs), employing the IT2-FAL (Interval Type-2 Fuzzy Adaptive Learning) activation function demonstrated a consistent and strong learning behavior, regardless of the specific hyperparameter settings. These findings highlight the superior performance and robustness of DNNs utilizing the IT2-FAL activation function compared to those employing ELUs and ReLUs when applied to the MNIST dataset.
- In the case of the Quickdraw Pictionary dataset, it was observed that S-DNNs utilizing ReLUs, PReLUs, and ELUs as activation functions exhibited a divergence between training and testing losses after 1,000 updates. This divergence is an indication of overfitting problem, where the model becomes

overly specialized to the training data and performs poorly on unseen data. In contrast, S-DNNs employing the IT2-FAL activation function did not encounter the overfitting issue. This implies the robust learning behavior of IT2-FAL, which is facilitated by its ability to generate various FMs. This also allows the IT2-FAL-based S-DNN to handle uncertainties in data and avoid the overfitting problem experienced by other activation functions. Thus, the IT2-FAL activation function proves to be advantageous in maintaining a balanced learning behavior and preventing overfitting in S-DNNs trained on the Quickdraw Pictionary dataset.

- In the CIFAR-10 dataset, it was observed that the BN layers generally led to a significant improvement in the performance of ReLU, PReLU, and ELU activation functions. The addition of BN layers helped to alleviate issues such as vanishing or exploding gradients, leading to enhanced learning and better classification results. In contrast, the SIT2-FRU activation function not only demonstrated satisfactory learning performance with or without BN layers but also achieved the best classification performance among the tested activation functions. It was found that SIT2-FRUs exhibited a high level of generalization capability, enabling them to perform well even in stacked DNNs with multiple convolutional layers. This highlights the superior performance and generalization capability of SIT2-FRUs, as they were able to achieve high learning performance and classification accuracy in complex DNN architectures, surpassing the other activation functions tested on the CIFAR-10 dataset.

The introduced study titled with “More than Accuracy: A Composite Learning Framework for Interval Type-2 Fuzzy Logic Systems”, present a new learning method IT2-FLSs to achieve high mean accuracy and effectively handle uncertainty. The proposed method addressed the design challenges associated with IT2-FLSs, including the definition of IT2-FLSs for uncertainty handling, the construction of a composite loss function, and the optimizing the proposed composite loss using the DL frameworks. The proposed approach provided detailed solutions for each challenge, aiming to enhance the accuracy of predictions while capturing and representing uncertainty. The method offered comprehensive insights into the methodology,

showcasing its potential for improving the performance of IT2-FLSs in various applications that require accurate and reliable modeling and prediction.

During the conducted sensitivity analyses, intra-model comparisons, and inter-model comparisons, we evaluated the performance of our proposed learning approach. The results of these analyses clearly demonstrated that training IT2-FLSs using our approach yielded impressive outcomes in terms of both accuracy and uncertainty coverage. However, when comparing different structures of rule antecedents and consequents, no clear winner emerged in terms of uncertainty representation. This suggests that the selection of MFs shapes remains an open question that should be addressed based on the specific properties of the data, similar to determining the number and structure of hidden layers and neurons in DNNs. Furthermore, we observed that employing the KM method as the CSCM for uncertainty processing generally resulted in better RMSE and PICP values. The KM method calculates the type-reduced set mixture's UMF and LMF, which implicitly introduces an additional regularization term to the loss function. This enhanced alignment between the logarithmic hyperbolic loss ($L_{log-cosh}$) and the tilted loss (l_τ), contributing to improved performance. In summary, our comparative analyses highlighted the effectiveness of our learning approach for IT2-FLSs in terms of accuracy and uncertainty coverage. While the choice of MF shapes remains an open question, the utilization of the KM method for uncertainty processing generally yielded better results in terms of RMSE and PICP values.

In the study titled with "Towards Reliable Uncertainty Quantification and High Precision with General Type-2 Fuzzy Systems", we introduce a learning method specifically designed for α-plane based GT2-FLSs with the aim of achieving accurate pointwise predictions and constructing reliable PIs. Our proposed learning algorithm utilizes a novel composite loss function comprising two terms: one for accuracy and the other for uncertainty quantification. To quantify uncertainty, we explicitly leverage the size parameters of SMFs, while for the accuracy task, we provide two alternative definitions of pointwise error, in which the shape parameters are learned in different ways (referred to as GT2-FLS-2 and GT2-FLS-1). The proposed approach exhibits a partially independent characteristic for our multi-objective task.

The results of our experiments demonstrate that GT2-FLS-2 outperforms GT2-FLS-1 and IT2-FLS in larger datasets. However, in smaller datasets, GT2-FLS-1 shows

slightly better performance compared to its counterparts. Therefore, we suggest utilizing the loss L_2, as the learned GT2-FLS-2 yields more robust and accurate results compared to other T2-FLSs in many cases.

REFERENCES

[1] **LeCun, Y., Haffner, P., & Bottou, L.** (2015). Bengio Y. *Hinton G: Deep learning. nature*, *521*, 436-444.

[2] **Schmidhuber, J.** (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85-117.

[3] **Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., & Quillen, D.** (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, *37*, 421-436.

[4] **Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P.** (2014). Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*.

[5] **Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y.** (2013, May). Maxout networks. In *International conference on machine learning*, 1319-1327. PMLR.

[6] **Krizhevsky, A., Sutskever, I., & Hinton, G. E.** (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*, 84-90.

[7] **Girshick, R., Donahue, J., Darrell, T., & Malik, J.** (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 580-587.

[8] **Ciresan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J.** (2011, September). Convolutional neural network committees for handwritten character classification. In *2011 International conference on document analysis and recognition*, 1135-1139. IEEE.

[9] **Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P. A., & Bottou, L.** (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, *11*.

[10] **Masci, J., Meier, U., Cireşan, D., & Schmidhuber, J.** (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*, 52-59). Springer Berlin Heidelberg.

[11] **Nair, V., & Hinton, G. E.** (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807-814.

[12] **Krizhevsky, A., & Hinton, G.** (2009). Learning multiple layers of features from tiny images. Retrieved from: http://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf.

[13] **Kaggle**. (2019). CNN tutorial. Retrieved from: https://www.kaggle.com/nhlr21/deep-keras-cnn-tutorial.

[14] **Bishop, C. M., & Nasrabadi, N. M.** (2006). *Pattern recognition and machine learning*, *4*, 738.

[15] **Murphy, K. P.** (2012). *Machine learning: a probabilistic perspective*. MIT press.

[16] **Pascanu, R., Mikolov, T., & Bengio, Y.** (2013, May). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, 1310-1318. Pmlr.

[17] **Li, H. X., and Gatland, H. B.,** (1996). Conventional fuzzy control and its enhancement, *IEEE Transactions on Systems, Man, and Cybernertics*, *26*, 791–796.

[18] **Guo, Y., and Woo, P. Y.** (2003). An adaptive fuzzy sliding mode controller for robotic manipulators, *IEEE Transactions on Systems, Man, and Cybernertics-Part A: Systems and Humans*, *33*, 149–159.

[19] **Zadeh, L. A.** (1996). Fuzzy logic= computing with words, *IEEE Transactions on Fuzzy Systems*, *4*, 103–111.

[20] **Bellman, R. E. and Zadeh, L. A.** (1970). Decision-making in a fuzzy environment, *Management Science, 17*, B–141–B–164.

[21] **Bloch, I.** (2015). Fuzzy sets for image processing and understanding, *Fuzzy Sets and Systems*, *281*, 280–291.

[22] **Chiu, S. L.** (1994). Fuzzy model identification based on cluster estimation, *Journal of Intelligent & fuzzy systems*, *2*, 267–278.

[23] **Ross, T. J.** (2009). *Fuzzy logic with engineering applications*, John Wiley & Sons.

[24] **Mendel J. M.** (2017). *Uncertain rule-based fuzzy systems: introduction and new directions*, Springer.

[25] **Castillo, O., Melin, P.** (2008). *Type-2 fuzzy logic: theory and applications*, Springer.

[26] **Mendel, J. M., Hagras, H., Tan W.W., Melek, W.W., and Ying, H.** (2014). *Introduction to type-2 fuzzy logic control*, John Wiley and IEEE Press, Hoboken, NJ.

[27] **Mendel, J.** (2007). Type-2 fuzzy sets and systems: an overview, *IEEE Computational Intelligence Magazine*, *2*, 20–29.

[28] **Wu, D.** (2012). On the fundamental differences between interval type-2 and type-1 fuzzy logic controllers, *IEEE Transactions on Fuzzy Systems*, *20*, 832–848.

[29] **Mitra, S., Hayashi, Y.** (2000). Neuro-fuzzy rule generation: survey in soft computing framework, *IEEE Trans. Neural Networks*, *11*, 748–76 8.

[30] **Yen J., Wang L., Gillespie C.W.** (1998). Improving the interpretability of TSK fuzzy models by combining global learning and local learning, *IEEE Transaction on Fuzzy Systems*, *6*, 530–537.

[31] **Jang J.R.** (1993). ANFIS: adaptive-network-based fuzzy inference system, *IEEE transactions on systems, man, and* cybernetics, *23***,** 665–685.

[32] **J.S.R. Jang, C.T. Sun**, (1995). Neuro-fuzzy modeling and control, *Proceedings of the IEEE, 83*, 378–406.

[33] **Shihabudheen, K. V., & Pillai, G. N.** (2018). Recent advances in neuro-fuzzy system: A survey. *Knowledge-Based Systems*, *152*, 136-162.

[34] **Juang, C. F., & Tsao, Y. W.** (2008). A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning. *IEEE transactions on fuzzy systems*, *16*, 1411-1424.

[35] **Aliev, R. A., Pedrycz, W., Guirimov, B. G., Aliev, R. R., Ilhan, U., Babagil, M.,** & **Mammadli, S.** (2011). Type-2 fuzzy neural networks with fuzzy clustering and differential evolution optimization. *Information Sciences*, *181*, 1591-1608.

[36] **Tolue, S. F., & Akbarzadeh-T, M. R.** (2013, August). Dynamic fuzzy learning rate in a self-evolving interval type-2 tsk fuzzy neural network. In *2013 13th Iranian Conference on Fuzzy Systems (IFSC)*, 1-6.

[37] **Lin, C. T., Pal, N. R., Wu, S. L., Liu, Y. T., & Lin, Y. Y.** (2014). An interval type-2 neural fuzzy system for online system identification and feature elimination. *IEEE transactions on neural networks and learning systems*, *26*, 1442-1455.

[38] **Juang, C. F., Huang, R. B., & Cheng, W. Y.** (2010). An interval type-2 fuzzy-neural network with support-vector regression for noisy regression problems. *IEEE Transactions on fuzzy systems*, *18*, 686-699.

[39] **Rajurkar, S., & Verma, N. K.** (2017, July). Developing deep fuzzy network with Takagi Sugeno fuzzy inference system. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1-6.

[40] **Wu, D., Yuan, Y., Huang, J., & Tan, Y.** (2019). Optimize TSK fuzzy systems for regression problems: Minibatch gradient descent with regularization, DropRule, and AdaBound (MBGD-RDA). *IEEE Transactions on Fuzzy Systems*, *28*, 1003-1015.

[41] **R. Chimatapu, R., Hagras, H., Starkey, A., & Owusu, G.** (2018, July). Interval type-2 fuzzy logic based stacked autoencoder deep neural network for generating explainable ai models in workforce optimization. In *2018 IEEE international conference on fuzzy systems (FUZZ-IEEE)*, 1-8.

[42] **Islam, M. A., Anderson, D. T., Pinar, A. J., Havens, T. C., Scott, G., & Keller, J. M.** (2019). Enabling explainable fusion in deep learning with fuzzy integral neural networks. *IEEE Transactions on Fuzzy Systems*, *28*, 1291-1300.

[43] **Anderson, D. T., Scott, G. J., Islam, M. A., Murray, B., & Marcum, R.** (2018). Fuzzy choquet integration of deep convolutional neural networks for remote sensing. *Computational Intelligence for Pattern Recognition*, 1-28.

[44] **Bölat, K., & Kumbasar, T.** (2021, July). Integrating Interval Type-2 Fuzzy Sets into Deep Embedding Clustering to Cope with Uncertainty. In *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1-6.

[45] **Beke, A., & Kumbasar, T.** (2019). Learning with type-2 fuzzy activation functions to improve the performance of deep neural networks. *Engineering Applications of Artificial Intelligence*, *85*, 372-384.

[46] **Hosseini-Pozveh, M. S., Safayani, M., & Mirzaei, A.** (2020). Interval type-2 fuzzy restricted boltzmann machine. *IEEE Transactions on Fuzzy Systems*, *29*, 1133-1142.

[47] **Erdem, D., & Kumbasar, T.** (2021, July). Enhancing the Learning of Interval Type-2 Fuzzy Classifiers with Knowledge Distillation. In *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1-6.

[48] **Beke, A., & Kumbasar, T.** (2021, July). Capturing Uncertainty with Interval Fuzzy Logic Systems through Composite Deep Learning. In *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1-6.

[49] **Beke, A., & Kumbasar, T.** (2022). More than Accuracy: A Composite Learning Framework for Interval Type-2 Fuzzy Logic Systems. *IEEE Transactions on Fuzzy Systems*.

[50] **Avcı, B., Beke, A., & Kumbasar, T.** (2023). Towards Reliable Uncertainty Quantification and High Precision with General Type-2 Fuzzy Systems. In *2023 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1-6.

[51] **Nie, M., & Tan, W. W.** (2008, June). Towards an efficient type-reduction method for interval type-2 fuzzy logic systems. In *2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence)*, 1425-1432.

[52] **Wang, Q., Ma, Y., Zhao, K., & Tian, Y.** (2020). A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, 1-26.

[53] **Zadeh, L. A.** (1965). Fuzzy sets, *Information and* Control, *8*, 338–353.

[54] **Zadeh, L. A.** (1975). The concept of a linguistic variable and it application to approximate reasoning — I, *Information Science*, *19*, 829–841.

[55] **Aisbett, J., Rickard, J. T., and Morgenthaler, D. G.** (2010). Type-2 fuzzy sets as functions on spaces. *IEEE Transactions on Fuzzy Systems*, *18*, 3-15.

[56] **Coupland, S., and John, R.** (2007). Geometric type-1 and type-2 fuzzy logic Systems, *IEEE Transactions on Fuzzy Systems*, *15*, 3–15.

[57] **Liang, Q., and Mendel, J.** (2000). Interval type-2 fuzzy logic systems: theory and design, *IEEE Transactions on Fuzzy Systems*, *8*, 535–550.

[58] **Mendel, J., John, R., Liu, F.** (2006). Interval type-2 fuzzy logic systems made simple. *IEEE Transactions on Fuzzy Systems*, *14*, 808–821.

[59] **Wu, D., and Mendel, J.** (2011). On the continuity of type-1 and interval type-2 fuzzy logic systems, *IEEE Transactions on Fuzzy Systems*, *19*, 179–192.

[60] **Jammeh, E. A., Fleury, M., Wagner, C., Hagras, H., Ghanbari, M.** (2009). Interval type-2 fuzzy logic congestion control for video streaming across IP networks, *IEEE Transactions on Fuzzy Systems*, *17*, 1123–1142.

[61] **Karnik, N. N. and Mendel, J. M.** (2001). Centroid of a type-2 fuzzy set. *Information Sciences*, *132*, 195-220.

[62] **Sakalli, A., Beke, A., and Kumbasar, T.** (2016). Gradient descent and extended Kalman filter based self-tuning interval type-2 fuzzy PID controllers. *IEEE International Conference on Fuzzy Systems*.

[63] **Sakalli, A., Beke, A., and Kumbasar, T.** (2018). Analyzing the control surfaces of type-1 and interval type-2 FLCs through an experimental study. *IEEE International Conference on Fuzzy Systems*.

[64] **Beke, A., & Kumbasar, T.** (2018, July). Single Vs. Double Input Interval Type-2 Fuzzy PID Controllers: Which One is Better?. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 1-7.

[65] **Beke, A., & Kumbasar, T.** (2017, July). Game of spheros: A real-world pursuit-evasion game with type-2 fuzzy logic. In *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on Fuzzy Systems*, 1-6.

[66] **Mendel, J. M.** (2010) Comments on "α-plane representation for type-2 fuzzy sets: theory and applications". *IEEE Transactions on Fuzzy Systems*, *18*, 229-230.

[67] **Zhai, D. and Mendel, J. M.** (2012). Comment on "toward general type-2 Fuzzy logic systems based on zSlices". *IEEE Transactions on Fuzzy Systems*, *20*, 996-997.

[68] **Mendel, J. M.** (2014). General type-2 fuzzy logic systems made simple: A tutorial. *IEEE Transactions on Fuzzy Systems*, *22*, 1162-1182.

[69] **Sakalli, A., Kumbasar, T., & Mendel, J. M.** (2020). Towards systematic design of general type-2 fuzzy logic controllers: analysis, interpretation, and tuning. *IEEE Transactions on Fuzzy Systems*, *29*, 226-239.

[70] **Horikawa, S., Furahashi, T., & Tokumaru, H.** (1992). A real-time learning algorithm using neural fuzzy networks with back-propagation algorithm. *IEEE Transactions Neural Networks*, *3*, 801-806.

[71] **Palit, A. K., Doeding, G., Anheier, W., & Popovic, D.** (2002, May). Backpropagation based training algorithm for Takagi-Sugeno type MIMO neuro-fuzzy network to forecast electrical load time series. In *2002 IEEE World Congress on Computational Intelligence. 2002 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE'02. Proceedings (Cat. No. 02CH37291)*, 86-91.

[72] **W. Wu, W., Li, L., Yang, J., & Liu, Y.** (2010). A modified gradient-based neuro-fuzzy learning algorithm and its convergence. *Information Sciences*, *180*, 1630-1642.

[73] **Qiao, J., Li, W., Zeng, X. J., & Han, H.** (2016). Identification of fuzzy neural networks by forward recursive input-output clustering and accurate similarity analysis. *Applied soft computing*, *49*, 524-543.

[74] **Shi, Y., & Mizumoto, M.** (2000). Some considerations on conventional neuro-fuzzy learning algorithms by gradient descent method. *Fuzzy sets and systems*, *112*, 51-63.

[75] **Han, M., Xi, J., Xu, S., & Yin, F. L.** (2004). Prediction of chaotic time series based on the recurrent predictor neural network. *IEEE transactions on signal processing*, *52*, 3409-3416.

[76] **Juang, C. F., & Lin, C. T.** (1998). An online self-constructing neural fuzzy inference network and its applications. *IEEE transactions on Fuzzy Systems*, *6*, 12-32.

[77] **Lee, S. J., & Ouyang, C. S.** (2003). A neuro-fuzzy system modeling with self-constructing rule generationand hybrid SVD-based learning. *IEEE Transactions on Fuzzy Systems*, *11*, 341-353.

[78] **Silva, A. M., Caminhas, W., Lemos, A., & Gomide, F.** (2014). A fast learning algorithm for evolving neo-fuzzy neuron. *Applied Soft Computing*, *14*, 194-209.

[79] **Zhang, J., Chung, H. S. H., & Hu, B. J.** (2004, June). Adaptive probabilities of crossover and mutation in genetic algorithms based on clustering technique. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, 2280-2287.

[80] **Kennedy, J., & Eberhart, R.** (1995, November). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, 1942-1948.

[81] **Lučić, P., & Teodorović, D.** (2003). Computing with bees: attacking complex transportation engineering problems. *International Journal on Artificial Intelligence Tools*, *12*, 375-394.

[82] **Dorigo, M., Birattari, M., & Stutzle, T.** (2006). Ant colony optimization. *IEEE computational intelligence magazine*, *1*, 28-39.

[83] **Das, S., & Suganthan, P. N.** (2010). Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, *15*, 4-31.

[84] **Zhang, W. B., & Ji, H. B.** (2013). Fuzzy extreme learning machine for classification. *Electronics Letters*, *49*, 448-450.

[85] **Sun, Z. L., Au, K. F., & Choi, T. M.** (2007). A neuro-fuzzy inference system through integration of fuzzy logic and extreme learning machines. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *37*, 1321-1331.

[86] **Shihabudheen, K. V., & Pillai, G. N.** (2017). Regularized extreme learning adaptive neuro-fuzzy algorithm for regression and classification. *Knowledge-Based Systems*, *127*, 100-113.

[87] **Lin, C. T., Yeh, C. M., Liang, S. F., Chung, J. F., & Kumar, N.** (2006). Support-vector-based fuzzy neural network for pattern classification. *IEEE Transactions on Fuzzy Systems*, *14*, 31-41.

[88] **Xiong, S. W., Liu, H. B., & Niu, X. X.** (2005, August). Fuzzy support vector machines based on FCM clustering. In *2005 International Conference on Machine Learning and Cybernetics*, 2608-2613.

[89] **Hang, J., Zhang, J., & Cheng, M.** (2016). Application of multi-class fuzzy support vector machine classifier for fault diagnosis of wind turbine. *Fuzzy Sets and Systems*, *297*, 128-140.

[90] **Riid, A., & Rüstern, E.** (2014). Adaptability, interpretability and rule weights in fuzzy rule-based systems. *Information Sciences*, *257*, 301-312.

[91] **Castillo, O., Amador-Angulo, L., Castro, J. R., & Garcia-Valdez, M.** (2016). A comparative study of type-1 fuzzy logic systems, interval type-2 fuzzy logic systems and generalized type-2 fuzzy logic systems in control problems. *Information Sciences*, *354*, 257-274.

[92] **Beke, A., & Kumbasar, T.** (2018). Type-2 fuzzy logic-based linguistic pursuing strategy design and its deployment to a real-world pursuit evasion game. *IEEE Transactions on Cybernetics*, *50*, 211-221.

[93] **Runkler, T., Coupland, S., & John, R.** (2017). Interval type-2 fuzzy decision making. *International Journal of Approximate Reasoning*, *80*, 217-224.

[94] **Huarng, K., & Yu, H. K.** (2005). A type 2 fuzzy time series model for stock index forecasting. *Physica A: Statistical Mechanics and its Applications*, *353*, 445-462.

[95] **Kingma, D. P., & Ba, J.** (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[96] **Dahl, G. E., Sainath, T. N., & Hinton, G. E.** (2013, May). Improving deep neural networks for LVCSR using rectified linear units and dropout. In *2013 IEEE international conference on acoustics, speech and signal processing*, 8609-8613.

[97] **Maas, A. L., Hannun, A. Y., & Ng, A. Y.** (2013, June). Rectifier nonlinearities improve neural network acoustic models. *In Proceeding of the International Conference on Machine Learning*, 3.

[98] **He, K., Zhang, X., Ren, S., & Sun, J.** (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026-1034.

[99] **Clevert, D. A., Unterthiner, T., & Hochreiter, S.** (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

[100] **Trottier, L., Giguere, P., & Chaib-Draa, B.** (2017, December). Parametric exponential linear unit for deep convolutional neural networks. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, 207-214.

[101] **Khosravi, A., Nahavandi, S., Creighton, D., & Naghavizadeh, R.** (2012, June). Prediction interval construction using interval type-2 fuzzy logic systems. In *2012 IEEE International Conference on Fuzzy Systems*, 1-7.

[102] **Cartagena, O., Parra, S., Muñoz-Carpintero, D., Marín, L. G., & Sáez, D.** (2021). Review on fuzzy and neural prediction interval modelling for nonlinear dynamical systems. *IEEE Access*, *9*, 23357-23384.

[103] **Chung, Y., Neiswanger, W., Char, I., & Schneider, J.** (2021). Beyond pinball loss: Quantile methods for calibrated uncertainty quantification. Advances in Neural Information Processing Systems, *34*, 10971-10984.

[104] **Zheng, Y., Xu, Z., & Wang, X.** (2021). The fusion of deep learning and fuzzy systems: A state-of-the-art survey. *IEEE Transactions on Fuzzy Systems*, *30*, 2783-2799.

[105] **Dombi, J., & Hussain, A.** (2022). Data-Driven Interval Type-2 Fuzzy Inference System Based on the Interval Type-2 Distending Function. *IEEE Transactions on Fuzzy Systems*.

[106] **Y. LeCun**, (1998). The MNIST database of handwritten digits. Retrieved March 10, 2019. Available from http://yann.lecun.com/exdb/mnist/.

[107] **Quickdraw**. (2019). Quickdraw Pictionary Dataset. Retrieved March 10, 2019. Available from https://quickdraw.withgoogle.com/data.

[108] **Rodrigues, F., & Pereira, F. C.** (2020). Beyond expectation: Deep joint mean and quantile regression for spatiotemporal problems. *IEEE transactions on neural networks and learning systems*, *31*, 5377-5389.

[109] **Xu, Q., Deng, K., Jiang, C., Sun, F., & Huang, X.** (2017). Composite quantile regression neural network with applications. *Expert Systems with Applications*, *76*, 129-139.

[110] **Chen, C., Li, Y., Yan, C., Guo, J., & Liu, G.** (2017). Least absolute deviation-based robust support vector regression. *Knowledge-Based Systems*, *131*, 183-194.

[111] **Ertuğrul, Ö. F.** (2018). A novel type of activation function in artificial neural networks: Trained activation function. *Neural Networks*, *99*, 148-157.

[112] **Ertuğrul, Ö. F.** (2020). A novel randomized machine learning approach: Reservoir computing extreme learning machine. *Applied Soft Computing*, *94*, 106433.

[113] **Gámez, J. C., García, D., González, A., & Pérez, R.** (2019). An approximation to solve regression problems with a genetic fuzzy rule ordinal algorithm. *Applied Soft Computing*, *78*, 13-28.

[114] **Singla, M., Ghosh, D., Shukla, K. K., & Pedrycz, W.** (2020). Robust twin support vector regression based on rescaled hinge loss. *Pattern Recognition*, *105*, 107395.

[115] **Kang, S.** (2021). K-nearest neighbor learning with graph neural networks. *Mathematics*, *9*, 830.

[116] **Wang, H., Wang, Y., & Hu, Q.** (2017). Self-adaptive robust nonlinear regression for unknown noise via mixture of Gaussians. *Neurocomputing*, *235*, 274-286.

[117] **Wu, D., Lin, C. T., & Huang, J.** (2019). Active learning for regression using greedy sampling. *Information Sciences*, *474*, 90-105.

[118] **Hwang, S. H., & Whang, S. E.** (2021). RegMix: Data Mixing Augmentation for Regression. *arXiv preprint arXiv:2106.03374*.

[119] **Wang, X., Chen, H., Cai, W., Shen, D., & Huang, H.** (2017). Regularized modal regression with applications in cognitive impairment prediction. *Advances in neural information processing systems*, *30*.

[120] **Balasundaram, S., & Prasad, S. C.** (2020). Robust twin support vector regression based on Huber loss function. *Neural Computing and Applications*, *32*, 11285-11309.

[121] **Thiagarajan, J. J., Venkatesh, B., Sattigeri, P., & Bremer, P. T.** (2020, April). Building calibrated deep models via uncertainty matching with auxiliary interval predictors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*, 6005-6012.

[122] **Partaourides, H., & Chatzis, S.** (2018). Deep learning with t-exponential Bayesian kitchen sinks. *Expert systems with applications*, *98*, 84-92.

[123] **Liu, C., Tang, L., & Liu, J.** (2019). Least squares support vector machine with self-organizing multiple kernel learning and sparsity. *Neurocomputing*, *331*, 493-504.

[124] **Quan, H., Srinivasan, D., & Khosravi, A.** (2013). Short-term load and wind power forecasting using neural network-based prediction intervals. *IEEE transactions on neural networks and learning systems*, *25*, 303-315.

CURRICULUM VITAE

Name Surname **:** Aykut BEKE

EDUCATION **:**

- **B.Sc.** **:** 2015, Istanbul Technical University, Faculty of Electrical and Electronics, Control and Automation Engineering Department
- **M.Sc.** **:** 2018, Istanbul Technical University, Faculty of Electrical and Electronics, Control and Automation Engineering Department

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2022-…: Control Systems Design Engineer, ASELSAN
- 2018-2021: Research Assistant, ITU, Control and Automation Engineering Department
- 2015-2017: Software Engineer, Getron Bilişim Hizmetleri
- 2015: Best Undergraduate Thesis Award by Elektrik Tesisat Mühendisleri Dernegi (ETMD).

PUBLICATIONS/PRESENTATIONS ON THE THESIS:

- **Beke, A.,** & Kumbasar, T. (2022). More than Accuracy: A Composite Learning Framework for Interval Type-2 Fuzzy Logic Systems. *IEEE transactions on Fuzzy Systems*.
- **Beke, A.,** & Kumbasar, T. (2019). Learning with type-2 fuzzy activation functions to improve the performance of deep neural networks. *Engineering Applications of Artificial Intelligence*, *85*, 372-384.
- Avcı, B.**, Beke, A.,** & Kumbasar, T. (2023). Towards Reliable Uncertainty Quantification and High Precision with General Type-2 Fuzzy Systems. In *2023 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (accepted).
- **Beke, A.,** & Kumbasar, T. (2021, July). Capturing Uncertainty with Interval Fuzzy Logic Systems through Composite Deep Learning. *In 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (pp. 1-6). IEEE.
- **Beke, A., &** Kumbasar, T. (2019). Interval Type-2 Fuzzy Systems as Deep Neural Network Activation Functions. *11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019).*

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS:

- **Beke, A.,** & Kumbasar, T. (2022). More than Accuracy: A Composite Learning Framework for Interval Type-2 Fuzzy Logic Systems. *IEEE transactions on Fuzzy Systems*.
- Tuna, T., **Beke, A.,** & Kumbasar, T. (2022). Deep learning frameworks to learn prediction and simulation focused control system models. *Applied Intelligence*, 52(1), 662-679.
- Erdem, D., **Beke, A.,** & Kumbasar, T. (2020). A deep learning-based pipeline for teaching control theory: transforming feedback control systems on whiteboard into MATLAB. IEEE Access, 8, 84631-84641.
- Etlik, U. B., Korkmaz, B., Beke, A., & Kumbasar, T. (2019). A fuzzy logic-based autonomous car control system for the JavaScript Racer game. *Transactions of the Institute of Measurement and Control*, 0142331219889526.
- **Beke, A.,** & Kumbasar, T. (2019). Learning with type-2 fuzzy activation functions to improve the performance of deep neural networks. *Engineering Applications of Artificial Intelligence*, *85*, 372-384.
- **Beke, A.,** & Kumbasar, T. (2018). Type-2 fuzzy logic-based linguistic pursuing strategy design and its deployment to a real-world pursuit evasion game. *IEEE transactions on cybernetics*, *50*(1), 211-221.
- Avcı, B.**, Beke, A.,** & Kumbasar, T. (2023). Towards Reliable Uncertainty Quantification and High Precision with General Type-2 Fuzzy Systems. In *2023 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (accepted).
- Candan, F., **Beke, A.,** Shen, C., Mihaylova, L. (2022, October). An Interacting Multiple Model Correntropy Kalman Filter Approach for Unmanned Aerial Vehicle Localisation. In *2022 Innovations in Intelligent Systems and Applications (INISTA)* (pp. 1-6). IEEE.
- **Beke, A.,** & Kumbasar, T. (2021, July). Capturing Uncertainty with Interval Fuzzy Logic Systems through Composite Deep Learning. *In 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (pp. 1-6). IEEE.
- Kaplan, M.R., Eraslan, A., **Beke, A., &** Kumbasar, T. (2019). Altitude and position control of parrot mambo minidrone with PID and fuzzy PID controllers. *11th International Conference on Electrical and Electronics Engineering (ELECO).*
- **Beke, A., &** Kumbasar, T. (2019). Interval Type-2 Fuzzy Systems as Deep Neural Network Activation Functions. *11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019).*
- Ovur, S.E., Candan, F., **Beke, A., &** Kumbasar, T. (2018, October). YAFT: A Fuzzy Logic based Real Time Two-Wheeled Inverted Pendulum Robot. In *2018 IEEE International Conference on Control Engineering & Information Technology (CEIT)*.
- Korkmaz, B., Etlik, U., Beke, A., & Kumbasar, T. (2018, October). Fuzzy Logic Based Self-Driving Racing Car Control System. In *2018 IEEE International Conference on Control Engineering & Information Technology (CEIT)*.
- **Beke, A.,** & Kumbasar, T. (2018, July). Single Vs. Double Input Interval Type-2 Fuzzy PID Controllers: Which One is Better?. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (pp. 1-7). IEEE.

- Sakalli, A., **Beke, A.,** & Kumbasar, T. (2018, July). Analyzing the Control Surfaces of Type-1 and Interval Type-2 FLCs through an Experimental Study. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* (pp. 1-6). IEEE.
- Candan, F., **Beke, A.,** & Kumbasar, T. (2018, July). Design and Deployment of Fuzzy PID Controllers to the nano quadcopter Crazyflie 2.0. In *2018 Innovations in Intelligent Systems and Applications (INISTA)* (pp. 1-6). IEEE.
- **Beke, A.,** & Kumbasar, T. (2017, July). Game of spheros: A real-world pursuit-evasion game with type-2 fuzzy logic. In *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on* (pp. 1-6). IEEE.
- **Beke, A.,** Yuceler, A. A., & Kumbasar, T. (2017, September). A rule based fuzzy gesture recognition system to interact with Sphero 2.0 using a smart phone. In *Artificial Intelligence and Data Processing Symposium (IDAP), 2017 International* (pp. 1-4). IEEE.
- Sakalli, A., **Beke, A.,** & Kumbasar, T. (2016, July). Gradient descent and extended kalman filter based self-tuning interval type-2 fuzzy PID controllers. In *Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on* (pp. 1592-1598). IEEE.
- Savran, A. I., **Beke, A.**, Kumbasar, T., & Yesil, E. (2015, September). An IMC based fuzzy self-tuning mechanism for fuzzy PID controllers. In *Innovations in Intelligent SysTems and Applications (INISTA), 2015 International Symposium on* (pp. 1-7). IEEE.