Review of Artificial Neural HyperNetworks and SuperHyperNetworks Based on Hypergraphs, SuperHypergraphs, and Powersets

Takaaki Fujita [1] *
[1] Independent Researcher, Shinjuku, Shinjuku-ku, Tokyo, Japan.

Abstract

Hypernetworks connect multiple nodes via hyperedges, modeling higher-order group relationships. SuperHyperNetworks build on this by drawing nodes and edges from nested powersets, capturing multi-level hierarchies with weighted links. A feedforward artificial neural network is a layered model in which neuron activations flow unidirectionally through weighted pairwise connections. In this paper, we introduce two extensions of this model using HyperGraph and SuperHyperGraph theory: the Artificial Neural HyperNetwork and the Artificial Neural SuperHyperNetwork. The Artificial Neural HyperNetwork replaces standard edges with weighted hyperedges, allowing activations to propagate nonlinearly through multi-neuron groupings across layers. The Artificial Neural SuperHyperNetwork further generalizes this approach by using nested superhyperedges to transmit activations through hierarchically grouped neurons over multiple layers.

Keywords: HyperGraph, SuperHyperGraph, Neural Network, Neural HyperNetwork, Neural SuperHyperNetwork

1 Preliminaries

In this section, we summarize the key definitions and notions that form the basis for the material discussed herein. Throughout this paper, unless otherwise noted, all structures are assumed to be finite, and the empty set is taken to be a subset of every set. Additionally, n will denote a nonnegative integer unless stated otherwise. For more in-depth treatments of any particular topic or operation, the reader is encouraged to consult the referenced sources.

1.1 SuperHyperGraph

Graph theory studies combinatorial properties and relationships of graphs—mathematical structures of vertices connected by edges [1, 2]. In classical graph theory, a hypergraph generalizes a standard graph by allowing edges—called hyperedges—to connect more than two vertices, thereby facilitating representation of higher-order relationships [3–7]. A *SuperHyperGraph* takes this idea further by incorporating iterated powerset constructions into the hypergraph framework. This notion has been proposed and analyzed in recent works [8–14]. Below, we lay out the associated definitions and related concepts.

Definition 1.1 (Base Set)**.** A *base set* S is the foundational set from which complex structures such as powersets and hyperstructures are derived. It is formally defined as:

$$S = \{x \mid x \text{ is an element within a specified domain}\}.$$

All elements in constructs like $\mathcal{P}(S)$ or $\mathcal{P}_n(S)$ originate from the elements of S.

Definition 1.2 (Powerset)**.** The *powerset* of a set S, denoted $\mathcal{P}(S)$, is the collection of all possible subsets of S, including both the empty set and S itself. Formally, it is expressed as:

$$\mathcal{P}(S) = \{A \mid A \subseteq S\}.$$

Definition 1.3 (n-th Powerset)**.** (cf. [10, 15])

The n-th powerset of a set H, denoted $P_n(H)$, is defined iteratively, starting with the standard powerset. The recursive construction is given by:

$$P_1(H) = P(H), \quad P_{n+1}(H) = P(P_n(H)), \quad \text{for } n \geq 1.$$

Similarly, the n-th non-empty powerset, denoted $P_n^*(H)$, is defined recursively as:

$$P_1^*(H) = P^*(H), \quad P_{n+1}^*(H) = P^*(P_n^*(H)).$$

Here, $P^*(H)$ represents the powerset of H with the empty set removed.

Definition 1.4 (Hypergraph)**.** [3, 16] A *hypergraph* $H = (V(H), E(H))$ consists of:

- A nonempty set $V(H)$ of vertices.
- A set $E(H)$ of hyperedges, where each hyperedge is a nonempty subset of $V(H)$, thereby allowing connections among multiple vertices.

Unlike standard graphs, hypergraphs are well-suited to represent higher-order relationships. In this paper, we restrict ourselves to the case where both $V(H)$ and $E(H)$ are finite.

Definition 1.5 (n-SuperHyperGraph)**.** [9, 17]

Let V_0 be a finite base set of vertices. For each integer $k \geq 0$, define the iterative powerset by

$$\mathcal{P}^0(V_0) = V_0, \quad \mathcal{P}^{k+1}(V_0) = \mathcal{P}(\mathcal{P}^k(V_0)),$$

where $\mathcal{P}(\cdot)$ denotes the usual powerset operation. An *n-SuperHyperGraph* is then a pair

$$\mathrm{SHT}^{(n)} = (V, E),$$

with

$$V \subseteq \mathcal{P}^n(V_0) \quad \text{and} \quad E \subseteq \mathcal{P}^n(V_0).$$

Each element of V is called an *n-supervertex* and each element of E an *n-superedge*.

1.2 HyperNetwork and SuperhyperNetwork

A hypernetwork is a network where edges (hyperedges) connect multiple nodes, enabling modeling of higher-order relationships, simultaneously capturing group interactions. A SuperHyperNetwork extends hypernetworks by using nested powerset-based nodes and edges, representing multi-level hierarchical group relationships with weighted dynamic connections. The definitions of HyperNetwork and SuperhyperNetwork are presented below [18].

Definition 1.6 (Hypernetwork)**.** [18, 19] A *hypernetwork* is an ordered triple

$$H = \big(V,\ \mathcal{E},\ w\big)$$

where

- V is a nonempty finite set of *nodes*;
- $\mathcal{E} \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$ is the set of *hyperedges*, each hyperedge $e \in \mathcal{E}$ being a nonempty subset of nodes (allowing multi-node interactions);
- $w : \mathcal{E} \to \mathbb{R}_{\geq 0}$ is a *weight or attribute function* on hyperedges (omitted if unweighted).

A *directed hypernetwork* may be defined by replacing $\mathcal{E} \subseteq \mathcal{P}(V)$ with a set of *ordered* tuples of nodes or by equipping each $e \in \mathcal{E}$ with a head-tail partition. One can further add a *node-labeling* $\ell_V : V \to L_V$ and a *hyperedge-labeling* $\ell_{\mathcal{E}} : \mathcal{E} \to L_{\mathcal{E}}$ to record types or properties.

Definition 1.7 (n-SuperHypernetwork)**.** [18] Let V_0 be a finite base set of *nodes*. Define the n-th iterated powerset recursively by

$$\mathcal{P}^0(V_0) = V_0, \qquad \mathcal{P}^{k+1}(V_0) = \mathcal{P}\big(\mathcal{P}^k(V_0)\big) \quad (k \geq 0).$$

An *n-superhypernetwork* is a tuple

$$\mathcal{N}^{(n)} = \big(V,\ \mathcal{E},\ w\big)$$

where

- $V \subseteq \mathcal{P}^n(V_0)$ is a finite set of *n-supernodes*;
- $\mathcal{E} \subseteq \mathcal{P}^n(V_0)$ is a finite set of *n-superedges*, each superedge $e \in \mathcal{E}$ being a nonempty subset of V;
- $w : \mathcal{E} \to \mathbb{R}_{\geq 0}$ is an optional *weight function* assigning a nonnegative real weight (or confidence) to each superedge.

In other words, both vertices and hyperedges of the network are drawn from the n-th powerset of the base node set, capturing up to n levels of hierarchical grouping.

1.3 Feedforward Artificial Neural Network

A feedforward artificial neural network is a layered model in which neurons propagate activations unidirectionally from input to output through weighted connections [20–29]. Related concepts include graph neural networks [30–36] and the like. The definition of a Feedforward Artificial Neural Network is provided below.

Definition 1.8 (Feedforward Artificial Neural Network)**.** A *feedforward artificial neural network* (ANN) is a function

$$f : \mathbb{R}^{n_0} \longrightarrow \mathbb{R}^{n_L}$$

determined by the following components:

- A positive integer L (the number of *layers* of the network).
- A sequence of positive integers $\{n_\ell\}_{\ell=0}^{L}$, where n_ℓ is the number of *units* (neurons) in layer ℓ. In particular, n_0 is the dimension of the input space and n_L is the dimension of the output space.
- For each layer $\ell = 1, 2, \ldots, L$:
  - A weight matrix
    $$W^{(\ell)} = \left[w_{ij}^{(\ell)} \right] \in \mathbb{R}^{n_\ell \times n_{\ell-1}},$$
    where $w_{ij}^{(\ell)}$ is the weight connecting the j-th neuron of layer $(\ell - 1)$ to the i-th neuron of layer ℓ.
  - A bias vector
    $$b^{(\ell)} = \left[b_i^{(\ell)} \right] \in \mathbb{R}^{n_\ell}, \quad \text{where } b_i^{(\ell)} \text{ is the bias of the } i\text{-th neuron in layer } \ell.$$
  - An *activation function*
    $$\sigma^{(\ell)} : \mathbb{R} \longrightarrow \mathbb{R} \quad \text{(applied component-wise)}.$$

The network computes its output via the following *feedforward* rules: given an input vector $x = a^{(0)} \in \mathbb{R}^{n_0}$, define for each layer $\ell = 1, 2, \ldots, L$

$$z^{(\ell)} = W^{(\ell)} a^{(\ell-1)} + b^{(\ell)} = \left[z_i^{(\ell)} \right] \in \mathbb{R}^{n_\ell}, \tag{1}$$

$$a^{(\ell)} = \sigma^{(\ell)}\left(z^{(\ell)} \right) = \left[\sigma^{(\ell)}(z_i^{(\ell)}) \right] \in \mathbb{R}^{n_\ell}, \tag{2}$$

where:

- $z^{(\ell)}$ is called the *pre-activation vector* for layer ℓ.
- $a^{(\ell)}$ is called the *activation vector* (output) of layer ℓ.
- The activation function $\sigma^{(\ell)}$ acts on each coordinate of $z^{(\ell)}$ independently; that is,
  $$\sigma^{(\ell)}(z^{(\ell)}) = \left(\sigma^{(\ell)}(z_1^{(\ell)}), \sigma^{(\ell)}(z_2^{(\ell)}), \ldots, \sigma^{(\ell)}(z_{n_\ell}^{(\ell)}) \right)^{\mathsf{T}}.$$

Finally, the output of the network is

$$f(x) = a^{(L)} = \sigma^{(L)}\left(W^{(L)} a^{(L-1)} + b^{(L)} \right).$$

Thus f is the composition

$$f(x) = \left(\sigma^{(L)} \circ \left(x \mapsto W^{(L)} x + b^{(L)} \right) \right) \circ \left(\sigma^{(L-1)} \circ \left(x \mapsto W^{(L-1)} x + b^{(L-1)} \right) \right) \circ \cdots \circ \left(\sigma^{(1)} \circ \left(x \mapsto W^{(1)} x + b^{(1)} \right) \right).$$

2 Result: Feedforward Artificial Neural HyperNetwork

A feedforward artificial neural hypernetwork integrates weighted hyperedge-based connections among neurons, propagating activations iteratively nonlinearly through multi-node groupings across layers.

Definition 2.1 (Feedforward Artificial Neural HyperNetwork)**.** A *feedforward artificial neural hypernetwork* (FANHyperNetwork) is a tuple

$$\mathcal{H} \;=\; \big(L,\; \{n_\ell\}_{\ell=0}^{L},\; \{\mathcal{E}^{(\ell)}\}_{\ell=1}^{L},\; \{w^{(\ell)}\}_{\ell=1}^{L},\; \{b^{(\ell)}\}_{\ell=1}^{L},\; \{\sigma^{(\ell)}\}_{\ell=1}^{L}\big),$$

where:

- $L \in \mathbb{Z}_{>0}$ is the total number of layers.
- For each $\ell = 0, 1, \ldots, L$, $n_\ell \in \mathbb{Z}_{>0}$ is the number of neurons in layer ℓ. In particular, n_0 is the dimension of the input space and n_L is the dimension of the output space.
- For each $\ell = 1, 2, \ldots, L$,
$$\mathcal{E}^{(\ell)} \;\subseteq\; \mathcal{P}\big(\{1, 2, \ldots, n_{\ell-1}\}\big) \setminus \{\emptyset\}$$
is a finite set of nonempty *hyperedges* at layer ℓ. Each hyperedge $e \in \mathcal{E}^{(\ell)}$ is a nonempty subset $e \subseteq \{1, 2, \ldots, n_{\ell-1}\}$ indicating a multi-input grouping from layer $\ell - 1$.
- For each $\ell = 1, 2, \ldots, L$,
$$w^{(\ell)} \colon \mathcal{E}^{(\ell)} \;\longrightarrow\; \mathbb{R}$$
is a *weight function* assigning a real weight $w_e^{(\ell)}$ to each hyperedge $e \in \mathcal{E}^{(\ell)}$.
- For each $\ell = 1, 2, \ldots, L$,
$$b^{(\ell)} \;=\; \big[b_i^{(\ell)}\big]_{i=1}^{n_\ell} \;\in\; \mathbb{R}^{n_\ell}$$
is the *bias vector* for layer ℓ. Here $b_i^{(\ell)}$ denotes the bias of the i-th neuron in layer ℓ.
- For each $\ell = 1, 2, \ldots, L$,
$$\sigma^{(\ell)} \colon \mathbb{R} \;\longrightarrow\; \mathbb{R}$$
is the *activation function* applied component-wise in layer ℓ.

Given an input vector $x = a^{(0)} \in \mathbb{R}^{n_0}$, the network propagates feedforward in layers $\ell = 1, \ldots, L$ as follows. Denote by $a^{(\ell-1)} = \big[a_j^{(\ell-1)}\big]_{j=1}^{n_{\ell-1}} \in \mathbb{R}^{n_{\ell-1}}$ the activations of layer $\ell - 1$. Then for each neuron $i = 1, 2, \ldots, n_\ell$ in layer ℓ, define its *pre-activation*

$$z_i^{(\ell)} \;=\; \sum_{e \in \mathcal{E}^{(\ell)} \colon i \text{ is target of } e} w_e^{(\ell)} \prod_{j \in e} a_j^{(\ell-1)} \;+\; b_i^{(\ell)}, \tag{3}$$

where "i is target of e" means that hyperedge e in layer ℓ contributes to neuron i. The *activation* of neuron i in layer ℓ is

$$a_i^{(\ell)} \;=\; \sigma^{(\ell)}\big(z_i^{(\ell)}\big).$$

Equivalently, writing $z^{(\ell)} = \big[z_i^{(\ell)}\big]_{i=1}^{n_\ell}$ and $a^{(\ell)} = \big[a_i^{(\ell)}\big]_{i=1}^{n_\ell}$, we have

$$z^{(\ell)} \;=\; \Big[z_i^{(\ell)}\Big] \;=\; \Big[\sum_{e \in \mathcal{E}^{(\ell)} \colon i \text{ target of } e} w_e^{(\ell)} \prod_{j \in e} a_j^{(\ell-1)} \;+\; b_i^{(\ell)}\Big], \quad a^{(\ell)} \;=\; \sigma^{(\ell)}\big(z^{(\ell)}\big),$$

where $\sigma^{(\ell)}$ acts entrywise on the vector $z^{(\ell)}$. The final output of the network is $f(x) = a^{(L)} \in \mathbb{R}^{n_L}$.

Example 2.2 (Simple Feedforward Artificial Neural HyperNetwork)**.** Consider a feedforward artificial neural hypernetwork $\mathcal{H}$ with the following specifications:

$$L = 2, \qquad n_0 = 2, \qquad n_1 = 2, \qquad n_2 = 1.$$

Thus the network has three layers: an input layer of dimension 2, one hidden layer of dimension 2, and an output layer of dimension 1.

Layer 1 (Input to Hidden). Let the input neurons be labeled $1, 2$. We define the hyperedges at layer 1 by

$$\mathcal{E}^{(1)} \;=\; \big\{\{1, 2\},\; \{1\}\big\}.$$

That is:

- Hyperedge $e_1^{(1)} = \{1, 2\}$ connects both input neurons to the first hidden neuron.
- Hyperedge $e_2^{(1)} = \{1\}$ connects only the first input neuron to the second hidden neuron.

Choose a weight function $w^{(1)} : \mathcal{E}^{(1)} \to \mathbb{R}$ and biases $b^{(1)} \in \mathbb{R}^2$ by

$$w^{(1)}_{\{1,2\}} = 0.8, \quad w^{(1)}_{\{1\}} = -0.5, \qquad b^{(1)} = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}.$$

Let the activation function on layer 1 be the ReLU:

$$\sigma^{(1)}(u) \;=\; \max\{\,0,\, u\}.$$

Layer 2 (Hidden to Output). Label the hidden-layer neurons by indices $1, 2$ (within layer 1). We define

$$\mathcal{E}^{(2)} \;=\; \big\{\{1\},\, \{1, 2\}\big\},$$

so that:

- Hyperedge $e_1^{(2)} = \{1\}$ connects the first hidden neuron (layer 1) to the output neuron.
- Hyperedge $e_2^{(2)} = \{1, 2\}$ connects both hidden neurons (layer 1) to the output neuron.

Choose weights and bias for layer 2 by

$$w^{(2)}_{\{1\}} = 1.2, \quad w^{(2)}_{\{1,2\}} = 0.6, \qquad b^{(2)} = \big[-0.3\big].$$

Let the activation function on layer 2 be the sigmoid:

$$\sigma^{(2)}(u) \;=\; \frac{1}{1 + e^{-u}}.$$

Feedforward Computation. Given an input vector $x = a^{(0)} = \big[a_1^{(0)},\, a_2^{(0)}\big]^T \in \mathbb{R}^2$, we compute layer 1 pre-activations and activations as follows:

$$z_1^{(1)} = w^{(1)}_{\{1,2\}}\,\big(a_1^{(0)} \cdot a_2^{(0)}\big) \;+\; b_1^{(1)} \;=\; 0.8\,a_1^{(0)}\,a_2^{(0)} \;+\; 0.1, \qquad a_1^{(1)} = \sigma^{(1)}\big(z_1^{(1)}\big) \;=\; \max\{0,\, 0.8\,a_1^{(0)}\,a_2^{(0)} + 0.1\},$$

$$z_2^{(1)} = w^{(1)}_{\{1\}}\,\big(a_1^{(0)}\big) \;+\; b_2^{(1)} \;=\; -0.5\,a_1^{(0)} \;+\; 0.2, \qquad a_2^{(1)} = \sigma^{(1)}\big(z_2^{(1)}\big) \;=\; \max\{0,\, -0.5\,a_1^{(0)} + 0.2\}.$$

Next, using these hidden activations $a^{(1)} = [\,a_1^{(1)},\, a_2^{(1)}\,]^T$, we compute layer 2:

$$\begin{aligned} z^{(2)} &= \underbrace{w^{(2)}_{\{1\}}\,\big(a_1^{(1)}\big)}_{\text{from hyperedge } \{1\}} \;+\; \underbrace{w^{(2)}_{\{1,2\}}\,\big(a_1^{(1)}\,a_2^{(1)}\big)}_{\text{from hyperedge } \{1,2\}} + b^{(2)} \\ &= 1.2\,a_1^{(1)} \;+\; 0.6\,\big(a_1^{(1)}\,a_2^{(1)}\big) \;-\; 0.3, \end{aligned} \qquad a^{(2)} \;=\; \sigma^{(2)}\big(z^{(2)}\big) \;=\; \frac{1}{1 + e^{-\,z^{(2)}}}.$$

Hence the final output of the network is $f(x) = a^{(2)} \in \mathbb{R}$.

Theorem 2.3. *The FANHyperNetwork defined above generalizes the standard feedforward artificial neural network.*

Proof. Recall that a *standard* feedforward ANN (Definition in classical form) uses, for each layer ℓ, a *weight matrix* $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and bias vector $b^{(\ell)} \in \mathbb{R}^{n_\ell}$, with the linear step

$$z_i^{(\ell)} \;=\; \sum_{j=1}^{n_{\ell-1}} W_{i,j}^{(\ell)}\, a_j^{(\ell-1)} \;+\; b_i^{(\ell)}, \quad a_i^{(\ell)} = \sigma^{(\ell)}\big(z_i^{(\ell)}\big).$$

To recover this from the FANHyperNetwork formulation, choose for each ℓ:

$$\mathcal{E}^{(\ell)} \;=\; \big\{\, \{j\} : j = 1, 2, \ldots, n_{\ell-1} \big\},$$

i.e. each hyperedge is a singleton $\{j\}$. Define

$$w_{\{j\}}^{(\ell)} \;=\; W_{i,j}^{(\ell)} \quad \text{and} \quad b_i^{(\ell)} \text{ as in the standard ANN.}$$

Then in formula (3), the product over $j \in e$ (which is a singleton) becomes $\prod_{j\in\{j\}} a_j^{(\ell-1)} = a_j^{(\ell-1)}$. Hence

$$z_i^{(\ell)} \;=\; \sum_{e=\{j\}\in\mathcal{E}^{(\ell)}:\, i \text{ target of } e} w_e^{(\ell)}\, a_j^{(\ell-1)} \;+\; b_i^{(\ell)} \;=\; \sum_{j=1}^{n_{\ell-1}} W_{i,j}^{(\ell)}\, a_j^{(\ell-1)} \;+\; b_i^{(\ell)}.$$

Thus the FANHyperNetwork reduces exactly to the standard feedforward ANN. Therefore, the standard ANN is a special case of the FANHyperNetwork. □

Definition 2.4 (Underlying Neuron HyperNetwork)**.** Let $\mathcal{H} = \big(L, \{n_\ell\}, \{\mathcal{E}^{(\ell)}\}, \{w^{(\ell)}\}, \{b^{(\ell)}\}, \{\sigma^{(\ell)}\}\big)$ be a FANHyperNetwork. Define its *underlying neuron hypernetwork* as the ordered triple

$$H_{\text{neurons}} \;=\; \big(V,\, \mathcal{E},\, w_{\text{all}}\big),$$

where:

- $V = \bigsqcup_{\ell=0}^{L} \{\, (\ell, i) \mid i = 1, 2, \ldots, n_\ell \}$ is the disjoint union of neuron-indices across all layers, so that each neuron in layer ℓ is labeled by (ℓ, i).
- $\mathcal{E} = \bigsqcup_{\ell=1}^{L} \big\{\, e_\ell \times \{(\ell,i)\} \;\big|\; e_\ell \in \mathcal{E}^{(\ell)},\ i = 1, 2, \ldots, n_\ell \big\}$, where

  $$e_\ell \times \{(\ell,i)\} \;=\; \big\{\, (\ell-1, j) \;\big|\; j \in e_\ell \big\} \;\cup\; \{\, (\ell,i)\}$$

  is the hyperedge in H_{neurons} that connects all neurons $\{(\ell-1, j) \mid j \in e_\ell\}$ in layer $\ell-1$ to the single neuron (ℓ, i) in layer ℓ.
- The weight function $w_{\text{all}} \colon \mathcal{E} \to \mathbb{R}$ is defined by

  $$w_{\text{all}}\big(e_\ell \times \{(\ell,i)\}\big) \;:=\; w_{e_\ell}^{(\ell)}, \quad \text{for each } e_\ell \times \{(\ell,i)\} \in \mathcal{E}.$$

Theorem 2.5. *The underlying neuron structure H_{neurons} of a FANHyperNetwork is a (directed) HyperNetwork in the sense of Definition 1 (Hypernetwork).*

Proof. We check the three components required by Definition 1 for $H_{\text{neurons}} = (V, \mathcal{E}, w_{\text{all}})$:

1. V is a nonempty finite set of nodes, since there are finitely many layers $\ell = 0, \ldots, L$, each with a finite number n_ℓ of neurons. Hence $V = \bigsqcup_{\ell=0}^{L} \{(\ell, i) \mid 1 \le i \le n_\ell\}$ is finite and nonempty.
2. $\mathcal{E} \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$. Indeed, each hyperedge in $\mathcal{E}$ has the form $e_\ell \times \{(\ell,i)\} = \{\, (\ell-1, j) \mid j \in e_\ell\} \cup \{(\ell,i)\}$, which is a nonempty subset of V. Moreover, $\mathcal{E}$ is finite because each $\mathcal{E}^{(\ell)}$ is finite and there are finitely many layers $\ell = 1, \ldots, L$ and neurons $i = 1, \ldots, n_\ell$. Thus $\mathcal{E} \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$.
3. The function $w_{\text{all}} \colon \mathcal{E} \to \mathbb{R}$ assigns each hyperedge $e_\ell \times \{(\ell,i)\}$ the real weight $w_{e_\ell}^{(\ell)}$. Hence w_{all} is a valid *weight function* on hyperedges.

Since all the requirements of Definition are satisfied, H_{neurons} is a (directed) hypernetwork whose nodes are the network's neurons and whose hyperedges encode each layer's multi-input connections. Therefore, the FANHyperNetwork indeed carries the structure of a hypernetwork on its neurons. □

3 Result: Feedforward Artificial Neural SuperHyperNetwork

A feedforward artificial neural superhypernetwork employs nested iterated-superhyperedge connections to propagate activations complexly through hierarchically grouped neurons across multiple layers.

Definition 3.1 (Feedforward Artificial Neural n-SuperHyperNetwork)**.** A *feedforward artificial neural n-superhypernetwork* (FANnSHN) is a tuple

$$\mathcal{H}^{(n)} \;=\; \Big(L,\ \{n_\ell\}_{\ell=0}^{L},\ \{\mathcal{E}^{(\ell,n)}\}_{\ell=1}^{L},\ \{w^{(\ell,n)}\}_{\ell=1}^{L},\ \{b^{(\ell)}\}_{\ell=1}^{L},\ \{\sigma^{(\ell)}\}_{\ell=1}^{L}\Big),$$

where:

- $L \in \mathbb{Z}_{>0}$ is the total number of layers.
- For each $\ell = 0, 1, \ldots, L$, $n_\ell \in \mathbb{Z}_{>0}$ is the number of neurons in layer ℓ. In particular, n_0 is the dimension of the input space and n_L is the dimension of the output space.
- For each layer $\ell = 1, 2, \ldots, L$:
  - Let $V^{(\ell-1)} = \{1, 2, \ldots, n_{\ell-1}\}$ denote the set of neuron-indices in layer $\ell - 1$. Define the n-fold iterated powerset recursively by
  $$\mathcal{P}^0\big(V^{(\ell-1)}\big) \;=\; V^{(\ell-1)}, \qquad \mathcal{P}^{k+1}\big(V^{(\ell-1)}\big) \;=\; \mathcal{P}\Big(\mathcal{P}^k\big(V^{(\ell-1)}\big)\Big) \quad (k \geq 0).$$
  - $\mathcal{E}^{(\ell,n)} \;\subseteq\; \mathcal{P}^n\big(V^{(\ell-1)}\big) \setminus \{\emptyset\}$ is a finite set of nonempty *n-superhyperedges* at layer ℓ. Each n-superhyperedge $e \in \mathcal{E}^{(\ell,n)}$ is a nonempty subset $e \subseteq \mathcal{P}^{n-1}\big(V^{(\ell-1)}\big)$, whose elements are themselves $(n-1)$-supernodes.
  - $w^{(\ell,n)} \colon \mathcal{E}^{(\ell,n)} \longrightarrow \mathbb{R}$ is a *weight function* assigning a real scalar $w_e^{(\ell,n)}$ to each n-superhyperedge e.
  - $b^{(\ell)} = \big[b_i^{(\ell)}\big]_{i=1}^{n_\ell} \in \mathbb{R}^{n_\ell}$ is the bias vector for layer ℓ, where $b_i^{(\ell)}$ is the bias of the i-th neuron in layer ℓ.
  - $\sigma^{(\ell)} \colon \mathbb{R} \longrightarrow \mathbb{R}$ is the activation function applied componentwise in layer ℓ.

Given an input $x = a^{(0)} \in \mathbb{R}^{n_0}$, the activations propagate layer-by-layer. Suppose at layer $\ell - 1$ we have activation values $a^{(\ell-1)} = \big[a_j^{(\ell-1)}\big]_{j=1}^{n_{\ell-1}} \in \mathbb{R}^{n_{\ell-1}}$. We define recursively a *value function* $\mathrm{Val}\big(\cdot\,;\, a^{(\ell-1)}\big)$ on $\mathcal{P}^k\big(V^{(\ell-1)}\big)$ for $0 \leq k \leq n$ by:

$$\mathrm{Val}\big(j\,;\, a^{(\ell-1)}\big) \;=\; a_j^{(\ell-1)}, \quad \text{for each } j \in V^{(\ell-1)};$$

and for any nonempty $S \in \mathcal{P}^k\big(V^{(\ell-1)}\big)$ with $k \geq 1$,

$$\mathrm{Val}\big(S\,;\, a^{(\ell-1)}\big) \;=\; \prod_{T \in S} \mathrm{Val}\big(T\,;\, a^{(\ell-1)}\big).$$

Thus if $k = 1$, then $S \subseteq V^{(\ell-1)}$ and $\mathrm{Val}(S) = \prod_{j \in S} a_j^{(\ell-1)}$. If $k > 1$, each $T \in S$ is itself a $(k-1)$-supernode and we multiply its recursively defined value.

For each neuron $i = 1, 2, \ldots, n_\ell$ in layer ℓ, its *pre-activation* is

$$z_i^{(\ell)} \;=\; \sum_{\substack{e \in \mathcal{E}^{(\ell,n)}:\\ i \text{ is target of } e}} w_e^{(\ell,n)}\ \mathrm{Val}\big(e\,;\, a^{(\ell-1)}\big) \;+\; b_i^{(\ell)}. \tag{4}$$

The corresponding *activation* is

$$a_i^{(\ell)} \;=\; \sigma^{(\ell)}\big(z_i^{(\ell)}\big).$$

Equivalently, letting $z^{(\ell)} = \big[z_i^{(\ell)}\big]_{i=1}^{n_\ell}$ and $a^{(\ell)} = \big[a_i^{(\ell)}\big]_{i=1}^{n_\ell}$, we write

$$z^{(\ell)} \;=\; \Big[z_i^{(\ell)}\Big]_{i=1}^{n_\ell} \;=\; \Big[\sum_{\substack{e \in \mathcal{E}^{(\ell,n)}:\\ i \text{ is target of } e}} w_e^{(\ell,n)}\,\mathrm{Val}\big(e\,;\, a^{(\ell-1)}\big) \;+\; b_i^{(\ell)}\Big]_{i=1}^{n_\ell}, \quad a^{(\ell)} \;=\; \sigma^{(\ell)}\big(z^{(\ell)}\big),$$

where $\sigma^{(\ell)}$ acts entrywise. The final output of the network is $f(x) = a^{(L)} \in \mathbb{R}^{n_L}$.

Example 3.2 (Feedforward Artificial Neural 2-SuperHyperNetwork)**.** Consider a feedforward artificial neural 2-superhypernetwork $\mathcal{H}^{(2)}$ with:

$$L = 2, \qquad n_0 = 2, \qquad n_1 = 2, \qquad n_2 = 1.$$

Thus there are three layers: input (dimension 2), one hidden layer (dimension 2), and output (dimension 1). We set $n = 2$.

Layer 1 (Input → Hidden). Label the input-layer neuron indices by $V^{(0)} = \{1, 2\}$. Then

$$\mathcal{P}^1\big(V^{(0)}\big) \;=\; \big\{\{1\},\, \{2\},\, \{1,2\}\big\}, \quad \mathcal{P}^2\big(V^{(0)}\big) \;=\; \mathcal{P}\big(\mathcal{P}^1(V^{(0)})\big).$$

Explicitly,

$$\mathcal{P}^2\big(V^{(0)}\big) \;=\; \big\{\{\{1\}\},\, \{\{2\}\},\, \{\{1,2\}\},\, \{\{1\},\{2\}\},\, \{\{1\},\{1,2\}\},\, \{\{2\},\{1,2\}\},\, \{\{1\},\{2\},\{1,2\}\}\big\}.$$

We choose two 2-superhyperedges in layer 1:

$$e_1^{(1,2)} = \big\{\{1\},\, \{2\}\big\}, \qquad e_2^{(1,2)} = \big\{\{1,2\}\big\}.$$

Thus:

- $e_1^{(1,2)} \subseteq \mathcal{P}^1(V^{(0)})$ has elements $\{1\}$ and $\{2\}$.
- $e_2^{(1,2)} \subseteq \mathcal{P}^1(V^{(0)})$ has the single element $\{1, 2\}$.

Define weights and biases for layer 1:

$$w^{(1,2)}_{e_1^{(1,2)}} = 0.5, \quad w^{(1,2)}_{e_2^{(1,2)}} = -0.7, \qquad b^{(1)} = \begin{bmatrix} 0.1 \\ -0.2 \end{bmatrix}.$$

Let the activation function on layer 1 be $\sigma^{(1)}(u) = \max\{0, u\}$ (ReLU).

Layer 2 (Hidden → Output). Label the hidden-layer neuron indices by $V^{(1)} = \{1, 2\}$. Then

$$\mathcal{P}^1\big(V^{(1)}\big) \;=\; \big\{\{1\},\, \{2\},\, \{1,2\}\big\}, \quad \mathcal{P}^2\big(V^{(1)}\big) \;=\; \mathcal{P}\big(\mathcal{P}^1(V^{(1)})\big).$$

We choose two 2-superhyperedges in layer 2:

$$e_1^{(2,2)} = \big\{\{1\}\big\}, \qquad e_2^{(2,2)} = \big\{\{1\},\, \{2\}\big\}.$$

Define weights and bias for layer 2:

$$w^{(2,2)}_{e_1^{(2,2)}} = 1.0, \quad w^{(2,2)}_{e_2^{(2,2)}} = 0.4, \qquad b^{(2)} = \big[0.0\big].$$

Let the activation function on layer 2 be the sigmoid $\sigma^{(2)}(u) = \dfrac{1}{1 + e^{-u}}$.

Value Function Definitions. At layer 1, given activations $a^{(0)} = \big[a_1^{(0)},\, a_2^{(0)}\big]^T$, we define

$$\mathrm{Val}\big(\{1\};\, a^{(0)}\big) = a_1^{(0)}, \quad \mathrm{Val}\big(\{2\};\, a^{(0)}\big) = a_2^{(0)}, \quad \mathrm{Val}\big(\{1,2\};\, a^{(0)}\big) = a_1^{(0)}\, a_2^{(0)}.$$

Then for each 2-supernode $S \subseteq \mathcal{P}^1(V^{(0)})$,

$$\mathrm{Val}\big(S;\, a^{(0)}\big) \;=\; \prod_{T \in S} \mathrm{Val}\big(T;\, a^{(0)}\big).$$

Hence:

- $\mathrm{Val}\big(e_1^{(1,2)};\, a^{(0)}\big) = \mathrm{Val}(\{1\};\, a^{(0)}) \cdot \mathrm{Val}(\{2\};\, a^{(0)}) = a_1^{(0)}\, a_2^{(0)}$.
- $\mathrm{Val}\big(e_2^{(1,2)};\, a^{(0)}\big) = \mathrm{Val}(\{1,2\};\, a^{(0)}) = a_1^{(0)}\, a_2^{(0)}$.

Layer 1 Computation. For hidden neuron $i = 1, 2$,

$$z_i^{(1)} \;=\; \sum_{\substack{e \in \mathcal{E}^{(1,2)}:\\ i \text{ is target of } e}} w_e^{(1,2)}\, \mathrm{Val}\big(e;\, a^{(0)}\big) \;+\; b_i^{(1)}.$$

Assume both hyperedges $e_1^{(1,2)}$ and $e_2^{(1,2)}$ target each hidden neuron i. Then

$$z_1^{(1)} = 0.5\,(a_1^{(0)}\, a_2^{(0)}) \;-\; 0.7\,(a_1^{(0)}\, a_2^{(0)}) \;+\; 0.1 \;=\; (-0.2)\, a_1^{(0)}\, a_2^{(0)} \;+\; 0.1,$$
$$z_2^{(1)} = 0.5\,(a_1^{(0)}\, a_2^{(0)}) \;-\; 0.7\,(a_1^{(0)}\, a_2^{(0)}) \;-\; 0.2 \;=\; (-0.2)\, a_1^{(0)}\, a_2^{(0)} \;-\; 0.2.$$

Thus

$$a_i^{(1)} = \sigma^{(1)}\big(z_i^{(1)}\big) = \max\big\{0,\; (-0.2)\, a_1^{(0)}\, a_2^{(0)} + b_i^{(1)}\big\}.$$

Layer 2 Computation. Given $a^{(1)} = [\, a_1^{(1)},\, a_2^{(1)}\,]^T$, we define

$$\mathrm{Val}\big(\{1\};\, a^{(1)}\big) = a_1^{(1)}, \quad \mathrm{Val}\big(\{2\};\, a^{(1)}\big) = a_2^{(1)}, \quad \mathrm{Val}\big(\{1\},\{2\};\, a^{(1)}\big) = a_1^{(1)}\, a_2^{(1)}.$$

For the single output neuron $i = 1$,

$$z^{(2)} \;=\; \sum_{\substack{e \in \mathcal{E}^{(2,2)}:\\ 1 \text{ is target of } e}} w_e^{(2,2)}\, \mathrm{Val}\big(e;\, a^{(1)}\big) \;+\; b^{(2)}.$$

Assume both $e_1^{(2,2)}$ and $e_2^{(2,2)}$ contribute to the output. Then

$$z^{(2)} = 1.0\, a_1^{(1)} \;+\; 0.4\,(a_1^{(1)}\, a_2^{(1)}) \;+\; 0 = a_1^{(1)} \;+\; 0.4\, a_1^{(1)}\, a_2^{(1)}.$$

Finally,

$$a^{(2)} = \sigma^{(2)}\big(z^{(2)}\big) = \frac{1}{1 + e^{-\left(a_1^{(1)} + 0.4\, a_1^{(1)}\, a_2^{(1)}\right)}}.$$

Hence the network's output is $f(x) = a^{(2)} \in \mathbb{R}$.

Example 3.3 (Feedforward Artificial Neural 3-SuperHyperNetwork: Medical Triage System)**.** Consider a medical triage application for early sepsis detection using three vital-sign inputs:

$$\text{Heart Rate (HR)}, \quad \text{Blood Pressure (BP)}, \quad \text{Body Temperature (Temp)}.$$

We build a feedforward artificial neural 3-superhypernetwork $\mathcal{H}^{(3)}$ with:

$$L = 2, \qquad n_0 = 3, \qquad n_1 = 2, \qquad n_2 = 1, \quad n = 3.$$

Thus there are three layers: input (dimension 3), one hidden layer (dimension 2), and output (dimension 1).

Layer 1 (Input → Hidden). Label the input-layer neuron indices by

$$V^{(0)} \;=\; \{\,1,\, 2,\, 3\},$$

where 1 corresponds to HR, 2 to BP, and 3 to Temp. We compute:

$$\mathcal{P}^1\big(V^{(0)}\big) \;=\; \big\{\{1\},\, \{2\},\, \{3\},\, \{1,2\},\, \{1,3\},\, \{2,3\},\, \{1,2,3\}\big\},$$
$$\mathcal{P}^2\big(V^{(0)}\big) \;=\; \big\{\{\{1\}\},\, \{\{2\}\},\, \{\{3\}\},\, \{\{1,2\}\},\, \{\{1,3\}\},\, \{\{2,3\}\},\, \{\{1,2,3\}\},\, \{\{1\},\{2\}\},\, \ldots,\, \{\{1\},\{2\},\{3\},\{1,2,3\}\}\big\},$$
$$\mathcal{P}^3\big(V^{(0)}\big) \;=\; \mathcal{P}\Big(\mathcal{P}^2\big(V^{(0)}\big)\Big).$$

We select two 3-superhyperedges in layer 1 as follows:

$$e_1^{(1,3)} \;=\; \big\{\{\{1\}\},\, \{\{2\}\},\, \{\{3\}\}\big\}, \qquad e_2^{(1,3)} \;=\; \big\{\{\{1,2\}\}\big\}.$$

Here:

- $e_1^{(1,3)} \subseteq \mathcal{P}^2(V^{(0)})$ has the three 2-supernodes $\{\{1\}\}$, $\{\{2\}\}$, $\{\{3\}\}$, each isolating a single vital sign.
- $e_2^{(1,3)} \subseteq \mathcal{P}^2(V^{(0)})$ has the single 2-supernode $\{\{1,2\}\}$, representing the combined interaction of Heart Rate and Blood Pressure.

Define weights and biases for layer 1:

$$w^{(1,3)}_{e_1^{(1,3)}} = 0.8, \quad w^{(1,3)}_{e_2^{(1,3)}} = -0.5, \qquad b^{(1)} = \begin{bmatrix} 0.2 \\ -0.1 \end{bmatrix}.$$

Let the activation function on layer 1 be ReLU:

$$\sigma^{(1)}(u) \;=\; \max\{0,\, u\}.$$

Layer 2 (Hidden → Output). Label the hidden-layer neuron indices by

$$V^{(1)} \;=\; \{\,1,\,2\}.$$

Then

$$\mathcal{P}^1\big(V^{(1)}\big) = \{\{1\},\,\{2\},\,\{1,2\}\}, \quad \mathcal{P}^2\big(V^{(1)}\big) = \mathcal{P}\big(\mathcal{P}^1(V^{(1)})\big), \quad \mathcal{P}^3\big(V^{(1)}\big) = \mathcal{P}\big(\mathcal{P}^2(V^{(1)})\big).$$

We choose two 3-superhyperedges in layer 2:

$$e_1^{(2,3)} = \big\{\{\{1\}\}\big\}, \qquad e_2^{(2,3)} = \big\{\{\{1\}\},\,\{\{2\}\}\big\}.$$

Define weights and bias for layer 2:

$$w^{(2,3)}_{e_1^{(2,3)}} = 1.2, \quad w^{(2,3)}_{e_2^{(2,3)}} = 0.6, \qquad b^{(2)} = \big[0.0\big].$$

Let the activation function on layer 2 be sigmoid:

$$\sigma^{(2)}(u) \;=\; \frac{1}{1+e^{-u}}.$$

Value Function Definitions (Layer 1). At layer 1, given input activations $a^{(0)} = \big[a_1^{(0)},\, a_2^{(0)},\, a_3^{(0)}\big]^T$, define:

$$\mathrm{Val}\big(\{\{1\}\};\, a^{(0)}\big) = a_1^{(0)}, \quad \mathrm{Val}\big(\{\{2\}\};\, a^{(0)}\big) = a_2^{(0)}, \quad \mathrm{Val}\big(\{\{3\}\};\, a^{(0)}\big) = a_3^{(0)},$$

$$\mathrm{Val}\big(\{\{1,2\}\};\, a^{(0)}\big) = \mathrm{Val}\big(\{1,2\};\, a^{(0)}\big) = a_1^{(0)}\, a_2^{(0)}.$$

Then for any 2-supernode $T \subseteq \mathcal{P}^1(V^{(0)})$,

$$\mathrm{Val}\big(T;\, a^{(0)}\big) = \prod_{S \in T} \mathrm{Val}\big(S;\, a^{(0)}\big).$$

Finally, for each 3-supernode $E \subseteq \mathcal{P}^2(V^{(0)})$,

$$\mathrm{Val}\big(E;\, a^{(0)}\big) = \prod_{T \in E} \mathrm{Val}\big(T;\, a^{(0)}\big).$$

Hence:

$$\mathrm{Val}\big(e_1^{(1,3)};\, a^{(0)}\big) = \mathrm{Val}\big(\{\{1\}\};\, a^{(0)}\big)\,\mathrm{Val}\big(\{\{2\}\};\, a^{(0)}\big)\,\mathrm{Val}\big(\{\{3\}\};\, a^{(0)}\big) = a_1^{(0)}\, a_2^{(0)}\, a_3^{(0)},$$

$$\mathrm{Val}\big(e_2^{(1,3)};\, a^{(0)}\big) = \mathrm{Val}\big(\{\{1,2\}\};\, a^{(0)}\big) = a_1^{(0)}\, a_2^{(0)}.$$

Layer 1 Computation. For each hidden neuron $i = 1, 2$,

$$z_i^{(1)} = \sum_{\substack{e \in \mathcal{E}^{(1,3)}: \\ i \text{ is target of } e}} w_e^{(1,3)} \operatorname{Val}\big(e;\, a^{(0)}\big) \;+\; b_i^{(1)}.$$

Assume both $e_1^{(1,3)}$ and $e_2^{(1,3)}$ target each hidden neuron. Then

$$z_1^{(1)} = 0.8\,(a_1^{(0)} a_2^{(0)} a_3^{(0)}) \;-\; 0.5\,(a_1^{(0)} a_2^{(0)}) \;+\; 0.2,$$
$$z_2^{(1)} = 0.8\,(a_1^{(0)} a_2^{(0)} a_3^{(0)}) \;-\; 0.5\,(a_1^{(0)} a_2^{(0)}) \;-\; 0.1.$$

Thus

$$a_i^{(1)} = \sigma^{(1)}\big(z_i^{(1)}\big) = \max\big\{0,\, 0.8\, a_1^{(0)} a_2^{(0)} a_3^{(0)} \;-\; 0.5\, a_1^{(0)} a_2^{(0)} \;+\; b_i^{(1)}\big\}.$$

Layer 2 Computation. Given hidden activations $a^{(1)} = \big[a_1^{(1)},\, a_2^{(1)}\big]^T$, define:

$$\operatorname{Val}\big(\{\{1\}\};\, a^{(1)}\big) = a_1^{(1)}, \quad \operatorname{Val}\big(\{\{2\}\};\, a^{(1)}\big) = a_2^{(1)}, \quad \operatorname{Val}\big(\{\{1\},\{2\}\};\, a^{(1)}\big) = a_1^{(1)}\, a_2^{(1)}.$$

For the single output neuron $i = 1$,

$$z^{(2)} = \sum_{\substack{e \in \mathcal{E}^{(2,3)}: \\ 1 \text{ is target of } e}} w_e^{(2,3)} \operatorname{Val}\big(e;\, a^{(1)}\big) \;+\; b^{(2)}.$$

Assuming both $e_1^{(2,3)}$ and $e_2^{(2,3)}$ contribute, we get:

$$z^{(2)} = 1.2\, a_1^{(1)} \;+\; 0.6\,(a_1^{(1)} a_2^{(1)}) \;+\; 0 = 1.2\, a_1^{(1)} \;+\; 0.6\, a_1^{(1)}\, a_2^{(1)}.$$

Finally,

$$a^{(2)} = \sigma^{(2)}\big(z^{(2)}\big) = \frac{1}{1 + \exp\big(-[\,1.2\, a_1^{(1)} + 0.6\, a_1^{(1)} a_2^{(1)}\,]\big)}.$$

Hence the network's output is

$$f(x) \;=\; a^{(2)} \;\in\; \mathbb{R}.$$

In this medical-triage example, $a^{(2)}$ represents the model's sepsis-risk score based on hierarchical interactions among heart rate, blood pressure, and temperature.

Theorem 3.4. *A feedforward artificial neural n-superhypernetwork (*FANnSHN*) with $n = 1$ coincides with the feedforward artificial neural hypernetwork (*FANHyperNetwork*).*

Proof. When $n = 1$, the iterated powerset $\mathcal{P}^1\big(V^{(\ell-1)}\big)$ is simply $\mathcal{P}\big(V^{(\ell-1)}\big)$, and a 1-superhyperedge $e \in \mathcal{E}^{(\ell,1)} \subseteq \mathcal{P}\big(V^{(\ell-1)}\big) \setminus \{\emptyset\}$ is an ordinary hyperedge. The value function then satisfies

$$\operatorname{Val}\big(e\,;\, a^{(\ell-1)}\big) \;=\; \prod_{j \in e} a_j^{(\ell-1)}, \quad \text{for each } e \subseteq V^{(\ell-1)}.$$

Hence the pre-activation formula (4) becomes

$$z_i^{(\ell)} \;=\; \sum_{\substack{e \in \mathcal{E}^{(\ell,1)}: \\ i \text{ is target of } e}} w_e^{(\ell,1)} \prod_{j \in e} a_j^{(\ell-1)} \;+\; b_i^{(\ell)},$$

which matches exactly the definition of a feedforward artificial neural hypernetwork (Definition of FANHyperNetwork). Therefore, setting $n = 1$ recovers the standard hypernetwork case. □

Definition 3.5 (Underlying Neuron n-SuperHyperNetwork)**.** Let $\mathcal{H}^{(n)} = \big(L, \{n_\ell\}, \{\mathcal{E}^{(\ell,n)}\}, \{w^{(\ell,n)}\}, \{b^{(\ell)}\}, \{\sigma^{(\ell)}\}\big)$ be a FANnSHN. Its *underlying neuron n-superhypernetwork* is the ordered triple

$$H_{\text{neurons}}^{(n)} \;=\; \big(V,\; \mathcal{E},\; w_{\text{all}}\big),$$

defined as follows:

- $V = \bigsqcup_{\ell=0}^{L} \{ (\ell, i) \mid i = 1, \ldots, n_\ell \}$ is the disjoint union of neuron labels, so that (ℓ, i) denotes the i-th neuron in layer ℓ.
- For each layer $\ell = 1, 2, \ldots, L$ and each neuron $i = 1, \ldots, n_\ell$, each n-superhyperedge

$$e \in \mathcal{E}^{(\ell,n)} \subseteq \mathcal{P}^n\big(V^{(\ell-1)}\big)$$

corresponds to a neuron-level hyperedge

$$e \times \{ (\ell, i) \} = \{ (\ell - 1, j) \mid j \in e \} \cup \{ (\ell, i) \},$$

where $\{ (\ell - 1, j) \mid j \in e \} \subseteq V$ is a subset of $(\ell - 1)$-layer neurons, and $\{(\ell, i)\}$ is the target neuron. Collecting all such edges across all layers and target neurons gives

$$\mathcal{E} = \bigsqcup_{\ell=1}^{L} \Big\{ e \times \{(\ell, i)\} \;\Big|\; e \in \mathcal{E}^{(\ell,n)},\ i = 1, \ldots, n_\ell \Big\}.$$

Each element of $\mathcal{E}$ is a nonempty subset of V.

- The weight function $w_{\text{all}} \colon \mathcal{E} \to \mathbb{R}$ is given by

$$w_{\text{all}}\big(e \times \{(\ell, i)\}\big) = w_e^{(\ell,n)} \quad \text{for each } e \times \{(\ell, i)\} \in \mathcal{E}.$$

Theorem 3.6. *The underlying neuron n-superhypernetwork $H^{(n)}_{\text{neurons}}$ is a (directed) n-superhypernetwork in the sense of Definition 2 (n-SuperHypernetwork).*

Proof. We verify that $H^{(n)}_{\text{neurons}} = (V, \mathcal{E}, w_{\text{all}})$ satisfies the three requirements of an n-superhypernetwork:

1° $V = \bigsqcup_{\ell=0}^{L} \{(\ell, i) \mid 1 \le i \le n_\ell\}$ is a finite, nonempty set, since each layer ℓ has n_ℓ neurons and there are finitely many layers.

2° Each element of $\mathcal{E}$ has the form

$$e \times \{(\ell, i)\} = \big\{ (\ell - 1, j) \;\big|\; j \in e \big\} \cup \{ (\ell, i) \},$$

where $e \in \mathcal{E}^{(\ell,n)} \subseteq \mathcal{P}^n(V^{(\ell-1)})$. Since $V^{(\ell-1)} = \{1, \ldots, n_{\ell-1}\}$, each e is a nonempty element of $\mathcal{P}^n(V^{(\ell-1)})$. Thus e itself is a subset of $\mathcal{P}^{n-1}(V^{(\ell-1)})$, and so the set $\{ (\ell - 1, j) \mid j \in e \}$ is a nonempty subset of $\{ (\ell - 1, 1), \ldots, (\ell - 1, n_{\ell-1}) \} \subseteq V$. Adding (ℓ, i) yields a nonempty subset of V. Hence each $e \times \{(\ell, i)\} \in \mathcal{E}$ is a nonempty subset of V. As $\mathcal{E}^{(\ell,n)}$ is finite for each ℓ and there are finitely many neurons i per layer, $\mathcal{E} \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$ is finite.

3° The function $w_{\text{all}} \colon \mathcal{E} \to \mathbb{R}$ is defined by

$$w_{\text{all}}\big(e \times \{(\ell, i)\}\big) = w_e^{(\ell,n)},$$

which is a valid assignment of nonnegative weights (or real weights) to each hyperedge. Thus w_{all} is a well-defined weight function on $\mathcal{E}$.

Since V is finite and nonempty, $\mathcal{E} \subseteq \mathcal{P}(V) \setminus \{\emptyset\}$ is finite, and w_{all} assigns a real number to each edge, all conditions of an n-superhypernetwork (Definition 2) are satisfied. Therefore, $H^{(n)}_{\text{neurons}}$ is a directed n-superhypernetwork whose nodes correspond to neurons and whose n-superhyperedges encode each multi-level grouping in the feedforward architecture. □

4 Conclusion and Future Works

In this paper, we introduced two extensions of the feedforward neural model using HyperGraph and SuperHyperGraph theory: the Artificial Neural HyperNetwork and the Artificial Neural SuperHyperNetwork. In future work, we plan to explore further extensions of these concepts using frameworks such as Fuzzy Sets [37], Intuitionistic Fuzzy Sets [38], Neutrosophic Sets [39], Picture Fuzzy Sets [40], Hesitant Fuzzy Sets [41], and Plithogenic Sets [42].

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] Jonathan L Gross, Jay Yellen, and Mark Anderson. *Graph theory and its applications*. Chapman and Hall/CRC, 2018.

[2] Reinhard Diestel. *Graph theory*. Springer (print edition); Reinhard Diestel (eBooks), 2024.

[3] Claude Berge. *Hypergraphs: combinatorics of finite sets*, volume 45. Elsevier, 1984.

[4] Yifan Feng, Jiashu Han, Shihui Ying, and Yue Gao. Hypergraph isomorphism computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[5] Song Feng, Emily Heath, Brett Jefferson, Cliff Joslyn, Henry Kvinge, Hugh D Mitchell, Brenda Praggastis, Amie J Eisfeld, Amy C Sims, Larissa B Thackray, et al. Hypergraph models of biological networks to identify genes critical to pathogenic viral response. *BMC bioinformatics*, 22(1):287, 2021.

[6] Derun Cai, Moxian Song, Chenxi Sun, Baofeng Zhang, Shenda Hong, and Hongyan Li. Hypergraph structure learning for hypergraph neural networks. In *IJCAI*, pages 1923–1929, 2022.

[7] 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.

[8] Takaaki Fujita and Florentin Smarandache. Fundamental computational problems and algorithms for superhypergraphs. *HyperSoft Set Methods in Engineering*, 3:32–61, 2025.

[9] Florentin Smarandache. *Extension of HyperGraph to n-SuperHyperGraph and to Plithogenic n-SuperHyperGraph, and Extension of HyperAlgebra to n-ary (Classical-/Neutro-/Anti-) HyperAlgebra*. Infinite Study, 2020.

[10] Florentin Smarandache. *Introduction to the n-SuperHyperGraph-the most general form of graph today*. Infinite Study, 2022.

[11] Takaaki Fujita and Florentin Smarandache. A concise study of some superhypergraph classes. *Neutrosophic Sets and Systems*, 77:548–593, 2024.

[12] Takaaki Fujita and Talal Ali Al-Hawary. Short note of superhyperclique-width and local superhypertree-width. *Neutrosophic Sets and Systems*, 86:811–837, 2025.

[13] Takaaki Fujita. Review of some superhypergraph classes: Directed, bidirected, soft, and rough. In *Advancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond (Second Volume)*. Biblio Publishing, 2024.

[14] Takaaki Fujita and Florentin Smarandache. *Superhypergraph neural networks and plithogenic graph neural networks: Theoretical foundations*. Infinite Study, 2025.

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

[16] Alain Bretto. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 1, 2013.

[17] Florentin Smarandache. n-superhypergraph and plithogenic n-superhypergraph. *Nidus Idearum*, 7:107–113, 2019.

[18] Takaaki Fujita. Exploration of graph classes and concepts for superhypergraphs and n-th power mathematical structures. *Advancing Uncertain Combinatorics through Graphization, Hyperization, and Uncertainization: Fuzzy, Neutrosophic, Soft, Rough, and Beyond*, 3(4):512.

[19] Takaaki Fujita. Hypergraph and superhypergraph approaches in electronics: A hierarchical framework for modeling power-grid hypernetworks and superhypernetworks. *Journal of Energy Research and Reviews*, 17(6):102–136, 2025.

[20] PG Benardos and G-C Vosniakos. Optimizing feedforward artificial neural network architecture. *Engineering applications of artificial intelligence*, 20(3):365–382, 2007.

[21] Russell Reed and Robert J MarksII. *Neural smithing: supervised learning in feedforward artificial neural networks*. MIT press, 1999.

[22] George Bebis and Michael Georgiopoulos. Feed-forward neural networks. *Ieee Potentials*, 13(4):27–31, 1994.

[23] Saman Razavi and Bryan A Tolson. A new formulation for feedforward neural networks. *IEEE Transactions on neural networks*, 22(10):1588–1598, 2011.

[24] Murat H Sazlı. A brief review of feed-forward neural networks. *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, 50(01), 2006.

[25] David J Montana, Lawrence Davis, et al. Training feedforward neural networks using genetic algorithms. In *IJCAI*, volume 89, pages 762–767, 1989.

[26] P Patrick Van Der Smagt. Minimisation methods for training feedforward neural networks. *Neural networks*, 7(1):1–11, 1994.

[27] John G Kuschewski, Stefen Hui, and Stanislaw H Zak. Application of feedforward neural networks to dynamical system identification and control. *IEEE transactions on control systems technology*, 1(1):37–49, 1993.

[28] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[29] Anne-Johan Annema. *Feed-forward neural networks: Vector decomposition analysis, Modelling and Analog Implementation*. Springer Science & Business Media, 1995.

[30] Xuexiong Luo, Jia Wu, Jian Yang, Shan Xue, Amin Beheshti, Quan Z Sheng, David McAlpine, Paul Sowman, Alexis Giral, and Philip S Yu. Graph neural networks for brain graph learning: A survey. *arXiv preprint arXiv:2406.02594*, 2024.

[31] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

[32] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, Bo Long, et al. Graph neural networks for natural language processing: A survey. *Foundations and Trends® in Machine Learning*, 16(2):119–328, 2023.

[33] Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. Elastic graph neural networks. In *International Conference on Machine Learning*, pages 6837–6849. PMLR, 2021.

[34] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[35] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 68:5680–5695, 2020.

[36] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with directed graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7912–7921, 2019.

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

[38] Krassimir T Atanassov and Krassimir T Atanassov. *Intuitionistic fuzzy sets*. Springer, 1999.

[39] Florentin Smarandache. A unifying field in logics: Neutrosophic logic. In *Philosophy*, pages 1–141. American Research Press, 1999.

[40] Bui Cong Cuong and Vladik Kreinovich. Picture fuzzy sets-a new concept for computational intelligence problems. In *2013 third world congress on information and communication technologies (WICT 2013)*, pages 1–6. IEEE, 2013.

[41] Vicenç Torra and Yasuo Narukawa. On hesitant fuzzy sets and decision. In *2009 IEEE international conference on fuzzy systems*, pages 1378–1382. IEEE, 2009.

[42] Florentin Smarandache. *Plithogenic set, an extension of crisp, fuzzy, intuitionistic fuzzy, and neutrosophic sets-revisited*. Infinite study, 2018.