mathematics

MDPI

Article

Decentralized Nonstationary Fuzzy Neural Network with Meta-Learning-Net

Zhen Zhang †, Meiling Yu † and Hui Jia *

College of Artificial Intelligence, Nankai University, Tianjin 300381, China; zzxhk@mail.nankai.edu.cn (Z.Z.); 1120230250@mail.nankai.edu.cn (M.Y.)

* Correspondence: 2120210368@mail.nankai.edu.cn

† These authors contributed equally to this work.

Abstract: The nonstationary fuzzy neural network (NFNN) has proven to be an effective and interpretable tool in machine learning, capable of addressing uncertainty problems similarly to type-2 fuzzy neural networks, while offering reduced computational complexity. However, the update of disturbance parameters in an NFNN is restricted due to the necessity of maintaining a regular membership function, which limits its learning capability. To address this limitation, we propose a Decentralized NFNN (DNFNN) that overcomes the obstacles in the backward update process and enhances the efficiency of large-scale optimization. Additionally, we demonstrate improved computational efficiency and establish the linear convergence of the proposed decentralized algorithm. By integrating a meta-learning network, we further enhance the output strategy of the NFNN, enabling it to adaptively determine the contribution of individual sub-networks. Experimental results on various UCI datasets, spanning multiple domains and exhibiting diverse dimensions and sizes, show that the DNFNN outperforms existing methods in terms of classification accuracy, robustness and practicality.

Keywords: fuzzy system; nonstationary fuzzy neural network; decentralized optimization; meta-learning; linear convergence

MSC: 68T07

check for updates

Academic Editor: Marjan Mernik

Received: 16 January 2025
Revised: 30 January 2025
Accepted: 6 February 2025
Published: 7 February 2025

Citation: Zhang, Z.; Yu, M.; Jia, H. Decentralized Nonstationary Fuzzy Neural Network with Meta-Learning-Net. *Mathematics* **2025**, *13*, 552. https://doi.org/10.3390/math13040552

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The combination of fuzzy inference systems and neural networks (FNNs) effectively integrates the interpretability of fuzzy rules with the learning capability of neural networks, making it widely applicable across various domains such as nonlinear systems [1,2], fuzzy control [3], optimization technologies [4], and classification [5], among others. Fuzzy neural networks can be classified into type-1 and type-2 FNNs based on the type of fuzzy sets they utilize. While type-1 FNNs offer a more compact model structure and lower training costs, their performance is less effective compared to type-2 FNNs when dealing with complex uncertainty problems due to the crisp membership grades [6–9]. The introduction of the interval type-2 FNN [10] sparked a surge in applications of this type [11–13]. Additionally, research [14] extended type-2 fuzzy systems by integrating the technique for the order of preference by similarity to the ideal solution. Motivated by simplicity and efficiency, several studies have sought to enhance the performance of type-2 fuzzy systems by utilizing interval type-2 fuzzy sets, thus simplifying type-2 neuro-fuzzy systems while maintaining the interval membership grades [14–16]. Despite the enhanced capability of type-2 fuzzy

sets in representing uncertainty, they present challenges in interpretability and incur higher computational costs.

To enhance the flexibility and applicability of fuzzy neural networks, Garibaldi et al. introduced nondeterministic fuzzy reasoning [17,18] and nonstationary fuzzy sets [19]. A nonstationary fuzzy set is composed of a series of related type-1 fuzzy sets, where the interrelationships among these sets are characterized by gradual changes in the membership functions over time. Notably, the aggregation of instances from a nonstationary fuzzy set often resembles the footprint of uncertainty (FOU) of a type-2 fuzzy set when visualized [19]. In the research [20], a comparative study of type-1, interval type-2, nonstationary, and dual-surface type-2 fuzzy controllers [16] was conducted on a simulated autonomous sailing problem, demonstrating the superior performance of the nonstationary mechanism. Furthermore, the study in [21] proposed an adaptive nonstationary fuzzy neural network capable of self-learning. However, the back-forward fuzzy networks [21] only update the connection weights and centers without addressing the widths of the fuzzy sets.

Following the backpropagation update in a nonstationary fuzzy neural network, the parameters may undergo significant perturbations, causing the fuzzification function to no longer approximate within an interval. To address this issue, we propose the incorporation of decentralized learning. Compared with centralized optimization, decentralized learning allows for parallel training and enhances the security and privacy of sensitive data [22–24]. Decentralized optimization has found widespread application in engineering and scientific fields [25–27], leveraging network communication to decompose complex, large-scale optimization tasks into smaller, manageable components, ultimately producing a consistent outcome. The study in [28] introduced a discrete-time subgradient algorithm to solve unconstrained distributed convex optimization problems, marking a significant advancement in decentralized learning. When the objective function is smooth, the algorithm simplifies to decentralized gradient descent. However, a limitation of these algorithms is the potential failure to converge when using a fixed step size [29]. To mitigate this issue, a gradient tracking-based distributed optimization algorithm with a fixed step size was proposed in [30]. Due to the simplicity and scalability of gradient tracking (GT), an increasing number of GT-based decentralized optimization algorithms have been developed to address various practical challenges [31,32]. In this paper, we propose a GT-based algorithm for nonstationary fuzzy neural networks, which demonstrates relatively faster iteration efficiency.

To enhance the learning capability of neural networks, some studies have focused on meta-learning [33–36] or reweighting methods [37–39], and the recent literature suggests the use of more precise analysis management in image classification tasks to improve model performance and reliability, as highlighted by [40]. The fundamental idea behind reweighting is to pre-collect a small set of unbiased meta-data with clean labels to simulate the underlying meta-knowledge and subsequently design a meta-learner to determine the weights of different samples. Typical methods include self-paced learning [41], iterative reweighting [42], and various other approaches [43]. These reweighting strategies improve the flexibility of sample weight evaluation and expand the feasible domains beyond the initial sample weight schemes. However, such methods still rely on certain assumptions for model construction, often requiring hyperparameters to be manually set or tuned via cross-validation. This adds complexity to their implementation and reduces their performance stability when applied to real-world problems. To address these challenges, this paper proposes a novel meta-learning strategy utilizing a multilayer perceptron (MLP) network as a meta-learner, designed to determine the influence proportions of each sub-network.

In this paper, we propose a decentralized nonstationary fuzzy neural network (DNFNN) architecture, which retains the fuzzy capabilities of a type-2 FNN while reducing

computational complexity through the use of multiple type-1 FNNs. The backpropagation process in the DNFNN follows a decentralized strategy, with the meta-learner assigning appropriate weights to each sub-network. The main contributions of this work are summarized as follows:

(i) To address the limitation of not being able to update the centers of the membership functions, we propose a decentralized update policy for the nonstationary fuzzy neural network (NFNN). During the learning process, all trained parameters converge to a consistent solution that satisfies the definition of nonstationary sets. The integration of nonstationary fuzzy sets and decentralized learning significantly enhances the reliability and interpretability of the hybrid model.

(ii) When training a DNFNN, it is necessary to harmonize the outputs of each sub-network. Current approaches typically rely on voting mechanisms or averaging techniques. However, different sub-networks require reweighting due to the varying contributions of samples, rather than assuming an equal contribution across all sub-networks. To address this issue, we propose a meta-learner for the DNFNN, which is capable of adaptively learning the appropriate propagation for each sub-FNN. The approach requires a set of unbiased meta-data to update a multilayer perceptron (MLP) with one hidden layer. Through the MLP, the hybrid model can automatically learn the weights, eliminating the need for manual pre-specification.

(iii) The convergence theorem of the proposed decentralized algorithm provides theoretical support for the implementation of the DNFNN. Also, a discussion is presented to illustrate the effectiveness of our work in comparison with other update policies and network structures.

(iv) Various experiments demonstrate that the trained parameters (including the centers and widths of the membership functions) converge to a consistent and optimal solution. Moreover, the simulation results highlight the effectiveness of our method, particularly in addressing class imbalance and noisy label classification problems.

The rest of the paper is organized as follows: In Section 2, we briefly describe some notations and preliminaries. Section 3 gives the update strategy for the DNFNN. In Section 4, we introduce a meta-learning-net for obtaining the degree of contribution of each sub-FNN. The convergence theorem is shown in Section 5. Section 6 presents the supporting numerical experiments. Some brief conclusions and discussions are given in Section 7.

2. Notations and Preliminaries

In this paper, vectors default to columns. Let $\|\cdot\|$ be the standard two-norm and $\mathcal{T} = \{1, 2, \ldots, T\}$ be the sets of sub-networks. The structure of a nonstationary fuzzy neural network is composed of a series of zero-order type-1 TSK fuzzy neural networks [18]. Each input data point is denoted by $\mathbf{x} = (x_1, x_2, \cdots, x_m)^T \in \mathbb{R}^m$. With the clustering method, we can create n rules, and rule $\mathbf{R}_i$ $(i = 1, 2, \cdots, n)$ can be written as:

$\mathbf{R}_i$: **IF** x_1 **is** A_{1i} **and** x_2 **is** A_{2i} **and** $\cdots$ **and** x_m **is** A_{mi},
THEN y **is** y_i,

where y_i is the real output, and A_{li} is a fuzzy variable which can be represented by a Gaussian membership function:

$$A_{li}(x_l) = \exp(-(x_l - a_{li})^2 d_{li}^2), \tag{1}$$

where a_{li} and d_{li} are the center and the inverse of the width of the Gaussian function A_{li}, $l = 1, \ldots, m$, respectively. This results in the construction of a fundamental zero-order type-1 TSK FNN, structured as a four-layer network:

Input layer: Each node in this layer represents a single input variable, which processes incoming data and transmits them directly to the next layer, the membership layer, without being influenced by connection weights.

The nodes in the membership layer perform fuzzification using Gaussian membership functions, which are expressed in a nonstationary form:

$$A_{li}(x_l, t) = \exp(-(x_l - c_{li}(t))^2 d_{li}^2), \tag{2}$$

where

$$c_{li}(t) = a_{li} + \alpha \sin(\theta(t-1)), \tag{3}$$

$l = 1, \ldots, m, i = 1, \ldots, n, t \in \mathcal{T}$. θ and α serve as the hyperparameters which introduce a perturbation to the center of the membership function to model expert variability.

Rule layer: within this layer, the ith node calculates the firing strength associated with the ith rule by applying the product T-norm:

$$h_i(\mathbf{x}, t) = A_{1i}(x_1, t) A_{2i}(x_2, t) \cdots A_{mi}(x_m, t) = \prod_{l=1}^{m} A_{li}(x_l, t). \tag{4}$$

Output layer: A fully connected network (combined with a dropout layer) links the rule layer to the output layer through weights $\mathbf{v} = (v_1, v_2, \ldots, v_n)$. The final output can be obtained by linearly combining the firing strength:

$$y = \sum_{i=1}^{n} h_i v_i. \tag{5}$$

In previous NFNN works [21], centralized SGD was employed to train the limited parameters, including only $\{v_i\}, \{d_{li}\}$, but without updating the center parameters $\{c_{li}\}, i = 1, \ldots, n, l = 1, \ldots, m$. This occurs because the variations are concentrated in the centers (2), rendering the membership functions unable to preserve nonstationary fuzzy sets. To address these challenges, we propose a decentralized gradient-based approach for training model parameters. This decentralized optimization method trains the entire network collaboratively while ensuring consistent convergence across individual networks. Even when the perturbed parameters are updated, the resulting membership functions resemble the uncertainty footprint of certain type-2 fuzzy sets. This demonstrates that the DNFNN can effectively handle uncertainty issues inherent in type-2 FNNs while maintaining computational efficiency comparable to a limited number of type-1 FNNs.

After T outputs are obtained from the T sub-networks, the final results can be computed via a voting mechanism or weighted averaging [21]. However, such final output cannot reflect the importance of different sub-networks because of the difference in input samples. Inspired by some meta-learning developments, we introduce meta-learning and modify the structure of the NFNN so that different sub-networks can adaptively learn their own importance levels in this paper.

3. Decentralized Update Algorithm

In this section, we introduce a decentralized algorithm, the Decentralized Probability Gradient Estimator (DPGE), for updating parameters of DNFNN. By considering the general loss function minimization problem

$$\min_{w \in \mathbb{R}^N} L(w) = \frac{1}{T} \sum_{t=1}^{T} l_t(w), \tag{6}$$

where $l_t(w)$ is the cost of tth sub-FNN, and $w \in \mathbb{R}^N$ is the updated parameter which consists of the center, the inverse of the width, and the fully connected network weights. The global cost $L(w)$ is defined as the average of local objectives.

Before formally introducing the algorithm, it is necessary to clarify the difference between centralized optimization and decentralized optimization. The traditional update strategy for NFNNs is centralized optimization, which has a leader node to collect losses from all sub-networks. In decentralized optimization, sub-networks are connected by a graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$, where each quantity $t \in \mathcal{T}$ denotes a sub-network, and each edge $e_{ij} \in \mathcal{E}$ denotes the communication link associated with a positive weight $M_{ij} > 0$. In this paper, we stipulate that the graph $\mathcal{G}$ is undirected and connected. The advantage of such design is that the sub-networks are no longer isolated but rather mutually influence each other. This implies the reception of more information from training samples, enabling the optimal weights of each sub-network to converge towards a common optimal solution.

As shown in Algorithm 1, we represent the aggregation iteration of the DPGE algorithm where parameters $\mathbf{w}, \mathbf{y} \in \mathbb{R}^{T\times N}$ are the matrix forms spanned by T sub-networks, and $\nabla L(\mathbf{w})$ is defined by

$$\nabla L(\mathbf{w}) := [\nabla f_1(w_1)^\top, \nabla f_2(w_2)^\top, \cdots, \nabla f_T(w_T)^\top] \in \mathbb{R}^{T\times N}.$$

The decentralized gradient descent is executed on step 2 in each iteration. A gradient estimator $\mathbf{y}^{k+1}$ is defined on step 3, which indicates that the DPGE utilizes the vanilla decentralized SGD update with probability p_t and uses the previous gradient $\mathbf{y}_k$ with a momentum item with probability $1 - p_t$. In particular, if $p_t = 1$, the algorithm reduces to the vanilla decentralized gradient descent. More details can be found in the convergence results of Section 5.

Algorithm 1 Decentralized Probability Gradient Estimator algorithm (DPGE)

Input: Number of sub-networks T; learning rate α; undirected graph weight matrix $\mathbf{M} = [M_{ij}] \in \mathbb{R}^{T\times T} > 0$; each sub-network t initializes with any arbitrary updated parameters $w_t^0 \in \mathbb{R}^N$ and $y_t^0 = \nabla l_t(w_{t,0})$; probability parameter $p_t \in (0, 1]$.

1: **for** $k = 0, 1, 2, \cdots$ **do**

2: $\mathbf{w}^{k+1} = \mathbf{M}\mathbf{w}^k - \alpha\mathbf{y}^k$;

3:
$$\mathbf{y}^{k+1} = \begin{cases} \nabla L\left(\mathbf{w}^{k+1}\right) & \text{with probability } p_t \\ \mathbf{M}\mathbf{y}^k + \nabla L\left(\mathbf{w}^{k+1}\right) - \nabla L\left(\mathbf{w}^k\right) & \text{with probability } 1 - p_t \end{cases}$$

4: **end for**

Output: The trained $\mathbf{w}*$ ($\mathbf{w}*$ is combined with T $w*$).

It is necessary to note that interactions between sub-networks can occur in real time, enabling the full reception of training sample information, which is particularly beneficial for training on large-scale datasets. Traditional NFNNs rely on independent training and updates, with all communication and coordination centralized. In contrast, with decentralization, sub-networks can update synchronously in real time, supporting parallel implementation. Distributed updating NFNNs eliminate the need for a central node to collect all sub-network parameters and errors, offering significant advantages for data privacy and security.

Compared to gradient descent, the DPGE updates require the computation of fewer gradients, leading to lower computational costs. Distributed gradient descent methods have also found broad applications in fields such as automation and communication. However, distributed SGD cannot theoretically guarantee convergence with a constant step size; as iterations progress, the convergence rate of distributed SGD may slow down.

In contrast, the distributed algorithm proposed in this paper ensures convergence with a constant step size and demonstrates superior numerical performance.

4. Decentralized Nonstationary Fuzzy Neural Network with Meta-Learning-Net

In this section, we introduce the structure of the proposed DNFNN with meta-learning-net, which has the potential to be applied in the area of medical diagnostics, financial forecasting, and drone swarm. By considering a classification problem with the training set $\{x_t, y_t\}_{t=1}^{T}$, where x_t, y_t represent the input samples and label vectors of the tth sub-FNN, respectively. Let $f(x_t, w)$ be the actual output of corresponding training data, where $w \in \mathbb{R}^N$ represents the model parameters to be trained. By minimizing the loss $\frac{1}{T}\sum_{t=1}^{T} l(y_t, f(x_t, w))$, called $L_t^{train}(w) = l(y_t, f(x_t, w))$ below, we can obtain the optimal trained parameter w^*.

As shown in Figure 1, the structure of the DNFNN can be divided into two nested updating processes: regular learning and meta-learning which rely on different input samples, where the parameters with $*$ are the optimal parameters obtained after training. Firstly, data samples are divided into two parts: training datasets $\{x_t^{train}, y_t^{train}\}_{t=1}^{T}$ and meta-datasets $\{x_t^{meta}, y_t^{meta}\}_{t=1}^{T}$. Generally, classical meta-learning methods typically assume balanced and unbiased data. However, we would like to clarify that in our approach, the meta-data requirements are not as stringent as those complex and deep meta-learning frameworks. The model we propose utilizes a relatively simple architecture, a single hidden-layer neural network for meta-learning. Due to this simplicity, the model is capable of functioning effectively with smaller and less strictly balanced meta-datasets. Naturally, if a highly balanced and unbiased dataset was used, it is evident that better model performance could be achieved. However, this would necessitate the additional workload of data preprocessing. Therefore, the meta-data available from sources like the UCI repository, which usually plays the role of the training set, suffice for training the meta-learner to enable fine-tuning of the network [36] (when faced with highly imbalanced or biased meta-data, data augmentation and generation techniques can be employed, such as SMOTE (Synthetic Minority Oversampling Technique), Generative Adversarial Networks (GANs), or other data generation strategies). Next, we provide a detailed introduction of the two updating processes.

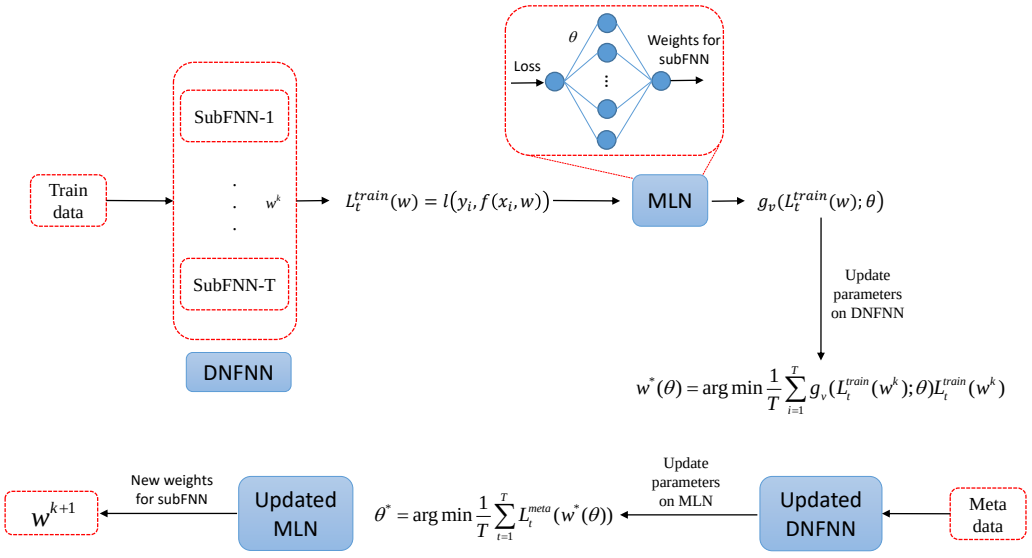


Figure 1. Structure of the decentralized nonstationary fuzzy neural network.

Regular learning: The training samples are first divided into T parts, followed by the forward process of the DNFNN. The training samples are input into T sub-networks, at which point we obtain the error function $L_t^{train}(w)$ for each sub-network. Next, we introduce an MLP (referred to as MLN) to generate the weights for each sub-network whose input is loss $L_t^{train}(w)$, with an intermediate layer of 50 nodes and the ReLU activation function. In our current implementation, we use a Grid Search to determine the hyperparameters, such as the learning rate, regularization parameters, and the number of iterations. The connection weights of the MLN are denoted as ζ, and the output is the error function weight $g_m(L_t^{train}(w);\zeta)$, $t = 1, 2, \cdots$. At this point, the entire network's error function becomes $\frac{1}{T}\sum_{t=1}^{T} g_m(L_t^{train}(w);\zeta)L_t^{train}(w)$, and we can update the weights w as follows:

$$w^*(\theta) = \arg\min_{w} \frac{1}{T}\sum_{t=1}^{T} g_m(L_t^{train}(w);\zeta)L_t^{train}(w). \tag{7}$$

Meta-learning: Through the regular learning process described above, the DNFNN updates its weight parameters. Subsequently, the meta-dataset is input into the updated DNFNN, producing a new error $L_t^{meta}(w^*(\zeta))$. At this stage, the parameter ζ can be refined by minimizing $L_t^{meta}(w^*(\zeta))$.

$$\zeta^* = \arg\min_{\zeta} \frac{1}{T}\sum_{t=1}^{T} L_t^{meta}(w^*(\zeta)), \tag{8}$$

where $L_t^{meta}(w^*(\zeta)) = l(y_i^{meta}, f(x_i^{meta}, w))$. After updating the MLN, the new error function weight of each sub-network $g_m(L_t^{train}(w);\zeta^*)$ can be obtained and used to update the parameters of the DNFNN, allowing the process to iterate continuously.

Equation (7) can be interpreted as representing the model's self-learning mechanism, wherein the weight function (i.e., MLN) serves as a guiding framework to enhance the learning process by prioritizing more informative samples. Meanwhile, Equation (8) reflects a self-correction mechanism that utilizes final feedback and incorporates more valuable reference information from the meta-datasets for optimization. For simple and balanced datasets, we can indeed use weighted averaging to simplify the model without sacrificing much accuracy. However, in most cases, the data we collect are often noisy and unbalanced, which highlights the practical applicability of meta-learning approaches.

Then, we would like to emphasize that incorporating meta-learning into a distributed system may introduce some additional computational costs, but our proposed model still demonstrates a significant reduction in computational complexity compared to type-2 FNNs. There are two key reasons for this:

1. **De-fuzzification in type-2 fuzzy systems:** One of the most computationally expensive operations in type-2 fuzzy systems is the "de-fuzzification" process, where fuzzy intervals are converted into a crisp output. This process requires extensive data sorting, which is highly resource-intensive. In contrast, our model does not require such de-fuzzification, as it operates with crisp outputs rather than interval-based outputs. This results in a significant reduction in computational cost.
2. **Sub-network complexity:** Although our model is decentralized and involves multiple sub-networks, the number of sub-networks required is relatively small. In our experiments, we found that even with as few as 30 sub-networks, our model outperformed type-2 FNNs in terms of accuracy and efficiency. Furthermore, the meta-learning process only trains a single hidden-layer MLP (multilayer perceptron), which has a limited number of hidden units, thus minimizing the additional computational burden brought by meta-learning.

5. Convergence Analysis

In this section, we present the convergence theorem of the proposed decentralized algorithm. With the updating parameters w, the backward propagation error function of the DNFNN can be defined as:

$$L(w) = \frac{1}{T}\sum_{t=1}^{T} l_t(w), \tag{9}$$

where T is the number of sub-networks, l_t is the error function computed by the tth sub-FNN. Now, we give some necessary and common assumptions on the aggregate function L.

Assumption 1. *$L(w)$ is μ-strongly convex, and each $l_t(w)$ is $\mathcal{L}$-smooth and may be nonconvex:*

$$L(y) \geq L(x) + \langle \nabla L(x), y - w\rangle + \frac{\mu}{2}\|y - w\|^2, \forall x, y \in \mathbb{R}^N,$$
$$\|\nabla l_t(y) - \nabla l_t(x)\| \leq \mathcal{L}\|y - w\|, \quad i \in \mathcal{V}, \forall x, y \in \mathbb{R}^N.$$

Assumption 1 implies that composite function L is also $\mathcal{L}$-smooth:

$$\|\nabla L(y) - \nabla L(x)\| \leq \mathcal{L}\|y - w\|, \forall x, y \in \mathbb{R}^N.$$

Remark 1. *Such assumptions (sum-of-nonconvex problems) have become very popular in the area of machine learning. For example, we generally minimize the average of m loss functions $g(x) = \frac{1}{m}\sum_{i=1}^{m} g_i(x)$ during some model training. To avoid over-fitting, we usually define $L(w) = g(x) + \frac{\lambda}{2}\|x\|^2$ by adding a quadratic term. By choosing an appropriate λ, $L(w)$ can be strongly convex and is easier to handle than $g(x)$, yet it is not guaranteed that each $f_i(x) = g_i(x) + \frac{\lambda}{2m}\|x\|^2$ is a convex function.*

Denote

$$\overline{w} = \frac{1}{T}\sum_{t=1}^{T} w_t, \quad \overline{y} = \frac{1}{T}\sum_{t=1}^{T}(\nabla l_t(w_t)), \tag{10}$$

and through the definition of $\mathbf{M}$ in Algorithm 1, we can define the following variables by the initialization $y_t^0 = \nabla l_t(w_t^0)$: (If $\mathbf{y}_{k+1} = \nabla L(\mathbf{w}_{k+1})$, Algorithm 1 degenerates into the stochastic gradient descent algorithm whose convergence has been thoroughly discussed in previous works. Therefore, we only need to focus on the case where $\mathbf{y}_{k+1} = \mathbf{M}\mathbf{y}_k + \nabla L(\mathbf{w}_{k+1}) - \nabla L(\mathbf{w}_k)$.)

$$\overline{w}^{k+1} = \overline{w}^k - \alpha\overline{y}^k, \tag{11}$$

$$\overline{y}^{k+1} - \frac{1}{T}\sum_{t=1}^{T}\nabla l_t(w_t^{k+1}) = \overline{y}^k - \frac{1}{T}\sum_{t=1}^{T}\nabla l_t(w_t^k). \tag{12}$$

We first give the convergence theorem to establish the iteration relationship between $\|\Pi\mathbf{w}^{k+1}\|$, $\|\Pi\mathbf{y}^{k+1}\|$, and $\|\Pi\mathbf{w}^k\|$, $\|\Pi\mathbf{y}^k\|$, where $\Pi = \mathbf{I} - \frac{\mathbf{1}\mathbf{1}^\top}{T}$. Then, we hope that the iteration sequences can converge to zero. Moreover, if the sub-networks are connected, it can be concluded that the second largest singular value σ of $\mathbf{M}$ is less than 1, that is, $\sigma = \|\mathbf{M} - \frac{1}{T}\mathbf{1}\mathbf{1}^T\|_2 < 1$. Also, we have the following classical consensus contraction:

$$\|\Pi\mathbf{M}\mathbf{x}\| = \left\|\left(\mathbf{M} - \frac{1}{T}\mathbf{1}\mathbf{1}^T\right)\left(\mathbf{I} - \frac{1}{T}\mathbf{1}\mathbf{1}^T\right)\mathbf{w}\right\| \leq \sigma\|\Pi\mathbf{w}\|. \tag{13}$$

Theorem 1. *If Assumption 1 is valid and the step size satisfies* $\alpha \leq \min\left\{\frac{(1-\sigma)^2}{8L}, \frac{11\mu}{144\mathcal{L}^2}\right\}$, *then the quantities* $\|\overline{w}^k - w^*\|^2$, $\|\Pi\mathbf{w}^k\|$ *and* $\|\Pi\mathbf{y}^k\|$ *all converge to 0 at a linear convergence rate* $\mathcal{O}\left(\max\left\{\frac{\mathcal{L}^2}{\mu^2}, \frac{\mathcal{L}}{\mu(1-\sigma)^2}\right\}\log\left(\frac{1}{\varepsilon}\right)\right)$.

Proof. By using Assumption 1, we have

$$\begin{aligned}
L(\overline{w}^{k+1}) \leq & L(\overline{w}^k) + \left\langle \nabla L(\overline{w}^k), \overline{w}^{k+1} - \overline{w}^k \right\rangle + \frac{\mathcal{L}}{2}\|\overline{w}^{k+1} - \overline{w}^k\|^2 \\
= & L(\overline{w}^k) + \left\langle \nabla L(\overline{w}^k), w - \overline{w}^k \right\rangle + \left\langle \nabla L(\overline{w}^k), \overline{w}^{k+1} - w \right\rangle \\
& + \frac{\mathcal{L}}{2}\|\overline{w}^{k+1} - \overline{w}^k\|^2 \\
\leq & L(w) - \frac{\mu}{2}\|\overline{w}^k - w\|^2 + \left\langle \nabla L(\overline{w}^k), \overline{w}^{k+1} - w \right\rangle + \frac{\mathcal{L}}{2}\|\overline{w}^{k+1} - \overline{w}^k\|^2 \\
= & L(w) - \frac{\mu}{2}\|\overline{w}^k - w\|^2 + \left\langle \overline{y}^k, \overline{w}^{k+1} - w \right\rangle \\
& + \left\langle \nabla L(\overline{w}^k) - \overline{y}^k, \overline{w}^{k+1} - w \right\rangle + \frac{\mathcal{L}}{2}\|\overline{w}^{k+1} - \overline{w}^k\|^2 \\
= & L(w) - \frac{\mu}{2}\|\overline{w}^k - w\|^2 \\
& + \frac{1}{2\alpha}\left(\|\overline{w}^k - w\|^2 - \|\overline{w}^{k+1} - w\|^2 - \|\overline{w}^{k+1} - \overline{w}^k\|^2\right) \\
& + \left\langle \nabla L(\overline{w}^k) - \overline{y}^k, \overline{w}^{k+1} - w \right\rangle + \frac{\mathcal{L}}{2}\|\overline{w}^{k+1} - \overline{w}^k\|^2 \\
\leq & L(w) - \frac{\mu}{2}\|\overline{w}^k - w\|^2 \\
& + \frac{1}{2\alpha}\left(\|\overline{w}^k - x\|^2 - \|\overline{w}^{k+1} - x\|^2 - \|\overline{w}^{k+1} - \overline{w}^k\|^2\right) \\
& + \frac{1}{\mu}\|\nabla L(\overline{w}^k) - \overline{y}^k\|^2 + \frac{\mu}{4}\|\overline{w}^{k+1} - w\|^2 \frac{\mathcal{L}}{2}\|\overline{w}^{k+1} - \overline{w}^k\|^2.
\end{aligned} \tag{14}$$

Because we stipulate that the graph $\mathcal{G}$ is undirected and connected, we can obtain that

$$\begin{aligned}
& L(\overline{w}^{k+1}) - L(w) \\
& \leq \left(\frac{1}{2\alpha} - \frac{\mu}{2}\right)\|\overline{w}^k - w\|^2 - \left(\frac{1}{2\alpha} - \frac{\mu}{4}\right)\|\overline{w}^{k+1} - w\|^2 \\
& - \left(\frac{1}{2\alpha} - \frac{\mathcal{L}}{2}\right)\|\overline{w}^{k+1} - \overline{w}^k\|^2 + \frac{1}{\mu}\left\|\frac{1}{m}\sum_i \nabla f_i(\overline{w}^k) - \frac{1}{m}\sum_i \nabla f_i(x_i^k)\right\|^2 \\
& \leq \left(\frac{1}{2\alpha} - \frac{\mu}{2}\right)\|\overline{w}^k - w\|^2 - \left(\frac{1}{2\alpha} - \frac{\mu}{4}\right)\|\overline{w}^{k+1} - w\|^2 \\
& - \left(\frac{1}{2\alpha} - \frac{\mathcal{L}}{2}\right)\|\overline{w}^{k+1} - \overline{w}^k\|^2 + \frac{\mathcal{L}^2}{\mu m}\sum_i \|\overline{w}^k - w_i^k\|^2 \\
& = \left(\frac{1}{2\alpha} - \frac{\mu}{2}\right)\|\overline{w}^k - w\|^2 - \left(\frac{1}{2\alpha} - \frac{\mu}{4}\right)\|\overline{w}^{k+1} - w\|^2 \\
& \leq -\left(\frac{1}{2\alpha} - \frac{\mathcal{L}}{2}\right)\|\overline{w}^{k+1} - \overline{w}^k\|^2 + \frac{\mathcal{L}^2}{\mu m}\|\Pi\mathbf{w}^k\|^2.
\end{aligned} \tag{15}$$

As $L(x^*) \leq L(w)$, we have

$$\begin{aligned}
\left(\frac{1}{2\alpha} - \frac{\mu}{4}\right)\|\overline{w}^{k+1} - w\|^2 \leq & \left(\frac{1}{2\alpha} - \frac{\mu}{2}\right)\|\overline{w}^k - w\|^2 \\
& - \left(\frac{1}{2\alpha} - \frac{\mathcal{L}}{2}\right)\|\overline{w}^{k+1} - \overline{w}^k\|^2 + \frac{\mathcal{L}^2}{\mu m}\|\Pi\mathbf{w}^k\|^2,
\end{aligned}$$

and

$$\begin{aligned}\|\overline{w}^{k+1}-w^*\|^2 \leq & \Big(1-\frac{\alpha\mu}{2}\Big)\|\overline{w}^k-w^*\|^2-\frac{\frac{1}{2\alpha}-\frac{\mathcal{L}}{2}}{\frac{1}{2\alpha}-\frac{\mu}{4}}\|\overline{w}^{k+1}-\overline{w}^k\|^2\\ &+\frac{\mathcal{L}^2}{\mu m(\frac{1}{2\alpha}-\frac{\mu}{4})}\|\Pi\mathbf{w}^k\|^2\\ =&\Big(1-\frac{\alpha\mu}{2}\Big)\|\overline{w}^k-w\|^2-\frac{1-\mathcal{L}\alpha}{1-\frac{\alpha\mu}{2}}\|\overline{w}^{k+1}-\overline{w}^k\|^2\\ &+\frac{2\alpha\mathcal{L}^2}{\mu m(1-\frac{\alpha\mu}{2})}\|\Pi\mathbf{w}^k\|^2.\end{aligned} \tag{16}$$

From (13), we have

$$\|\Pi\mathbf{w}^{k+1}\| \leq \sigma\|\Pi\mathbf{w}^k\|+\alpha\|\Pi\mathbf{y}^k\|,$$

and

$$\begin{aligned}\|\Pi\mathbf{y}^{k+1}\| &\leq \sigma\|\Pi\mathbf{y}^k\|+\|\nabla f(\mathbf{w}^{k+1})-\nabla f(\mathbf{w}^k)\|\\ &\leq \sigma\|\Pi\mathbf{y}^k\|+L\Big(\|\Pi\mathbf{w}^{k+1}\|+\|\Pi\mathbf{w}^k\|+\sqrt{m}\|\overline{w}^{k+1}-\overline{w}^k\|\Big).\end{aligned}$$

Thus, we have

$$\begin{aligned}&(c_1-\mathcal{L}c_2)\|\Pi\mathbf{w}^{k+1}\|+c_2\|\Pi\mathbf{y}^{k+1}\|\\ \leq&(\sigma c_1+\mathcal{L}c_2)\|\Pi\mathbf{w}^k\|+(\alpha c_1+\sigma c_2)\|\Pi\mathbf{y}^k\|+\sqrt{m}\mathcal{L}c_2\|\overline{w}^{k+1}-\overline{w}^k\|.\end{aligned}$$

Letting

$$\begin{aligned}&\sigma c_1+\mathcal{L}c_2 \leq \frac{1+\sigma}{2}(c_1-\mathcal{L}c_2) \Rightarrow c_2=\frac{1-\sigma}{3+\sigma}\frac{c_1}{\mathcal{L}},\\ &\alpha c_1+\sigma c_2 \leq \frac{1+\sigma}{2}c_2 \Rightarrow \alpha \leq \frac{1-\sigma}{2}\frac{c_2}{c_1}=\frac{1-\sigma}{2}\frac{1-\sigma}{3+\sigma}\frac{1}{\mathcal{L}}.\end{aligned} \tag{17}$$

we have

$$\begin{aligned}&(c_1/c_2-\mathcal{L})\|\Pi\mathbf{w}^{k+1}\|+\|\Pi\mathbf{y}^{k+1}\|\\ \leq&\frac{1+\sigma}{2}\Big((c_1/c_2-\mathcal{L})\|\Pi\mathbf{w}^k\|+\|\Pi\mathbf{y}^k\|\Big)+\sqrt{m}\mathcal{L}\|\overline{w}^{k+1}-\overline{w}^k\|,\end{aligned}$$

and

$$\begin{aligned}&\left(\|\Pi\mathbf{w}^{k+1}\|+\frac{1}{c_1/c_2-\mathcal{L}}\|\Pi\mathbf{y}^{k+1}\|\right)^2\\ \leq&\left(\frac{1+\sigma}{2}\left(\|\Pi\mathbf{w}^k\|+\frac{1}{c_1/c_2-\mathcal{L}}\|\Pi\mathbf{y}^k\|\right)+\frac{\sqrt{m}\mathcal{L}}{c_1/c_2-\mathcal{L}}\|\overline{w}^{k+1}-\overline{w}^k\|\right)^2\\ \leq&(1+\delta)\frac{(1+\sigma)^2}{4}\left(\|\Pi\mathbf{w}^k\|+\frac{1}{(c_1/c_2-\mathcal{L})}\|\Pi\mathbf{y}^k\|\right)^2\\ &+(1+\frac{1}{\delta})\frac{m\mathcal{L}^2}{(c_1/c_2-\mathcal{L})^2}\|\overline{w}^{k+1}-\overline{w}^k\|^2\\ =&\frac{3+\sigma}{4}\left(\|\Pi\mathbf{w}^k\|+\frac{1}{(c_1/c_2-\mathcal{L})}\|\Pi\mathbf{y}^k\|\right)^2\\ &+\frac{3+\delta}{2-\sigma-\sigma^2}\frac{m\mathcal{L}^2}{(c_1/c_2-\mathcal{L})^2}\|\overline{w}^{k+1}-\overline{w}^k\|^2,\end{aligned} \tag{18}$$

where $\delta = \frac{2-\sigma-\sigma^2}{(1+\sigma)^2}$.

Combining (16) and (18), we have

$$\begin{aligned}
&\|\overline{w}^{k+1} - w^*\|^2 + c_3\left(\|\Pi\mathbf{w}^{k+1}\| + \frac{1}{c_1/c_2 - \mathcal{L}}\|\Pi\mathbf{y}^{k+1}\|\right)^2 \\
\leq&\left(1 - \frac{\alpha\mu}{2}\right)\|\overline{w}^k - w\|^2 - \frac{1-\mathcal{L}\alpha}{1-\frac{\alpha\mu}{2}}\|\overline{w}^{k+1} - \overline{w}^k\|^2 \\
&+ \frac{2\alpha\mathcal{L}^2}{\mu m(1-\frac{\alpha\mu}{2})}\left(\|\Pi\mathbf{w}^k\| + \frac{1}{(c_1/c_2 - \mathcal{L})}\|\Pi\mathbf{y}^k\|\right)^2 \\
&+ \frac{3+\sigma}{4}c_3\left(\|\Pi\mathbf{w}^k\| + \frac{1}{(c_1/c_2 - \mathcal{L})}\|\Pi\mathbf{y}^k\|\right)^2 \\
&+ c_3\frac{3+\sigma}{2-\sigma-\sigma^2}\frac{m\mathcal{L}^2}{(c_1/c_2 - \mathcal{L})^2}\|\overline{w}^{k+1} - \overline{w}^k\|^2 \\
\leq&\left(1 - \frac{\alpha\mu}{2}\right)\|\overline{w}^k - w\|^2 + \frac{7+\sigma}{8}c_3\left(\|\Pi\mathbf{w}^k\| + \frac{1}{(c_1/c_2 - \mathcal{L})}\|\Pi\mathbf{y}^k\|\right)^2 \\
\leq&\max\left\{1-\frac{\alpha\mu}{2}, \frac{7+\sigma}{8}\right\}\left(\|\overline{w}^k - w\|^2 + c_3\left(\|\Pi\mathbf{w}^k\| + \frac{1}{(c_1/c_2 - \mathcal{L})}\|\Pi\mathbf{y}^k\|\right)^2\right).
\end{aligned}$$

By letting

$$\begin{aligned}
&\frac{1-\sigma}{8}c_3 = \frac{2\alpha\mathcal{L}^2}{\mu m(1-\frac{\alpha\mu}{2})}, \\
&c_3\frac{3+\sigma}{2-\sigma-\sigma^2}\frac{m\mathcal{L}^2}{(c_1/c_2 - \mathcal{L})^2} \leq \frac{1-\mathcal{L}\alpha}{1-\frac{\alpha\mu}{2}}.
\end{aligned} \tag{19}$$

At last, we only need to consider α satisfying (17) and (19). $\alpha \leq \frac{(1-\sigma)^2}{8L}$ satisfies (17). Equation (19) is equivalent to

$$\begin{aligned}
&\frac{8}{1-\sigma}\frac{2\alpha\mathcal{L}^2}{\mu m(1-\frac{\alpha\mu}{2})}\frac{3+\sigma}{2-\sigma-\sigma^2}\frac{m\mathcal{L}^2}{(c_1/c_2 - \mathcal{L})^2} \leq \frac{1-\mathcal{L}\alpha}{1-\frac{\alpha\mu}{2}} \\
\Leftrightarrow&\frac{8}{1-\sigma}\frac{2\alpha\mathcal{L}^2}{\mu m}\frac{3+\sigma}{2-\sigma-\sigma^2}\frac{m\mathcal{L}^2}{(\frac{3+\sigma}{1-\sigma}\mathcal{L} - \mathcal{L})^2} \leq 1-\mathcal{L}\alpha \\
\Leftrightarrow&\frac{4\alpha\mathcal{L}^2}{\mu}\frac{3+\sigma}{2-\sigma-\sigma^2}\frac{1-\sigma}{(1+\sigma)^2} \leq 1-\mathcal{L}\alpha \\
\Leftrightarrow&\frac{4\alpha\mathcal{L}^2}{\mu}\frac{3+\sigma}{(1+\sigma)^2} \leq 1-\mathcal{L}\alpha \\
\Leftarrow&\frac{12\alpha\mathcal{L}^2}{\mu} \leq 1-\mathcal{L}\alpha.
\end{aligned}$$

Thus, we only need $\alpha \leq \frac{11\mu}{144\mathcal{L}^2}$, and we have

$$\begin{aligned}
&\|\overline{w}^{k+1} - w^*\|^2 + c_3\left(\|\Pi\mathbf{w}^{k+1}\| + \frac{1}{c_1/c_2 - \mathcal{L}}\|\Pi\mathbf{y}^{k+1}\|\right)^2 \\
\leq&\max\left\{1 - \frac{11\mu^2}{288\mathcal{L}^2}, 1 - \frac{(1-\sigma)^2\mu}{16L}, \frac{7+\sigma}{8}\right\} \\
&\times\left(\|\overline{w}^k - w\|^2 + c_3\left(\|\Pi\mathbf{w}^k\| + \frac{1}{(c_1/c_2 - \mathcal{L})}\|\Pi\mathbf{y}^k\|\right)^2\right).
\end{aligned}$$

Thus, the algorithm needs $O\left((\frac{\mathcal{L}^2}{\mu^2} + \frac{\mathcal{L}}{\mu(1-\sigma)^2})\log\frac{1}{\epsilon}\right)$ iterations to find an ϵ-optimal solution. □

6. Numerical Experiments

Some numerical experiments were conducted to conduct a comprehensive exploration of the proposed DNFNN, including the test accuracy across different networks, the noise tolerance capability of DNFNN, the convergence performance of regression problems, and the test accuracy after adding noise to datasets. We employed two main strategies to evaluate the performance of our model to ensure the experimental results provided a fair assessment of the generalization capability of the proposed DNFNN algorithm. On one hand, we used a 10-fold cross-validation, which means the dataset was divided into 10 subsets. The model was trained on nine subsets and validated on the remaining one, and this process was repeated 10 times with different validation sets. On the other hand, we also used a standard training/test split, where the dataset was divided into two parts, one for training with validation and the other for testing. This method allowed us to assess how well the model generalized to unseen data, further helping to mitigate overfitting. It should be noted that the numerical experiments presented in the paper focus primarily on classification and regression tasks, which are directly applicable to real-world applications such as image recognition, pattern detection, time-series forecasting, financial prediction, or even environmental modeling.

6.1. Classification Problems

Six classification datasets of various sizes from UCI (http://archive.ics.uci.edu/ml/index.php accessed on 1 December 2024) were utilized in our experiments. The detailed information of these datasets is listed in Table 1. The column "Imbalance rate" represents the disparity in the number of samples between categories of datasets. Each feature in each dataset is normalized to $[-1, 1]$ by the following scaling:

$$x_j^{'} = \frac{x_j - x_{mean}}{x_{\max} - x_{\min}}. \tag{20}$$

To minimize the impact of random weight initialization and enable cross-validation, each experiment was repeated ten times using randomly sampled training data (seventy percent of the total), with average results reported. All experiments were conducted on the PyCharm platform, utilizing an Intel 3.30 GHz CPU and 8 GB of memory. It should be noted that the main focus of our work lay in the model innovation, and we have not extended our approach to hardware deployment yet.

Table 1. The description of the classification datasets.

No.	Datasets	Samples	Dimensions	Classes	Imbalance Rate
1	Glass	214	9	6	8.44
2	Liver	345	6	2	1.38
3	Sonar	208	60	2	1.14
4	Vehicle	846	18	4	1.10
5	Leukemia	72	7129	2	1.88
6	CNS	42	7129	5	2.5
7	Ecoli	336	7	8	71.50
8	CLAVE	10,796	16	4	10.58

The hyperparameters of the DNFNN were $\{T, \alpha, \theta\}$, where T is the number of sub-networks, α and θ are the parameters for perturbation of centers as used in the following equation (i.e., Equation (3)):

$$c_{li}(t) = a_{li} + \alpha \sin(\theta(t-1)).$$

and we selected the hyperparameters $\alpha = 0.3$, $\theta = 0.1$, and $T = 30$ using a Grid Search approach. These values provided a balance between model performance and training efficiency. Initially, experiments were performed to evaluate the classification accuracy and computation time of the proposed DNFNN, comparing it with related methods such as type-1 TSK FNN, interval type-2 FNN [44], and ANFNN [21]. For consistency, the same experimental settings, including learning rate and fuzzy rule generation strategy, were applied to all models. The maximum iteration count was set to 5000. The average test accuracies and runtime are summarized in Table 2. We have used bold font for the best testing accuracy.

Table 2. Performance comparison between DNFNN and related methods.

# Datasets	Type-1 FNN		Interval Type-2 FNN		ANFNN		DNFNN	
	TeAcc	Time (s)	TeAcc	Time (s)	TeAcc	Time (s)	TeAcc	Time (s)
1 Glass	0.657	1.3	0.671	38.0	0.673	21.1	**0.677**	15.9
2 Liver	0.622	1.2	0.679	11.3	0.691	10.7	**0.711**	10.5
3 Sonar	0.785	7.8	0.847	15.9	0.849	10.8	**0.877**	10.1
4 Vehicle	0.634	4.1	0.644	33.7	0.677	31.1	**0.702**	25.1
5 Leukemia	0.944	11.6	0.951	311.7	0.955	106.6	**0.957**	95.4
6 CNS	0.731	33.6	0.752	847.7	0.756	282.4	**0.803**	225.4
7 CLAVE	0.708	28.3	0.711	811.6	0.721	401.1	**0.727**	411.7
8 Ecoli	0.830	0.7	0.851	17.5	0.857	8.2	**0.881**	8.5

As shown in Table 2, the DNFNN achieved the highest test accuracy for most datasets and attained comparable accuracy for the remaining ones. Compared to a single type-1 FNN, the DNFNN consistently outperformed in terms of accuracy across all datasets, with a notable improvement of over seven percent in test accuracy for the CNS dataset. When compared to the interval type-2 FNN, we not only achieved higher classification accuracy but also significantly reduced the training time cost. Even when compared to the most recent fuzzy network algorithm, ANFNN, we achieved a 3% accuracy improvement on the Sonar and Vehicle datasets, and even a 5% improvement on the CNS dataset. Furthermore, our model outperformed all the other models across all datasets. These results validate that the new update strategy improves both accuracy and efficiency, demonstrating that the DNFNN offers a promising alternative to other fuzzy neural networks in various scenarios, with significant potential for real-world applications.

6.2. Robustness of DNFNN

In this section, we introduced random noise to certain model parameters to assess the robustness of the DNFNN: (1) noise in centers and widths, and (2) noise in weights. The injected noise was additive Gaussian noise with a zero mean and a standard deviation of 0.5. Based on our observations, the signal-to-noise ratio of the Gaussian center, Gaussian width, and connection weights were 16.110, 18.677, and 14.444, respectively. For each dataset, we report the average performance across six experiments.

(1) Noise in centers and widths: After training the model, random noise was added to the centers and widths separately, and the corresponding testing accuracies were recorded.

The results are summarized in Table 3. As observed, the DNFNN showed a decrease in accuracy after noise was added, but the decline was insignificant. A similar trend was seen when additive noise was applied to the widths. These findings demonstrate that the DNFNN exhibits exceptional robustness and noise tolerance.

Table 3. DNFNN with additive noise to the centers and widths.

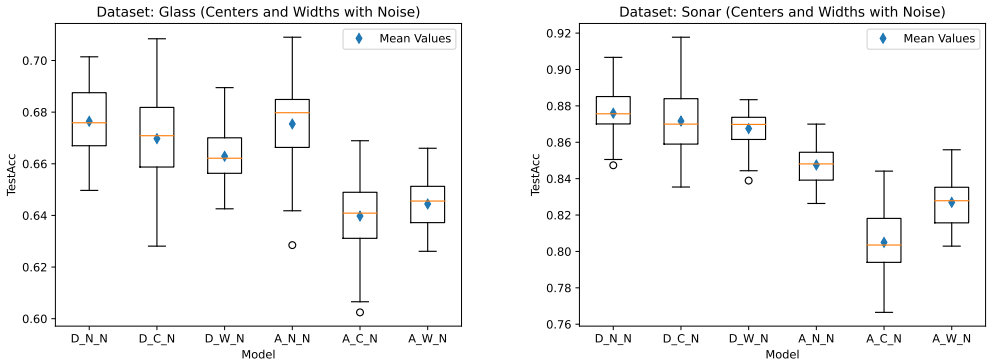
# Datasets	None	Centers	Widths
	TestAcc (std)	TestAcc (std)	TestAcc (std)
1 Glass	67.71% (±2.14)	66.33% (±2.77)	66.34% (±2.18)
2 Liver	71.19% (±3.17)	71.06% (±4.09)	69.77% (±4.51)
3 Sonar	87.78% (±2.79)	87.06% (±3.79)	87.01% (±3.54)
4 Vehicle	70.20% (±2.01)	68.41% (±2.41)	68.03% (±2.89)
5 Leukemia	95.71% (±3.77)	94.14% (±3.99)	94.09% (±4.37)
6 CNS	80.31% (±6.29)	78.77% (±7.14)	78.21% (±7.26)

To further illustrate its robustness, we select four representative datasets—Glass, Sonar, Leukemia, and CNS—and present box plots for 50 numerical experimental results, as shown in Figure 2. In each plot, the first three boxes represent the DNFNN without noise, the DNFNN with noise in the center, and the DNFNN with noise in the width. The following three boxes represent the corresponding results for the ANFNN model. The ordinate indicates the test accuracy, with the small blue rhombus and orange line representing the average and median values, respectively. As shown in Figure 2, the test accuracies of both models declined when noise was introduced. However, the DNFNN demonstrated superior resistance to interference compared to the ANFNN.

(2) Noise in weights: In this experiment, noise was injected into the weights between the rule layer and the output layer. The average testing accuracies, with and without noise injection, for both ANFNN and DNFNN are reported in Table 4. As in previous experiments, the DNFNN yielded more stable results (see Table 4). After noise was added, the average accuracy loss for the ANFNN and DNFNN was approximately 3.5% and 1.5%, respectively.

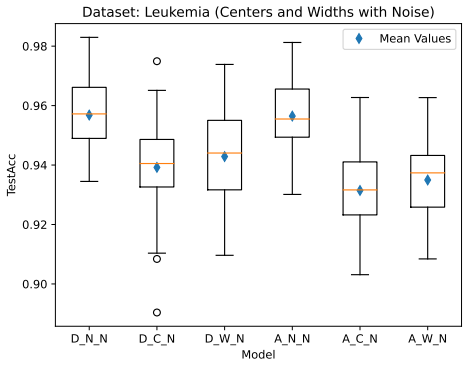
Table 4. Performance comparison of the ANFNN and DNFNN when noise was added to the weights.

# Datasets	ANFNN	ANFNN (W)	DNFNN	DNFNN (W)
	TestAcc (std)	TestAcc (std)	TestAcc (std)	TestAcc (std)
1 Glass	67.30% (±4.44)	64.07% (±4.87)	67.71% (±2.66)	66.07% (±2.79)
2 Liver	69.12% (±3.19)	65.04% (±6.17)	71.16% (±2.13)	69.24 % (±3.14)
3 Sonar	84.92% (±4.47)	79.24% (±4.49)	87.76% (±3.77)	85.29% (±3.81)
4 Vehicle	67.70% (±3.91)	64.71% (±5.07)	70.21% (±3.08)	66.07% (±4.11)
5 Leukemia	95.50% (±3.77)	94.09% (±4.91)	95.70% (±2.09)	94.99% (±2.88)
6 CNS	75.60% (±8.72)	72.61% (±10.91)	80.30% (±9.09)	78.06% (±10.57)

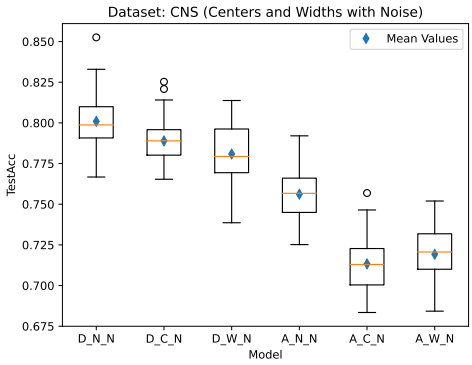


(**a**) Glass

(**b**) Sonar



(**c**) Leukemia

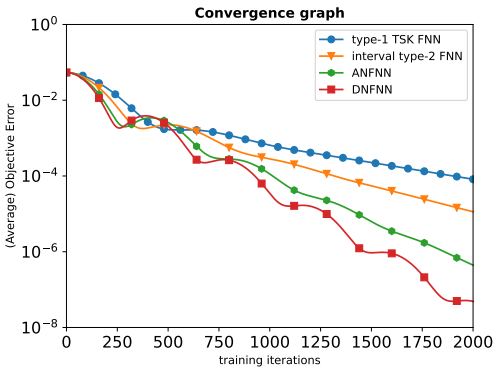


(**d**) CNS

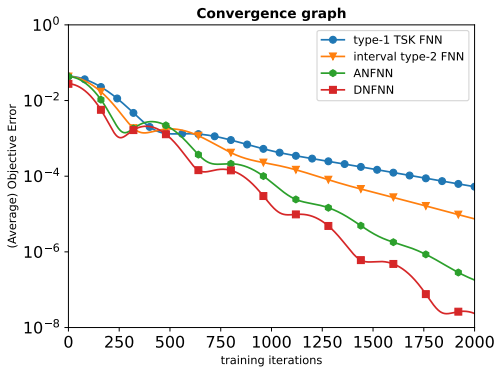
Figure 2. Box plots to compare the noise tolerance of the ANFNN and DNFNN for different datasets.

6.3. Performance on Regression Problems

We also performed experiments on regression tasks using the "Concrete Compressive Strength" dataset, which can be accessed via UCI (https://archive.ics.uci.edu/dataset/165/concrete+compressive+strength accessed on 1 December 2024). To evaluate the efficacy of the DNFNN, we compared its convergence performance with several related fuzzy approaches, including the type-1 TSK FNN, interval type-2 FNN, and ANFNN. The results in Figure 3 consistently demonstrate that DNFNN exhibits superior convergence efficiency, achieving both a faster convergence rate and a smaller final objective error.



(**a**) Twenty sub-networks



(**b**) Thirty sub-networks

Figure 3. The convergence curves for regression task under different sub-networks.

6.4. Performance Comparison of DNFNN and Other Typical Neural Networks When Adding Noise

In addition, we conducted a comparative experiment where we evaluated the performance of the DNFNN against several other methods, including type-1 FNN, type-2 FNN (interval type-2 FNN), five-layer MLP, RBF neural network, and ANFNN. Among these, the five-layer MLP and RBF neural network are classic neural network architectures that are not based on fuzzy systems. We selected three datasets for testing: Glass, Liver, and CNS. Gaussian noise with a mean of zero and a standard deviation of 0.3 was added to each dataset. The final classification accuracies over the three datasets are summarized in the Table 5. From Table 5, it is evident that the models based on fuzzy systems demonstrated superior classification accuracy. The RBF neural network, which employed Gaussian functions as activation functions, outperformed the MLP that used ReLU. Overall, the DNFNN maintained a high level of classification accuracy, indicating that the proposed method had a significant advantage in addressing uncertainty issues.

Table 5. Performance comparison of DNFNN and other methods when noise was added to the datasets.

# Datasets	Type-1 FNN	Type-2 FNN	5-Layer-MLN	RBFNN	ANFNN	DNFNN
1 Glass	64.09%	66.27%	62.13%	63.27%	66.32%	66.51%
2 Liver	61.22%	66.81%	60.27%	61.36%	67.75%	68.21%
6 CNS	71.43%	73.02%	69.27%	70.94%	74.03%	78.27%

7. Conclusions and Discussion

In this paper, we proposed a decentralized update strategy and an efficient fuzzy neural network based on nonstationary fuzzy sets. By employing a meta-learning approach, we assigned varying weights to each sub-network, thereby enhancing the adaptive capabilities of the network. We substantiated the rigor of the DNFNN by proving the linear convergence of the decentralized backward update algorithm. Numerical simulations demonstrated that the proposed DNFNN outperformed existing fuzzy neural networks in terms of classification and prediction accuracy. Additionally, the robustness of the DNFNN against noise was validated under three different scenarios using six datasets from the UCI repository. The convergence curves for regression tasks revealed that the proposed model achieved superior convergence efficiency. Also, decentralized optimization allowed for parallel training and the security and privacy of sensitive data. Looking ahead, we plan to further reduce the computational cost by improving the backpropagation algorithm and utilizing transfer learning. In addition, we will explore the potential applications of the DNFNN in areas such as communication compression, incorporating asynchronous networks, IoT systems, and biopotential signal processing.

Author Contributions: Methodology, M.Y.; Formal analysis, Z.Z.; Resources, Z.Z.; Data curation, M.Y.; Writing—review & editing, H.J.; Project administration, H.J.; Funding acquisition, H.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Han, H.G.; Chen, Z.Y.; Liu, H.X.; Qiao, J.F. A self-organizing interval type-2 fuzzy-neural-network for modeling nonlinear systems. *Neurocomputing* **2018**, *290*, 196–207. [CrossRef]
2. Liu, J.; Wang, W.; Golnaraghi, F.; Kubica, E. A neural fuzzy framework for system mapping applications. *Knowl.-Based Syst.* **2010**, *23*, 572–579. [CrossRef]
3. Liu, F.; Liu, C.; Rao, H.; Xu, Y.; Huang, T. Reliable impulsive synchronization for fuzzy neural networks with mixed controllers. *Neural Netw.* **2021**, *143*, 759–766. [CrossRef] [PubMed]
4. Huang, W.; Xu, Z.; Zhu, L. The minimum regret path problem on stochastic fuzzy time-varying networks. *Neural Netw.* **2022**, *153*, 450–460. [CrossRef]
5. Kadak, U. Multivariate fuzzy neural network interpolation operators and applications to image processing. *Expert Syst. Appl.* **2022**, *206*, 117771. [CrossRef]
6. Karnik, N.N.; Mendel, J.M.; Liang, Q. Type-2 fuzzy logic systems. *IEEE Trans. Fuzzy Syst.* **1999**, *7*, 643–658. [CrossRef]
7. Karnik, N.N.; Mendel, J.M. Centroid of a type-2 fuzzy set. *Inf. Sci.* **2001**, *132*, 195–220. [CrossRef]
8. Zhou, S.M.; Da Xu, L. A new type of recurrent fuzzy neural network for modeling dynamic systems. *Knowl.-Based Syst.* **2001**, *14*, 243–251. [CrossRef]
9. Liu, Y.; Wang, X.k.; Hou, W.h.; Liu, H.; Wang, J.q. A novel hybrid model combining a fuzzy inference system and a deep learning method for short-term traffic flow prediction. *Knowl.-Based Syst.* **2022**, *255*, 109760. [CrossRef]
10. Zhang, Z. On interval type-2 rough fuzzy sets. *Knowl.-Based Syst.* **2012**, *35*, 1–13. [CrossRef]
11. Castro, J.R.; Castillo, O.; Melin, P.; Rodríguez-Díaz, A. A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks. *Inf. Sci.* **2009**, *179*, 2175–2193. [CrossRef]
12. Wang, C.H.; Cheng, C.S.; Lee, T.T. Dynamical optimal training for interval type-2 fuzzy neural network (T2FNN). *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2004**, *34*, 1462–1477. [CrossRef]
13. Juang, C.F.; Tsao, Y.W. A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning. *IEEE Trans. Fuzzy Syst.* **2008**, *16*, 1411–1424. [CrossRef]
14. Dymova, L.; Sevastjanov, P.; Tikhonenko, A. An interval type-2 fuzzy extension of the TOPSIS method using alpha cuts. *Knowl.-Based Syst.* **2015**, *83*, 116–127. [CrossRef]
15. Mohammadzadeh, A.; Kayacan, E. A non-singleton type-2 fuzzy neural network with adaptive secondary membership for high dimensional applications. *Neurocomputing* **2019**, *338*, 63–71. [CrossRef]
16. Birkin, P.A.; Garibaldi, J.M. A novel dual-surface type-2 controller for micro robots. In Proceedings of the International Conference on Fuzzy Systems, Barcelona, Spain, 18–23 July 2010; pp. 1–8. [CrossRef]
17. Ozen, T.; Garibaldi, J.; Musikasuwan, S. Modelling the variation in human decision making. In Proceedings of the IEEE Annual Meeting of the Fuzzy Information, 2004. Processing NAFIPS '04, Banff, AB, Canada, 27–30 June 2004; Volume 2, pp. 617–622. [CrossRef]
18. Garibaldi, J.M.; Ozen, T. Uncertain Fuzzy Reasoning: A Case Study in Modelling Expert Decision Making. *IEEE Trans. Fuzzy Syst.* **2007**, *15*, 16–30. [CrossRef]
19. Garibaldi, J.M.; Jaroszewski, M.; Musikasuwan, S. Nonstationary fuzzy sets. *IEEE Trans. Fuzzy Syst.* **2008**, *16*, 1072–1086. [CrossRef]
20. Benatar, N.; Aickelin, U.; Garibaldi, J.M. A comparison of non-stationary, type-2 and dual surface fuzzy control. In Proceedings of the 2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011), Taipei, Taiwan, 27–30 June 2011; pp. 1193–1200. [CrossRef]
21. Chang, Q.; Zhang, Z.; Wei, F.; Wang, J.; Pedrycz, W.; Pal, N.R. Adaptive Nonstationary Fuzzy Neural Network. *Knowl.-Based Syst.* **2024**, *288*, 111398. [CrossRef]
22. Chen, M.; Gündüz, D.; Huang, K.; Saad, W.; Bennis, M.; Feljan, A.V.; Poor, H.V. Distributed learning in wireless networks: Recent progress and future challenges. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 3579–3605. [CrossRef]
23. Nedić, A.; Olshevsky, A.; Shi, W. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM J. Optim.* **2017**, *27*, 2597–2633. [CrossRef]
24. Huang, L.; Wu, J.; Shi, D.; Dey, S.; Shi, L. Differential privacy in distributed optimization with gradient tracking. *IEEE Trans. Autom. Control.* **2024**, *69*, 5727–5742. [CrossRef]
25. Saramago, S.; Steffen, V. Trajectory modeling of robot manipulators in the presence of obstacles. *J. Optim. Theory Appl.* **2001**, *110*, 17–34. [CrossRef]
26. Candes, E.J.; Tao, T. Decoding by linear programming. *IEEE Trans. Inf. Theory* **2005**, *51*, 4203–4215. [CrossRef]
27. Donoho, D.L. Compressed sensing. *IEEE Trans. Inf. Theory* **2006**, *52*, 1289–1306. [CrossRef]
28. Nedić, A.; Ozdaglar, A. Distributed subgradient methods for multi-agent optimization. *IEEE Trans. Autom. Control* **2009**, *54*, 48–61. [CrossRef]

29. Shi, W.; Ling, Q.; Wu, G.; Yin, W. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM J. Optim.* **2015**, *25*, 944–966. [CrossRef]
30. Xu, J.; Zhu, S.; Soh, Y.C.; Xie, L. Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes. In Proceedings of the 2015 54th IEEE Conference on Decision and Control (CDC), Osaka, Japan, 15–18 December 2015; IEEE: New York, NY, USA, 2015; pp. 2055–2060.
31. Xi, C.; Mai, V.S.; Xin, R.; Abed, E.H.; Khan, U.A. Linear convergence in optimization over directed graphs with row-stochastic matrices. *IEEE Trans. Autom. Control* **2018**, *63*, 3558–3565. [CrossRef]
32. Song, Z.; Shi, L.; Pu, S.; Yan, M. Compressed gradient tracking for decentralized optimization over general directed networks. *IEEE Trans. Signal Process.* **2022**, *70*, 1775–1787. [CrossRef]
33. Lake, B.M.; Salakhutdinov, R.; Tenenbaum, J.B. Human-level concept learning through probabilistic program induction. *Science* **2015**, *350*, 1332–1338. [CrossRef] [PubMed]
34. Jun, S.; Zongben, X.; Deyu, M. Small sample learning in big data era. *arXiv* **2018**, arXiv:1808.04572.
35. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 1126–1135.
36. Finn, C.B. *Learning to Learn with Gradients*; University of California: Berkeley, CA, USA, 2018.
37. Freund, Y.; Schapire, R.E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [CrossRef]
38. Ross, T.Y.; Dollár, G. Focal loss for dense object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2980–2988.
39. Chang, H.S.; Learned-Miller, E.; McCallum, A. Active bias: Training more accurate neural networks by emphasizing high variance samples. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1003–1013.
40. Laganà, F.; Pratticò, D.; Angiulli, G.; Oliva, G.; Pullano, S.A.; Versaci, M.; La Foresta, F. Development of an Integrated System of sEMG Signal Acquisition, Processing, and Analysis with AI Techniques. *Signals* **2024**, *5*, 476–493. [CrossRef]
41. Kumar, M.; Packer, B.; Koller, D. Self-paced learning for latent variable models. *Adv. Neural Inf. Process. Syst.* **2010**, *23*, 1189–1197.
42. De La Torre, F.; Black, M.J. A framework for robust subspace learning. *Int. J. Comput. Vis.* **2003**, *54*, 117–142. [CrossRef]
43. Jiang, L.; Zhou, Z.; Leung, T.; Li, L.J.; Li, F.-F. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 2304–2313.
44. Wang, J.; Chang, Q.; Gao, T.; Zhang, K.; Pal, N.R. Sensitivity analysis of Takagi–Sugeno fuzzy neural network. *Inf. Sci.* **2022**, *582*, 725–749. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.