HyperPattern Recognition and SuperHyperPattern Recognition with some applications

Takaaki Fujita [1] *
[1] Independent Researcher, Shinjuku, Shinjuku-ku, Tokyo, Japan. Takaaki.fujita060@gmail.com

Abstract

A hyperfunction maps each input to a subset of outputs, generalizing classical functions to represent multi-valued or uncertain outcomes [1]. A superhyperfunction extends this concept by mapping collections of sets to higher-order powerset values, thereby capturing layered hierarchical uncertainties [2]. A Pattern Recognizer assigns labels to instances by minimizing empirical loss on training data. In this paper, we introduce and analyze HyperPattern Recognition and SuperHyperPattern Recognition, which extend the Pattern Recognizer framework via hyperfunctions and superhyperfunctions. Our treatment is purely theoretical, and we anticipate future computational experiments on real-world datasets to empirically validate the proposed models.

Keywords: Function, HyperFunction, SuperHyperFunction, Pattern Recognition, HyperPattern Recognition, SuperHyperPattern Recognition

Structure of this paper

The format of this paper is described below.

1 Preliminaries **1**
1.1 Hyperfunction and n-Superhyperfunction 1
1.2 (m,n)-Superhyperfunction 3
2 Result: HyperPattern Recognition **4**
2.1 Pattern Recognition 4
2.2 HyperPattern recognition 5
2.3 SuperHyperPattern recognition 6
3 Conclusion **10**

1 Preliminaries

We introduce here the fundamental definitions and notation used throughout this paper. Unless stated otherwise, all sets under consideration are finite.

1.1 Hyperfunction and n-Superhyperfunction

Hyperstructures [3–5] and n-Superhyperstructures [6–10] are frameworks for modeling hierarchical concepts in various domains. For example, SuperHyperGraph [11–14] and SuperHyperAlgebra [1, 15, 16] are concrete instances of an n-Superhyperstructure. In the context of functions, the notions of Hyperfunction and n-Superhyperfunction are well-known [2, 17, 18]. Relevant definitions and theorems are presented below.

Definition 1.1 (Universe)**.** Let U be a nonempty finite set, called the *universe* or *base set*. Every subsequent construction—powersets, hyperstructures, and so on—is built upon U.

Definition 1.2 (Power Set)**.** [19, 20] The *power set* of U is

$$\mathcal{P}(U) \;=\; \{\, A \mid A \subseteq U \}.$$

Definition 1.3 (Iterated Powerset)**.** [21–24] For each integer $n \geq 1$, the n-fold iterated powerset of U is defined by

$$\mathcal{P}^1(U) = \mathcal{P}(U), \quad \mathcal{P}^{n+1}(U) = \mathcal{P}\big(\mathcal{P}^n(U)\big).$$

If one wishes to exclude the empty set at each iteration, replace $\mathcal{P}$ with

$$\mathcal{P}^*(X) \;=\; \mathcal{P}(X) \setminus \{\emptyset\}.$$

Definition 1.4 (Hyperoperation)**.** (cf. [6]) A *hyperoperation* is a generalization of a binary operation where the result of combining two elements is a set, not a single element. Formally, for a set S, a hyperoperation $\circ$ is defined as:

$$\circ : S \times S \to \mathcal{P}(S),$$

where $\mathcal{P}(S)$ is the powerset of S.

Definition 1.5 (Hyperfunction)**.** [2, 17] A *Hyperfunction* is a function where the domain remains a classical set S, but the codomain is extended to the powerset of S, denoted $\mathcal{P}(S)$. Formally, a Hyperfunction f is defined as:

$$f : S \to \mathcal{P}(S).$$

For any $x \in S$, $f(x) \subseteq S$ is a subset of S. This allows the function to map an element of S to multiple elements, enabling greater flexibility compared to classical functions.

Example 1.6 (Thesaurus as a Hyperfunction)**.** Let S be the set of all English words. We define a hyperfunction

$$f\colon S \longrightarrow \mathcal{P}(S)$$

by letting $f(w)$ be the set of synonyms of the word w. Concretely:

$$f(\text{“happy”}) = \{\text{“joyful”, “content”, “pleased”}\},$$

$$f(\text{“fast”}) = \{\text{“quick”, “rapid”, “swift”}\}.$$

This hyperfunction models a thesaurus lookup, mapping each word to the (possibly empty) set of its synonyms.

Definition 1.7 (SuperHyperOperations)**.** [21] Let H be a non-empty set, and let $P(H)$ be the powerset of H. Define the n-th powerset $P^n(H)$ recursively:

$$P^0(H) = H, \quad P^{k+1}(H) = P(P^k(H)) \text{ for } k \geq 0.$$

A *SuperHyperOperation* of order (m, n) is an m-ary operation:

$$\circ^{(m,n)} : H^m \to P_*^n(H)$$

where $P_*^n(H)$ denotes the n-th powerset of H possibly excluding the empty set (for a classical-type SuperHyperOperation) or including it (for a Neutrosophic-type SuperHyperOperation).

If the codomain is $P_*^n(H)$ without the empty set, we call it a *classical-type (m,n)-SuperHyperOperation*. If the codomain is $P^n(H)$ including the empty set, we call it a *Neutrosophic (m,n)-SuperHyperOperation*.

In either case, these SuperHyperOperations are higher-order generalizations of hyperoperations, capturing multi-level complexity through n-th powerset constructions.

Definition 1.8 (n-Superhyperfunction)**.** [2,17,18,25,26] An *n-Superhyperfunction* generalizes the concept of a Hyperfunction by using the n-th powerset $\mathcal{P}_n(S)$ as the codomain. Formally, for $n \geq 2$, an n-Superhyperfunction f is defined as:

$$f : \mathcal{P}_r(S) \to \mathcal{P}_n(S),$$

where $0 \leq r \leq n$, and $\mathcal{P}_n(S)$ is the n-th powerset of S. This definition allows f to map subsets of S (from $\mathcal{P}_r(S)$) to elements in the n-th powerset $\mathcal{P}_n(S)$.

Example 1.9 (Recipe Finder as a 2-Superhyperfunction)**.** Let S be the finite set of all ingredients available in a recipe database, and let

$$\mathcal{R} \subseteq \mathcal{P}(S)$$

be the collection of all valid recipes, each represented as a subset of S. Define

$$f : \mathcal{P}(S) = \mathcal{P}_1(S) \longrightarrow \mathcal{P}\big(\mathcal{P}(S)\big) = \mathcal{P}_2(S)$$

by

$$f(I) = \big\{ R \subseteq S \mid I \subseteq R \text{ and } R \in \mathcal{R} \big\}.$$

In words, for any input ingredient set $I \subseteq S$, $f(I)$ is the set of all recipes R that contain I. Thus f maps a subset of ingredients to a subset of the powerset of S, i.e. a set of recipes. This exactly realizes a 2-Superhyperfunction in practice.

1.2 (m, n)-Superhyperfunction

An (m, n)-Superhyperfunction is a mapping from the m-th iterated powerset of S to its n-th iterated powerset, capturing hierarchical uncertainties (cf. [2]).

Definition 1.10 ((m, n)-Superhyperfunction)**.** Let S be a nonempty set and let m, n be integers with $m, n \geq 1$. Denote by $\mathcal{P}_k(S)$ the k-fold iterated powerset of S. An (m, n)-*Superhyperfunction* on S is a mapping

$$f : \mathcal{P}_m(S) \longrightarrow \mathcal{P}_n(S).$$

This notion recovers the usual n-Superhyperfunction when $m = r \leq n$, i.e. a mapping $\mathcal{P}_r(S) \to \mathcal{P}_n(S)$.

Example 1.11 (Corporate Hierarchy)**.** Let S be the set of all employees in a company. We construct:

- $\mathcal{P}_1(S)$: the set of all *departments*, each a subset of S;
- $\mathcal{P}_2(S)$: the set of all *divisions*, each a set of departments;
- $\mathcal{P}_3(S)$: the set of all *business units*, each a set of divisions.

Let

$$D_1 = \big\{\{A, B\}, \{C\}\big\}, \quad D_2 = \big\{\{A\}, \{B, C, D\}\big\},$$
$$\mathcal{D} = \{D_1, D_2\} \subseteq \mathcal{P}_2(S), \quad U_1 = \{D_1, D_2\}, \quad \mathcal{U} = \{U_1\} \subseteq \mathcal{P}_3(S).$$

Define

$$f\colon \mathcal{P}_2(S) \longrightarrow \mathcal{P}_3(S), \quad f(D) = \{\, U \in \mathcal{U} \mid D \in U\}.$$

Then f maps each division D to the set of business units that include it. For instance,

$$f(D_1) = \{U_1\}, \quad f(D_2) = \{U_1\}.$$

Hence f is a $(2, 3)$-Superhyperfunction on S.

Example 1.12 (Manufacturing Supply Chain)**.** Let

$$S = \{\text{Steel, Plastic, Glass, Rubber}\}.$$

Then the 2-fold iterated powerset is

$$\mathcal{P}_2(S) = \{P_1, P_2, P_3\},$$

where

$$P_1 = \{\text{Steel, Rubber}\}, \quad P_2 = \{\text{Plastic, Glass}\}, \quad P_3 = \{\text{Steel, Glass}\}.$$

The 3-fold iterated powerset is

$$\mathcal{P}_3(S) = \{A_1, A_2\},$$

where

$$A_1 = \{P_1, P_2\}, \quad A_2 = \{P_3\}.$$

Define

$$f\colon \mathcal{P}_2(S) \longrightarrow \mathcal{P}_3(S), \qquad f(P) = \{\, A \in \mathcal{P}_3(S) \mid P \in A\}.$$

Concretely,

$$f(P_1) = \{A_1\}, \quad f(P_2) = \{A_1\}, \quad f(P_3) = \{A_2\}.$$

Thus f is a $(2, 3)$-Superhyperfunction mapping each intermediate component P_i to the set of final assemblies A_j in which it appears.

Theorem 1.13 (Generalization of n-Superhyperfunctions)**.** *Let S be a nonempty set and $n \geq 1$. For any integer r with $0 \leq r \leq n$, every n-Superhyperfunction*

$$f : \mathcal{P}_r(S) \longrightarrow \mathcal{P}_n(S)$$

can be embedded into an (m, n)-Superhyperfunction for any $m \geq r$. Concretely, define the canonical embedding

$$\varphi_{r,m} : \mathcal{P}_r(S) \longrightarrow \mathcal{P}_m(S) \quad by \quad \varphi_{r,r}(A) = A, \quad \varphi_{r,k+1}(A) = \{\varphi_{r,k}(A)\} \quad (k \geq r).$$

Then $\varphi_{r,m}$ is injective, and the extension

$$F(X) \;=\; \begin{cases} f\big(\varphi_{r,m}^{-1}(X)\big), & X \in \mathrm{Im}(\varphi_{r,m}), \\ \emptyset, & \textit{otherwise}, \end{cases}$$

defines an (m, n)-Superhyperfunction

$$F : \mathcal{P}_m(S) \longrightarrow \mathcal{P}_n(S)$$

which extends f.

Proof. We proceed in two steps:

(1) Injectivity of $\varphi_{r,m}$. By induction on $k \geq r$, observe that

$$\varphi_{r,r}(A) = A, \quad \varphi_{r,k+1}(A) = \{\varphi_{r,k}(A)\}$$

always produces a singleton around the previous image. Hence if $A \neq B$ in $\mathcal{P}_r(S)$, then $\varphi_{r,k}(A) \neq \varphi_{r,k}(B)$ for all $k \geq r$, proving injectivity of $\varphi_{r,m}$.

(2) Construction of the extension F. Since $\varphi_{r,m}$ is injective, every X in its image can be written uniquely as $X = \varphi_{r,m}(A)$ for some $A \in \mathcal{P}_r(S)$. We define

$$F : \mathcal{P}_m(S) \longrightarrow \mathcal{P}_n(S) \quad \text{by} \quad F(X) = \begin{cases} f\big(\varphi_{r,m}^{-1}(X)\big), & X \in \mathrm{Im}(\varphi_{r,m}), \\ \emptyset, & \text{otherwise.} \end{cases}$$

Clearly $F(X) \in \mathcal{P}_n(S)$ for every $X \in \mathcal{P}_m(S)$, so F is an (m, n)-Superhyperfunction. Moreover, if $A \in \mathcal{P}_r(S)$, then

$$F\big(\varphi_{r,m}(A)\big) = f\big(\varphi_{r,m}^{-1}(\varphi_{r,m}(A))\big) = f(A),$$

showing that F indeed extends the original f. □

2 Result: HyperPattern Recognition

2.1 Pattern Recognition

A Pattern Recognizer is a function mapping instances to labels by minimizing empirical loss on training data (cf. [27–31]). Pattern Recognition is a concept widely used across many domains, including machine learning [28, 32–34], artificial intelligence [35–37], and neural networks [38–40].

Definition 2.1 (Pattern Recognition)**.** (cf. [41–43]) Let X be a nonempty set of *input instances* and Y a nonempty set of *labels*. Suppose there is an unknown "ground-truth" function

$$g : X \longrightarrow Y,$$

and a training set of n labeled examples

$$D \;=\; \{(x_i, y_i) \mid x_i \in X,\ y_i = g(x_i),\ i = 1, \ldots, n\}.$$

A *pattern recognizer* is a function

$$h \;:\; X \longrightarrow Y$$

chosen from a hypothesis space $\mathcal{H}$ so as to *approximate* g. The quality of h is measured by a *loss function* $\ell : Y \times Y \to \mathbb{R}_{\geq 0}$. The goal of learning is to find

$$\hat{h} \;=\; \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell\big(h(x_i),\ y_i\big),$$

which under suitable conditions also minimizes the expected loss $\mathbb{E}_{x \sim P_X}[\ell(h(x), g(x))]$.

Example 2.2 (Email Spam Classification)**.** Let X be the set of all incoming email messages, and $Y = \{\text{spam, ham}\}$ the two possible labels. We collect a training set

$$D = \big\{(x_i, y_i) \mid x_i \in X,\ y_i \in \{\text{spam}, \text{ham}\},\ i = 1, \ldots, n\big\},$$

where each x_i is represented by a feature vector of token frequencies, header metadata, and known black-list indicators. A pattern recognizer

$$h \colon X \longrightarrow Y$$

can be implemented, for example, by a Naive Bayes classifier:

$$h(x) = \arg \max_{y \in \{\text{spam,ham}\}} P(y) \prod_{w \in V} P(w \mid y)^{\#_w(x)},$$

where V is the vocabulary, and $\#_w(x)$ the count of word w in email x. Training estimates $P(y)$ and $P(w \mid y)$ from D. The chosen h minimizes the empirical zero–one loss $\ell(h(x_i), y_i) = \mathbf{1}_{h(x_i) \neq y_i}$, and under standard assumptions also approximately minimizes the true error rate $\mathbb{E}_{x \sim P_X}[\mathbf{1}_{h(x) \neq g(x)}]$.

2.2 HyperPattern recognition

A HyperPattern Recognizer is a hyperfunction mapping instances to sets of candidate labels, optimizing nested set-loss over training data.

Definition 2.3 (HyperPattern Recognizer)**.** Let X be a nonempty set of input instances, Y a nonempty set of labels, and

$$D = \{(x_i, y_i) \mid x_i \in X,\ y_i \in Y,\ i = 1, \ldots, n\}$$

a training set. A *HyperPattern Recognizer* is a *hyperfunction*

$$H \ : \ X \longrightarrow \mathcal{P}(Y),$$

mapping each instance x to a *set* of labels $H(x) \subseteq Y$. To measure prediction quality, define a *set-loss*

$$L\big(A, y\big) = \min_{y' \in A} \ell\big(y', y\big), \qquad A \subseteq Y,\ y \in Y,$$

where $\ell \colon Y \times Y \to \mathbb{R}_{\geq 0}$ is the standard label-loss. The learning goal is to choose

$$\hat{H} \ = \ \arg\min_{H} \ \frac{1}{n} \sum_{i=1}^{n} L\big(H(x_i),\ y_i\big),$$

over a hypothesis class of hyperfunctions. This generalizes classical pattern recognition by allowing $H(x)$ to be non-singleton.

Example 2.4 (Image Classification with Top-5 Predictions)**.** Let X be the set of all input images and Y the finite set of object categories (e.g. $\{\text{cat}, \text{dog}, \ldots\}$). We collect a labeled dataset

$$D = \{(x_i, y_i) \mid x_i \in X,\ y_i \in Y,\ i = 1, \ldots, n\}.$$

A HyperPattern Recognizer $H \colon X \to \mathcal{P}(Y)$ is implemented by a neural network that estimates class probabilities $P(y \mid x; \theta)$. For each image x, define

$$H(x) = \big\{\, y \in Y \mid P(y \mid x; \theta) \text{ is among the top-5 values} \big\}.$$

Thus $H(x)$ is the set of the five most likely labels. Using the zero–one label-loss $\ell(y', y) = \mathbf{1}_{y' \neq y}$, the induced set-loss is

$$L\big(H(x), y\big) = \min_{y' \in H(x)} \mathbf{1}_{y' \neq y} = \begin{cases} 0, & y \in H(x), \\ 1, & y \notin H(x). \end{cases}$$

Training adjusts θ to minimize the empirical average $\frac{1}{n} \sum_{i=1}^{n} L\big(H(x_i), y_i\big)$, which corresponds exactly to maximizing the standard Top-5 accuracy used in practice.

Example 2.5 (Fingerprint-Based Identity Verification)**.** Let X be the set of fingerprint scans and Y the finite set of enrolled user identities. From a labeled database

$$D = \big\{(x_i, y_i) \mid x_i \in X,\ y_i \in Y,\ i = 1, \dots, n\big\}$$

we learn similarity scores $s(y \mid x)$ for each $y \in Y$. Define the HyperPattern Recognizer

$$H \colon X \longrightarrow \mathcal{P}(Y) \quad \text{by} \quad H(x) = \big\{\text{the top-3 identities } y \in Y \text{ maximizing } s(y \mid x)\big\}.$$

Thus $H(x)$ is the set of the three most likely matches for scan x. Using zero–one label-loss $\ell(y', y) = \mathbf{1}_{y' \neq y}$, the induced set-loss is

$$L\big(H(x), y\big) = \min_{y' \in H(x)} \mathbf{1}_{y' \neq y} = \begin{cases} 0, & y \in H(x), \\ 1, & y \notin H(x). \end{cases}$$

Training adjusts the matching model to minimize the empirical average $\frac{1}{n} \sum_{i=1}^{n} L\big(H(x_i), y_i\big)$, which corresponds in practice to optimizing the Top-3 identification accuracy.

Theorem 2.6 (HyperPattern Recognizer Generalizes Pattern Recognizer)**.** *Let $\mathcal{H}$ be a hypothesis class of ordinary functions $h \colon X \to Y$, and let $\mathcal{H}_{\text{hyper}}$ be the corresponding class of hyperfunctions $H \colon X \to \mathcal{P}(Y)$. Define the embedding*

$$\varphi \colon \mathcal{H} \longrightarrow \mathcal{H}_{\text{hyper}}, \quad \varphi(h)(x) = \{\, h(x)\}.$$

Then:

1. *φ is injective and $\varphi(h)$ is a valid hyperpattern recognizer.*
2. *For every $h \in \mathcal{H}$ and all (x, y), $L\big(\varphi(h)(x), y\big) = \ell\big(h(x), y\big)$.*
3. *Consequently, if $\hat{h} = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_i \ell(h(x_i), y_i)$, then $\varphi(\hat{h}) = \arg\min_{H \in \varphi(\mathcal{H})} \frac{1}{n} \sum_i L(H(x_i), y_i)$.*

Thus classical pattern recognition is recovered as a special case of HyperPattern Recognition.

Proof. **(1) Injectivity:** If $h_1 \neq h_2$ then there exists x with $h_1(x) \neq h_2(x)$, so $\varphi(h_1)(x) = \{h_1(x)\} \neq \{h_2(x)\} = \varphi(h_2)(x)$.

(2) Loss preservation: For any $x \in X$ and true label y,

$$L\big(\varphi(h)(x), y\big) = \min_{y' \in \{h(x)\}} \ell(y', y) = \ell\big(h(x), y\big).$$

(3) Optimality correspondence: Since φ preserves pointwise loss, the empirical risk of h equals that of $\varphi(h)$. Therefore the minimizer $\hat{h}$ of $\sum \ell$ maps to the minimizer $\varphi(\hat{h})$ of $\sum L$ restricted to $\varphi(\mathcal{H})$.

Because $\varphi(\mathcal{H}) \subseteq \mathcal{H}_{\text{hyper}}$, this shows HyperPattern Recognition truly generalizes the classical setting. □

2.3 SuperHyperPattern recognition

An (m, n)-SuperHyperPattern Recognizer maps the m-th iterated powerset of inputs to the n-th iterated powerset of labels, optimizing multilevel loss.

Definition 2.7 ((m, n)-SuperHyperPattern Recognizer)**.** Let X be a nonempty set of input instances and Y a nonempty set of labels. Fix integers $m, n \geq 0$. Denote by

$$\mathcal{P}_k(X),\ \mathcal{P}_k(Y) \quad (k = 0, 1, 2, \dots)$$

the k-fold iterated powersets of X and Y. A (m, n)-*SuperHyperPattern Recognizer* is a mapping

$$F \ : \ \mathcal{P}_m(X) \longrightarrow \mathcal{P}_n(Y)$$

selected from a hypothesis class $\mathcal{H}_{m,n} \subseteq \{\mathcal{P}_m(X) \to \mathcal{P}_n(Y)\}$. To evaluate predictions, define the *flattening operator*

$$\mathrm{flat}_0(y) = \{y\}, \quad \mathrm{flat}_{k+1}(A) = \bigcup_{B \in A} \mathrm{flat}_k(B), \quad A \subseteq \mathcal{P}_k(Y).$$

Given a label-loss $\ell : Y \times Y \to \mathbb{R}_{\geq 0}$, the induced *nested-set-loss* is

$$L_n(A, y) = \min_{y' \in \mathrm{flat}_n(A)} \ell(y', y), \quad A \in \mathcal{P}_n(Y),\ y \in Y.$$

A learning algorithm seeks

$$\widehat{F} = \arg\min_{F \in \mathcal{H}_{m,n}} \frac{1}{N} \sum_{i=1}^{N} L_n\big(F(\varphi_{0,m}(x_i)),\ y_i\big),$$

where the *canonical embedding* $\varphi_{0,0}(x) = x,\ \varphi_{0,k+1}(x) = \{\varphi_{0,k}(x)\}$ identifies X with $\mathrm{Im}(\varphi_{0,m}) \subseteq \mathcal{P}_m(X)$.

Example 2.8 (Hierarchical Document Topic Extraction)**.** Let X be the set of all sentences. Then

$$\mathcal{P}_1(X) = \{\text{paragraphs}\}, \quad \mathcal{P}_2(X) = \{\text{documents}\},$$

where each paragraph $P \subseteq X$ is a set of sentences and each document $D \subseteq \mathcal{P}_1(X)$ is a set of paragraphs. Let Y be the finite set of all possible topics, so

$$\mathcal{P}_1(Y) = \{\text{topic-sets for a paragraph}\}, \quad \mathcal{P}_2(Y) = \{\text{collections of paragraph topic-sets for a document}\}.$$

Assume we have a training corpus of documents labeled by ground-truth paragraph topics. First, train a HyperPattern Recognizer

$$H \colon \mathcal{P}_1(X) \to \mathcal{P}_1(Y), \quad H(P) = \{\text{topics occurring in paragraph } P\}.$$

Then define the $(2, 2)$-SuperHyperPattern Recognizer

$$F \colon \mathcal{P}_2(X) \to \mathcal{P}_2(Y), \quad F(D) = \big\{\, H(P) \mid P \in D \big\}.$$

Thus F maps each document D (a set of paragraphs) to the set of topic-sets for its paragraphs. Using the nested-set-loss $L_2(A, y) = \min_{y' \in \mathrm{flat}_2(A)} \ell(y', y)$, training adjusts H (and hence F) to minimize the average $\frac{1}{N} \sum_i L_2(F(D_i),\ y_i)$, thereby optimizing hierarchical topic extraction performance.

Example 2.9 (Customer Purchase Profiling)**.** Let X be the set of all products in an online store. Define

$$\mathcal{P}_1(X) = \{\text{shopping carts}\}, \quad \mathcal{P}_2(X) = \{\text{customer histories}\},$$

where each shopping cart $C \subseteq X$ is a set of products purchased in one session, and each customer history $H \subseteq \mathcal{P}_1(X)$ is the set of that customer's shopping carts. Let Y be the finite set of product categories (e.g. $\{\text{electronics}, \text{apparel}, \dots\}$). Then

$$\mathcal{P}_1(Y) = \{\text{category-sets per cart}\}, \quad \mathcal{P}_2(Y) = \{\text{sets of category-sets per customer}\}.$$

First train a HyperPattern Recognizer

$$G \colon \mathcal{P}_1(X) \longrightarrow \mathcal{P}_1(Y), \quad G(C) = \{\text{categories of products in cart } C\}.$$

Then define the $(2, 2)$-SuperHyperPattern Recognizer

$$F \colon \mathcal{P}_2(X) \longrightarrow \mathcal{P}_2(Y), \quad F(H) = \big\{\, G(C) \mid C \in H \big\}.$$

Thus F maps each customer history H to the set of category-sets from each of their carts. Using the nested-set-loss $L_2(A, y) = \min_{y' \in \mathrm{flat}_2(A)} \ell(y', y)$, training adjusts G (and hence F) to minimize the empirical average $\frac{1}{N} \sum_{i=1}^{N} L_2\big(F(H_i),\ y_i\big)$, where each $y_i \in Y$ is the primary category label for customer i. This approach captures both session-level and customer-level purchase patterns.

Example 2.10 (Multi-Scale Project Sentiment Modeling)**.** Let X be the set of all email messages in an organization. Then

$$\mathcal{P}_1(X) = \{\text{email threads}\}, \quad \mathcal{P}_2(X) = \{\text{projects}\},$$

where each project $D \subseteq \mathcal{P}_1(X)$ is the set of threads related to that project. Let $Y = \{\text{positive}, \text{neutral}, \text{negative}\}$ denote sentiment labels. Define

$$H_1 \colon \mathcal{P}_1(X) \to \mathcal{P}_1(Y), \quad H_1(T) = \{\text{sentiment}(m) \mid m \in T\},$$

so $H_1(T)$ is the set of all sentiments expressed within thread T. Next, for each calendar week w in a project's timeline, define

$$H_2 \colon \mathcal{P}_2(X) \times W \longrightarrow \mathcal{P}_2(Y), \quad H_2(D, w) = \big\{ H_1(T) \mid T \in D,\ \text{week}(T) = w \big\},$$

where W is the set of weeks during which D is active. Finally, the $(2,3)$-SuperHyperPattern Recognizer

$$F \colon \mathcal{P}_2(X) \longrightarrow \mathcal{P}_3(Y)$$

is given by

$$F(D) = \big\{ H_2(D, w) \mid w \in W \big\},$$

mapping each project D to the set of its weekly sentiment-sets $H_2(D, w)$. Using a nested-set-loss

$$L_3(A, y) = \min_{y' \in \text{flat}_3(A)} \ell(y', y), \quad A \in \mathcal{P}_3(Y),\ y \in Y,$$

training adjusts the underlying sentiment extractor sentiment$(\cdot)$ to minimize $\frac{1}{N} \sum_{i=1}^{N} L_3\big(F(D_i),\, y_i\big)$, where each $y_i \in Y$ is the ground-truth overall sentiment label for project D_i. This captures sentiment patterns at thread, week, and project levels.

Example 2.11 (Retail Cart Regional Category Mapping)**.** Let X be the set of all products in a retail chain. A *shopping cart* $C \subseteq X$ belongs to $\mathcal{P}_1(X)$. Let Y be the set of all product categories (e.g. $\{\text{Electronics}, \text{Apparel}, \dots\}$). We first extract the cart's categories:

$$H_1 \colon \mathcal{P}_1(X) \longrightarrow \mathcal{P}_1(Y), \qquad H_1(C) = \{\text{category}(x) \mid x \in C\}.$$

Next, let $\mathcal{D} \subseteq \mathcal{P}_1(Y)$ be *department clusters* (e.g. $\{\text{Phones}, \text{Computers}\} \subseteq Y$), $\mathcal{R} \subseteq \mathcal{P}_2(Y)$ be *region clusters* grouping departments, and $\mathcal{Q} \subseteq \mathcal{P}_3(Y)$ be *headquarter clusters* grouping regions. Define the $(1,3)$-SuperHyperPattern Recognizer

$$F \colon \mathcal{P}_1(X) \longrightarrow \mathcal{P}_3(Y), \qquad F(C) = \big\{ Q \in \mathcal{Q} \mid \exists\, R \in Q,\ \exists\, D \in R :\ D \cap H_1(C) \neq \emptyset \big\}.$$

Thus $F(C)$ returns all headquarter clusters Q that oversee any region R containing a department D matching one of the cart's categories. To train, one minimizes the nested-set-loss

$$L_3(A, y) = \min_{y' \in \text{flat}_3(A)} \ell(y', y), \quad A \in \mathcal{P}_3(Y),\ y \in Y,$$

thereby optimizing hierarchical regional mapping accuracy for each cart.

Theorem 2.12 (Generalization of Pattern and HyperPattern Recognizers)**.** *Let $\mathcal{H}_0 \subseteq \{X \to Y\}$ be a hypothesis class of ordinary pattern recognizers, and let $\mathcal{H}_{m,n} \subseteq \{\mathcal{P}_m(X) \to \mathcal{P}_n(Y)\}$ be the corresponding superhyper class. Define the embeddings*

$$\varphi_{0,k} : X \to \mathcal{P}_k(X), \quad \varphi_{0,0}(x) = x,\ \varphi_{0,k+1}(x) = \{\varphi_{0,k}(x)\},$$
$$\psi_{0,k} : Y \to \mathcal{P}_k(Y), \quad \psi_{0,0}(y) = y,\ \psi_{0,k+1}(y) = \{\psi_{0,k}(y)\}.$$

Then the map

$$\Phi : \mathcal{H}_0 \longrightarrow \mathcal{H}_{m,n}, \quad \Phi(h)(X) = \begin{cases} \psi_{0,n}\big(h(\varphi_{0,m}^{-1}(X))\big), & X \in \text{Im}(\varphi_{0,m}), \\ \emptyset, & \textit{otherwise}, \end{cases}$$

is injective and loss-preserving*: for every $(x, y) \in X \times Y$,*

$$L_n\big(\Phi(h)(\varphi_{0,m}(x)),\ y\big) = \ell\big(h(x),\ y\big).$$

Consequently, the optimal classical recognizer $\hat{h} = \arg\min_{h \in \mathcal{H}_0} \frac{1}{N} \sum_i \ell(h(x_i), y_i)$ corresponds under Φ to the optimal superhyper recognizer $\widehat{F} = \Phi(\hat{h})$ within $\Phi(\mathcal{H}_0) \subseteq \mathcal{H}_{m,n}$.

Proof. **Injectivity.** If $h_1 \neq h_2$ then there exists x with $h_1(x) \neq h_2(x)$, so $\psi_{0,n}(h_1(x)) \neq \psi_{0,n}(h_2(x))$, hence $\Phi(h_1) \neq \Phi(h_2)$.

Loss preservation. For any $x \in X$ and $y \in Y$,

$$L_n\big(\Phi(h)(\varphi_{0,m}(x)), y\big) = \min_{y' \in \mathrm{flat}_n(\{\psi_{0,n-1}(h(x))\})} \ell(y', y) = \ell\big(h(x), y\big),$$

since $\mathrm{flat}_n(\psi_{0,n}(h(x))) = \{h(x)\}$.

Optimality correspondence. Because Φ preserves the empirical risk, the minimizer $\hat{h}$ of $\sum_i \ell(h(x_i), y_i)$ maps to the minimizer $\Phi(\hat{h})$ of $\sum_i L_n(\Phi(h)(\varphi_{0,m}(x_i)), y_i)$ over $\Phi(\mathcal{H}_0)$. Thus classical and hyperpattern recognizers are recovered as special cases $(m, n) = (0, 0)$ and $(0, 1)$. □

Theorem 2.13 (Embedding of Classical Recognizers)**.** *Let $\mathcal{H}_0 \subseteq \{X \to Y\}$ be a class of ordinary pattern recognizers and let $\varphi_{0,m} \colon X \to \mathcal{P}_m(X)$, $\psi_{0,n} \colon Y \to \mathcal{P}_n(Y)$ be the canonical embeddings*

$$\varphi_{0,0}(x) = x,\ \varphi_{0,k+1}(x) = \{\varphi_{0,k}(x)\}, \quad \psi_{0,0}(y) = y,\ \psi_{0,k+1}(y) = \{\psi_{0,k}(y)\}.$$

Define

$$\Phi \colon \mathcal{H}_0 \longrightarrow \mathcal{H}_{m,n}, \quad \Phi(h)\big(X\big) = \begin{cases} \psi_{0,n}\big(h(\varphi_{0,m}^{-1}(X))\big), & X \in \mathrm{Im}(\varphi_{0,m}), \\ \emptyset, & \textit{otherwise.} \end{cases}$$

Then Φ is injective, and for every (x, y),

$$L_n\big(\Phi(h)(\varphi_{0,m}(x)), y\big) = \ell\big(h(x), y\big).$$

Thus classical pattern recognizers embed loss-preservingly into (m, n)-SuperHyperPattern recognizers.

Proof. Injectivity follows since if $h_1 \neq h_2$ then there exists x with $h_1(x) \neq h_2(x)$, so $\Phi(h_1)(\varphi_{0,m}(x)) = \psi_{0,n}(h_1(x)) \neq \psi_{0,n}(h_2(x)) = \Phi(h_2)(\varphi_{0,m}(x))$. Loss-preservation holds because

$$L_n\big(\Phi(h)(\varphi_{0,m}(x)), y\big) = \min_{y' \in \mathrm{flat}_n(\{\psi_{0,n-1}(h(x))\})} \ell(y', y) = \ell(h(x), y),$$

noting that $\mathrm{flat}_n(\psi_{0,n}(h(x))) = \{h(x)\}$. □

Theorem 2.14 (Composition Closure)**.** *Let*

$$F \in \mathcal{H}_{m,n}, \quad G \in \mathcal{H}_{n,p},$$

where

$$F \colon \mathcal{P}_m(X) \to \mathcal{P}_n(Y), \quad G \colon \mathcal{P}_n(Y) \to \mathcal{P}_p(Z).$$

Define the composition

$$G \circ F \ :\ \mathcal{P}_m(X) \xrightarrow{F} \mathcal{P}_n(Y) \xrightarrow{G} \mathcal{P}_p(Z).$$

Then $G \circ F \in \mathcal{H}_{m,p}$. Moreover, if L_p is the nested-set-loss on $\mathcal{P}_p(Z)$, it satisfies

$$L_p\big((G \circ F)(U),\, z\big) \ = \min_{z' \in \mathrm{flat}_p\left(G(F(U))\right)} \ell(z', z), \quad U \in \mathcal{P}_m(X),\ z \in Z.$$

Proof. By definition, $G \circ F$ maps $\mathcal{P}_m(X)$ into $\mathcal{P}_p(Z)$, so it lies in $\{\mathcal{P}_m(X) \to \mathcal{P}_p(Z)\} = \mathcal{H}_{m,p}$. The loss formula is immediate from the definition of nested-set-loss L_p. □

Theorem 2.15 (Flattening Reduction)**.** *Every (m, n)-SuperHyperPattern Recognizer $F \colon \mathcal{P}_m(X) \to \mathcal{P}_n(Y)$ induces a classical recognizer*

$$h_F \colon X \longrightarrow Y, \qquad h_F(x) = \arg \min_{y \in \mathrm{flat}_n\left(F(\varphi_{0,m}(x))\right)} \ell(y, y),$$

where ties may be broken arbitrarily. Moreover, for all (x, y),

$$\ell\big(h_F(x), y\big) = L_n\big(F(\varphi_{0,m}(x)),\ y\big).$$

Thus h_F has exactly the same pointwise loss as F.

Proof. Since $F(\varphi_{0,m}(x)) \subseteq \mathcal{P}_n(Y)$, flattening produces a nonempty set of labels $\mathrm{flat}_n(F(\varphi_{0,m}(x)))$. By definition of the set-loss,

$$L_n\big(F(\varphi_{0,m}(x)), y\big) = \min_{y' \in \mathrm{flat}_n(F(\varphi_{0,m}(x)))} \ell(y', y).$$

Choosing $h_F(x)$ to attain this minimum yields $\ell(h_F(x), y) = L_n(F(\varphi_{0,m}(x)), y)$. □

3 Conclusion

In this paper, we introduced and analyzed HyperPattern Recognition and SuperHyperPattern Recognition, which extend the classical Pattern Recognizer framework through the use of hyperfunctions and superhyperfunctions. In future work, we plan to explore further extensions that incorporate Graph Neural Networks [44–46], Fuzzy Sets [47–49], Intuitionistic Fuzzy Sets [50, 51], Rough Sets [52, 53], Plithogenic Sets [54, 55], Neutrosophic Sets [56–59], q-rung Orthopair Fuzzy Sets [60–63], and HyperFuzzy Sets [64–66]. We also anticipate the development of software libraries and programming tools to facilitate practical experimentation and application of these concepts.

Funding

This study did not receive any financial or external support from organizations or individuals.

Acknowledgments

We extend our sincere gratitude to everyone who provided insights, inspiration, and assistance throughout this research. We particularly thank our readers for their interest and acknowledge the authors of the cited works for laying the foundation that made our study possible. We also appreciate the support from individuals and institutions that provided the resources and infrastructure needed to produce and share this paper. Finally, we are grateful to all those who supported us in various ways during this project.

Data Availability

This research is purely theoretical, involving no data collection or analysis. We encourage future researchers to pursue empirical investigations to further develop and validate the concepts introduced here.

Ethical Approval

As this research is entirely theoretical in nature and does not involve human participants or animal subjects, no ethical approval is required.

Conflicts of Interest

The authors confirm that there are no conflicts of interest related to the research or its publication.

Disclaimer

This work presents theoretical concepts that have not yet undergone practical testing or validation. Future researchers are encouraged to apply and assess these ideas in empirical contexts. While every effort has been made to ensure accuracy and appropriate referencing, unintentional errors or omissions may still exist. Readers are advised to verify referenced materials on their own. The views and conclusions expressed here are the authors' own and do not necessarily reflect those of their affiliated organizations.

References

[1] Florentin Smarandache. Extension of hyperalgebra to superhyperalgebra and neutrosophic superhyperalgebra (revisited). In *International Conference on Computers Communications and Control*, pages 427–432. Springer, 2022.

[2] Florentin Smarandache. *SuperHyperFunction, SuperHyperStructure, Neutrosophic SuperHyperFunction and Neutrosophic SuperHyperStructure: Current understanding and future directions*. Infinite Study, 2023.

[3] GR Amiri, R Mousarezaei, and S Rahnama. Soft hyperstructures and their applications. *New Mathematics and Natural Computation*, pages 1–19, 2024.

[4] Gulay Oguz and Bijan Davvaz. Soft topological hyperstructure. *J. Intell. Fuzzy Syst.*, 40:8755–8764, 2021.

[5] T Vougiouklis. Fundamental relations in hv-structures. the'judging from the results' proof. *Journal of Algebraic Hyperstructures and Logical Algebras*, 1(1):21–36, 2020.

[6] Florentin Smarandache. Superhyperstructure & neutrosophic superhyperstructure, 2024. Accessed: 2024-12-01.

[7] Ajoy Kanti Das, Rajat Das, Suman Das, Bijoy Krishna Debnath, Carlos Granados, Bimal Shil, and Rakhal Das. A comprehensive study of neutrosophic superhyper bci-semigroups and their algebraic significance. *Transactions on Fuzzy Sets and Systems*, 8(2):80, 2025.

[8] Takaaki Fujita. Entropy reimagined: Theoretical foundations of hyperentropy and superhyperentropy with real-world and engineering applications. *Spectrum of Engineering and Management Sciences*, 3(1):262–286, 2025.

[9] Takaaki Fujita. Toward a unified framework for knot theory, hyperknot theory, and superhyperknot theory via superhyperstructures. *Neutrosophic Knowledge*, 6:55–71, 2025.

[10] Takaaki Fujita. An introduction and reexamination of hyperprobability and superhyperprobability: Comprehensive overview. *Asian Journal of Probability and Statistics*, 27(5):82–109, 2025.

[11] Takaaki Fujita. Knowledge superhypergraphs, multimodal superhypergraphs, lattice superhypergraphs, and hyperbolic superhypergraphs: Concepts and applications. *Journal of Operational and Strategic Analytics*, 3(2):95–119, 2025.

[12] Berrocal Villegas Salomón Marcos, Montalvo Fritas Willner, Berrocal Villegas Carmen Rosa, Flores Fuentes Rivera María Yissel, Espejo Rivera Roberto, Laura Daysi Bautista Puma, and Dante Manuel Macazana Fernández. Using plithogenic n-superhypergraphs to assess the degree of relationship between information skills and digital competencies. *Neutrosophic Sets and Systems*, 84:513–524, 2025.

[13] Min Huang, Fenghua Li, et al. Optimizing ai-driven digital resources in vocational english learning using plithogenic n-superhypergraph structures for adaptive content recommendation. *Neutrosophic Sets and Systems*, 88:283–295, 2025.

[14] Mohammad Hamidi, Florentin Smarandache, and Elham Davneshvar. Spectrum of superhypergraphs via flows. *Journal of Mathematics*, 2022(1):9158912, 2022.

[15] Abdullah Kargın and Memet Şahin. Superhyper groups and neutro–superhyper groups. *2023 Neutrosophic SuperHyperAlgebra And New Types of Topologies*, 25, 2023.

[16] Takaaki Fujita and Florentin Smarandache. *Neutrosophic TwoFold SuperhyperAlgebra and Anti SuperhyperAlgebra*. Infinite Study, 2025.

[17] Florentin Smarandache. *The SuperHyperFunction and the Neutrosophic SuperHyperFunction (revisited again)*, volume 3. Infinite Study, 2022.

[18] Maissam Jdid, Florentin Smarandache, and Takaaki Fujita. A linear mathematical model of the vocational training problem in a company using neutrosophic logic, hyperfunctions, and superhyperfunction. *Neutrosophic Sets and Systems*, 87:1–11, 2025.

[19] Yutong Song and Yong Deng. Entropic explanation of power set. *International Journal of Computers, Communications & Control*, 16(4):4413, 2021.

[20] Michael Rathjen. Constructive zermelo-fraenkel set theory, power set, and the calculus of constructions. In *Epistemology versus Ontology: Essays on the Philosophy and Foundations of Mathematics in Honour of Per Martin-Löf*, pages 313–349. Springer, 2012.

[21] Florentin Smarandache. Foundation of superhyperstructure & neutrosophic superhyperstructure. *Neutrosophic Sets and Systems*, 63(1):21, 2024.

[22] Marzieh Rahmati and Mohammad Hamidi. Extension of g-algebras to superhyper g-algebras. *Neutrosophic Sets and Systems*, 55:557–567, 2023.

[23] Adel Al-Odhari. A brief comparative study on hyperstructure, super hyperstructure, and n-super superhyperstructure. *Neutrosophic Knowledge*, 6:38–49, 2025.

[24] Takaaki Fujita. Extensions of multidirected graphs: Fuzzy, neutrosophic, plithogenic, rough, soft, hypergraph, and superhypergraph variants. *International Journal of Topology*, 2(3):11, 2025.

[25] Takaaki Fujita, Maisam Jdid, and Florentin Smarandache. Hyperfunctions and superhyperfunctions in linear programming: Foundations and applications. *International Journal of Neutrosophic Science*, 26(4):65–76, 2025.

[26] Takaaki Fujita. A theoretical exploration of hyperconcepts: Hyperfunctions, hyperrandomness, hyperdecision-making, and beyond (including a survey of hyperstructures). *Advancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond*, 344(498):111, 2025.

[27] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern recognition*. Elsevier, 2006.

[28] Anil K Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):4–37, 2000.

[29] Andrew R Webb. *Statistical pattern recognition*. John Wiley & Sons, 2003.

[30] Xu-Yao Zhang, Cheng-Lin Liu, and Ching Y Suen. Towards robust pattern recognition: A review. *Proceedings of the IEEE*, 108(6):894–922, 2020.

[31] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[32] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[33] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[34] Francy Shu and Jeff Shu. An eight-camera fall detection system using human fall pattern recognition via machine learning by a low-cost android box. *Scientific reports*, 11(1):2471, 2021.

[35] Chi Hau Chen. *Pattern recognition and artificial intelligence*. Elsevier, 2013.

[36] Tanzila Saba and Amjad Rehman. Effects of artificially intelligent tools on pattern recognition. *International Journal of machine learning and cybernetics*, 4(2):155–162, 2013.

[37] Jayanta Kumar Basu, Debnath Bhattacharyya, and Tai-hoon Kim. Use of artificial neural network in pattern recognition. *International journal of software engineering and its applications*, 4(2), 2010.

[38] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[39] Sergios Theodoridis and Konstantinos Koutroumbas. Pattern recognition and neural networks. In *Advanced course on artificial intelligence*, pages 169–195. Springer, 1999.

[40] Albert Nigrin. *Neural networks for pattern recognition*. MIT press, 1993.

[41] Danyang Li and Minghua Wu. Pattern recognition receptors in health and diseases. *Signal transduction and targeted therapy*, 6(1):291, 2021.

[42] Shilpa Rani, Kamlesh Lakhwani, and Sandeep Kumar. Three dimensional objects recognition & pattern recognition technique; related challenges: A review. *Multimedia Tools and Applications*, 81(12):17303–17346, 2022.

[43] Haixia Mei, Jingyi Peng, Tao Wang, Tingting Zhou, Hongran Zhao, Tong Zhang, and Zhi Yang. Overcoming the limits of cross-sensitivity: pattern recognition methods for chemiresistive gas sensor array. *Nano-micro letters*, 16(1):269, 2024.

[44] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565, 2019.

[45] Yue Gao, Yifan Feng, Shuyi Ji, and Rongrong Ji. Hgnn+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3181–3199, 2022.

[46] Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. *arXiv preprint arXiv:2105.00956*, 2021.

[47] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[48] Hans-Jürgen Zimmermann. *Fuzzy set theory—and its applications*. Springer Science & Business Media, 2011.

[49] Lotfi A Zadeh. A note on z-numbers. *Information sciences*, 181(14):2923–2932, 2011.

[50] Krassimir T Atanassov. *On intuitionistic fuzzy sets theory*, volume 283. Springer, 2012.

[51] Muhammad Akram, Bijan Davvaz, and Feng Feng. Intuitionistic fuzzy soft k-algebras. *Mathematics in Computer Science*, 7:353–365, 2013.

[52] Zdzis law Pawlak. *Rough sets: Theoretical aspects of reasoning about data*, volume 9. Springer Science & Business Media, 2012.

[53] Said Broumi, Florentin Smarandache, and Mamoni Dhar. Rough neutrosophic sets. *Infinite Study*, 32:493–502, 2014.

[54] Fazeelat Sultana, Muhammad Gulistan, Mumtaz Ali, Naveed Yaqoob, Muhammad Khan, Tabasam Rashid, and Tauseef Ahmed. A study of plithogenic graphs: applications in spreading coronavirus disease (covid-19) globally. *Journal of ambient intelligence and humanized computing*, 14(10):13139–13159, 2023.

[55] P Sathya, Nivetha Martin, and Florentine Smarandache. Plithogenic forest hypersoft sets in plithogenic contradiction based multi-criteria decision making. *Neutrosophic Sets and Systems*, 73:668–693, 2024.

[56] Said Broumi, Mohamed Talea, Assia Bakali, and Florentin Smarandache. Single valued neutrosophic graphs. *Journal of New theory*, (10):86–101, 2016.

[57] Antonios Paraskevas, Michael Madas, Florentin Smarandache, and Takaaki Fujita. Utility-based upside-down logic: A neutrosophic decision-making framework under uncertainty. *Journal of Decisions and Operations Research*, 2025.

[58] Muhammad Akram, Hafsa M Malik, Sundas Shahzadi, and Florentin Smarandache. Neutrosophic soft rough graphs with application. *Axioms*, 7(1):14, 2018.

[59] Said Broumi, Assia Bakali, Mohamed Talea, and Florentin Smarandache. An isolated bipolar single-valued neutrosophic graphs. In *Information Systems Design and Intelligent Applications: Proceedings of Fourth International Conference INDIA 2017*, pages 816–822. Springer, 2018.

[60] Muhammad Asif, Doha A Kattan, Dragan Pamučar, and Ghous Ali. q-rung orthopair fuzzy matroids with application to human trafficking. *Discrete Dynamics in Nature and Society*, 2021(1):8261118, 2021.

[61] Muhammad Akram, Samirah Alsulami, Faruk Karaaslan, and Ayesha Khan. q-rung orthopair fuzzy graphs under hamacher operators. *Journal of Intelligent & Fuzzy Systems*, 40(1):1367–1390, 2021.

[62] Anam Luqman, Muhammad Akram, and Ahmad N. Al-Kenani. q-rung orthopair fuzzy hypergraphs with applications. *Mathematics*, 7(3):260, 2019.

[63] Muhammad Akram, Sundas Shahzadi, Areen Rasool, and Musavarah Sarwar. Decision-making methods based on fuzzy soft competition hypergraphs. *Complex & Intelligent Systems*, 8(3):2325–2348, 2022.

[64] Young Bae Jun, Kul Hur, and Kyoung Ja Lee. Hyperfuzzy subalgebras of bck/bci-algebras. *Annals of Fuzzy Mathematics and Informatics*, 2017.

[65] Yong Lin Liu, Hee Sik Kim, and J. Neggers. Hyperfuzzy subsets and subgroupoids. *J. Intell. Fuzzy Syst.*, 33:1553–1562, 2017.

[66] Jayanta Ghosh and Tapas Kumar Samanta. Hyperfuzzy sets and hyperfuzzy group. *Int. J. Adv. Sci. Technol*, 41:27–37, 2012.