A Discrete Non-Additive Integral Based Interval-Valued Neural Network for Enhanced Prediction Reliability

Yassine Hmidy and Mouna Ben Mabrouk
Sogetilabs at Capgemini, Paris, France
{*yassine.hmidy, mouna.benmabrouk*}*@sogeti.com*

Keywords: Neural Networks, Regression, Interval-Valued, Aggregation Function, Trustworthiness.

Abstract: In this paper, we propose to replace the perceptron of classical feedforward neural networks by a new aggregation function. In a recent paper, it has been shown that this new aggregation is a relevant learning model, simple to use, and informative as it outputs an interval whose size is correlated to the prediction error of the model. Unlike a classical neural network whose perceptron are usually composed of a linear aggregation and an activation function, the model we propose here is a mere composition of those aggregation functions. In order to show the relevance of using such a neural network, we rely on experiments comparing its performances with those of a classical neural network.

1 INTRODUCTION

An aggregation function is by definition any function that computes a single output value from a vector of input values. Among these, there are parametric aggregation functions whose computation involves operations between a vector of parameters and the vector of input values. The most commonly used in machine learning model is the arithmetic mean. It is the core operation of each perceptron in classical neural networks.

Though a wide range of activation functions have been customized, from sigmoid functions to ReLU functions, the arithmetic mean still remains unchanged in most of the current neural network. In this paper, we propose a conceptual breakthrough, not only by changing the arithmetic mean for another parametric aggregation function, but by changing the total perceptron. The new perceptron we propose is based on a parametric aggregation function named the Macsum aggregation (Strauss et al., 2022). The Macsum aggregation outputs an interval whose bounds are discrete Choquet integrals. The learning abilities of the Macsum aggregation is presented in (Hmidy et al., 2022b). In this paper we build a neural network whose neurons computes the center of Macsum aggregations except for the last neuron which computes a Macsum aggregation. Therefore the output is interval-valued. In (Hmidy et al., 2022b), the correlation between the width of the interval-valued output of the Macsum aggregation and the prediction error made by the model has been demonstrated. An experiment we carried on demonstrated that the neural network we propose keeps this property.

In summary, the paper contribution is three fold:

- a new learning model, that is a neural network whose perceptrons compute discrete Choquet integrals based aggregation functions,
- the possibility to assess the trustworthiness of a prediction comparing to another through the size of its interval-valued output,
- the unnecessity of using activation functions thanks to the non-linearity naturally induced by the composition of Macsum aggregations.

2 RELATED WORK

In this section we situate our work w.r.t. the literature.

2.1 Choquet Integral Based Neural Network

The Choquet integral is a generalization of Lebesgue integral to non-additive measures (Grabisch, 2015). It is commonly used in decision theory to account for situations that cannot be represented by additive set functions (or games). The relevance of using a Choquet integral as a neuron in a neural network implementation for decision analysis is presented in (Chiang, 1999). In (Shi-Hong and Zheng-You, 2005), a

Hmidy, Y. and Ben Mabrouk, M.
A Discrete Non-Additive Integral Based Interval-Valued Neural Network for Enhanced Prediction Reliability.
DOI: 10.5220/0013155900003890

In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 2*, pages 345-352
ISBN: 978-989-758-737-5; ISSN: 2184-433X

Choquet integral-based neural network turned out to be efficient in multi-classification tasks. In (Machado et al., 2015), the use of a bipolar Choquet integral led to promising results for classification in a context of decision making where underlying scales are bipolar. More recently, in (Bresson et al., 2020), a neural representation of a hierarchical 2-additive Choquet integral has shown efficiency in classification, regression and ranking settings .

2.2 Interval Neural Network

An interval neural network is a neural network whose output is interval-valued.

The interval-valued nature of the output can be due to the use of interval parameters (Garczarczy, 2000; Oala et al., 2020; Beheshti et al., 1998). In these architectures the forward propagation is expressed similarly to classical neural networks by using interval arithmetic instead of usual arithmetic.

This interval-valued nature can also be induced by considering a neural network with interval-valued inputs. Many neural networks have been extended to learn from interval-valued inputs. In (Rossi and Conan-Guez, 2002; Kowalski and Kulczycki, 2017), a probabilistic neural network allows to learn from interval-valued data considering an interval as a uniform distribution between two bounds. We can also find in the literature extension of neural networks to fuzzy input vectors (Hisao and Manabu, 2000).

2.3 Prediction Intervals

Neural networks are used in prediction tasks due to their strong performance and flexibility in modeling complex functions. With the increasing use of neural networks comes the need for developing tools to estimate the uncertainty of their predictions. Prediction intervals is a way to give a measure of uncertainty that can be used on regression problems. Many techniques exist to build such an interval in the case of neural networks. For instance, in (Khosravi et al., 2014), the bootstrap approach (Efron, 1979) is used to build prediction intervals of the neural network outputs. In (Mancini et al., 2021) is presented another method that consists in building multiple neural network models and use their outputs to build the prediction interval.

In this paper we present a feedforward neural network based on a Choquet integral w.r.t. a parametric set function. This neural network outputs an interval that can be considered as an error prediction.

3 PRELIMINARIES

3.1 Notations and Definitions

- $\Omega = \{1, \ldots, N\} \subset \mathbb{N}$.
- $\forall A \subseteq \Omega$, A^c is the complementary of A in Ω.
- $\mathbb{R}$ is the set of real numbers.
- A vector is a function $\boldsymbol{x} : \Omega \to \mathbb{R}$ defined by a discrete subset of $\mathbb{R}^{\mathbb{N}}$ denoted $\boldsymbol{x} = (x_1, \cdots, x_N) \in \mathbb{R}^{\mathbb{N}}$.
- $\overline{\underline{x}} = [\underline{x}, \overline{x}]$ is a real interval whose lower bound is $\underline{x} \in \mathbb{R}$ and upper bound is $\overline{x} \in \mathbb{R}$.
- $\mathbb{IR}$ is the set of real intervals.
- A set function is a function $\mu : 2^{\Omega} \to \mathbb{R}$ that maps any subset of Ω onto a real value. To a set function μ is associated its complementary set function μ^c: $\forall A \subseteq \Omega$, $\mu^c(A) = \mu(\Omega) - \mu(A^c)$. Usually, $\mu(\emptyset) = 0$ where $\emptyset$ is the emptyset of Ω.
- A set function μ is said to be submodular if $\forall A, B \subseteq \Omega$, $\mu(A \cup B) + \mu(A \cap B) \leq \mu(A) + \mu(B)$.
- A set function μ is said to be additive if $\forall A, B \subseteq \Omega$, $\mu(A \cup B) + \mu(A \cap B) = \mu(A) + \mu(B)$.
- If a set function μ is submodular then its complementary μ^c is supermodular.
- The discrete Choquet integral w.r.t. a set function μ is denoted $\check{\mathbb{C}}_\mu$ (Grabisch et al., 2000) and defined by :

$$\check{\mathbb{C}}_\mu(\boldsymbol{x}) = \sum_{k=1}^{N} x_{(k)} . (\mu(A_{(k)}) - \mu(A_{(k+1)})). \qquad (1)$$

  $(.)$ being the permutation that sorts the element of $\boldsymbol{x}$ in increasing order: $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}$ and $A_{(i)}$ $(i \in \Omega)$ being the coalition of Ω such that $A_{(i)} = \{(i), \ldots, (N)\}$ with $A_{(N+1)} = \emptyset$.
- If a set function μ is submodular then $\forall \boldsymbol{x} \in \mathbb{R}^N$, $\check{\mathbb{C}}_\mu(\boldsymbol{x}) \geq \check{\mathbb{C}}_{\mu^c}(\boldsymbol{x})$ (Grabisch, 2016).

3.2 Macsum Aggregation

Let $\boldsymbol{\theta} \in \mathbb{R}^N$ be a vector used as a parameter. The Macsum set function $\nu_{\boldsymbol{\theta}}$ and its complementary set function $\nu^c_{\boldsymbol{\theta}}$, has been defined in (Strauss et al., 2022) as $\forall A \subseteq \Omega$:

$$\nu_{\boldsymbol{\theta}}(A) = \max_{i \in A} \theta_i^+ + \min_{i \in \Omega} \theta_i^- - \min_{i \in A^c} \theta_i^-, \qquad (2)$$

$$\nu^c_{\boldsymbol{\theta}}(A) = \min_{i \in A} \theta_i^- + \max_{i \in \Omega} \theta_i^+ - \max_{i \in A^c} \theta_i^+, \qquad (3)$$

with $\forall i \in \Omega$, $\theta_i^+ = \max(0, \theta_i)$ and $\theta_i^- = \min(0, \theta_i)$.

The Macsum set function being a parametric set function that is submodular (Strauss et al., 2022), $\check{\mathbb{C}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) \geq \check{\mathbb{C}}_{\nu^c_{\boldsymbol{\theta}}}(\boldsymbol{x})$. The Macsum aggregation $\mathcal{A}_{\nu_{\boldsymbol{\theta}}}$ is defined by using the Macsum set function such that: $\forall \boldsymbol{x} \in \mathbb{R}^N$, $\overline{\underline{y}} = [\underline{y}, \overline{y}] = \mathcal{A}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) = [\check{\mathbb{C}}_{\nu^c_{\boldsymbol{\theta}}}(\boldsymbol{x}), \check{\mathbb{C}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})]$.

Let $\boldsymbol{\psi} \in \mathbb{R}^N$, $\forall A \subseteq \Omega$, we denote $\lambda_{\boldsymbol{\psi}}$ the linear parametric set function defined by:

$$\lambda_{\boldsymbol{\psi}}(A) = \sum_{i \in A} \psi_i.$$

A consequence of the submodularity of the Macsum set function is that it dominates a set of linear parametric set function (Strauss et al., 2022).

The set of all parameters $\boldsymbol{\psi}$ such that $\lambda_{\boldsymbol{\psi}}$ is dominated by the Macsum set function w.r.t. the parameter $\boldsymbol{\theta}$ can be written this way: $\mathcal{M}(\boldsymbol{\theta}) = \{\boldsymbol{\psi} \in \mathbb{R}^N | \forall A \subseteq \Omega, \nu^c_{\boldsymbol{\theta}}(A) \leq \lambda_{\boldsymbol{\psi}}(A) \leq \nu_{\boldsymbol{\theta}}(A)\}$.

This set is convex (Strauss et al., 2022), which means that $\forall \psi_1, \psi_2 \in \mathcal{M}(\boldsymbol{\theta}), \gamma\psi_1 + (1-\gamma)\psi_2 \in \mathcal{M}(\boldsymbol{\theta})$ with $\gamma \in [0,1]$.

Let $\boldsymbol{\psi} \in \mathbb{R}^N$, the linear aggregation is defined by using the linear operator $\lambda_{\boldsymbol{\psi}}$ as:

$$\mathcal{A}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x}) = \check{\mathbb{C}}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x}) = \sum_{i \in \Omega} \psi_i . x_i.$$

Therefore, the Macsum aggregation can be interpreted as:

$$\begin{aligned} \mathcal{A}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) &= \left\{ \mathcal{A}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x}) / \boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\theta}) \right\} \\ &= [\underline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}), \overline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})], \end{aligned} \tag{4}$$

this set being convex (Strauss et al., 2022): $\forall \boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\theta})$, $\exists y \in \mathcal{A}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})$ such that $y = \mathcal{A}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x})$ and $\forall y \in \mathcal{A}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})$, $\exists \boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\theta})$ such that $y = \mathcal{A}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x})$.

As the Macsum aggregation is a set of linear aggregation whose bounds depend on the same parameter, we can learn a set of linear aggregation by learning a Macsum aggregation through the updating of one parameter, with the usual gradient descent method as shown in (Hmidy et al., 2022b).

3.3 Computation of the Macsum Aggregation

The upper and lower bound of the Macsum aggregation of a vector $\boldsymbol{x} \in \mathbb{R}^N$ w.r.t. a parameter $\boldsymbol{\theta} \in \mathbb{R}^N$ can easily be obtained by considering expressions (1), (2) and (3):

$$\begin{aligned} \overline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) &= \sum_{k=1}^{N} x_{(k)} \cdot \left(\max_{i=k}^{N} \theta^+_{(i)} - \min_{i=1}^{k-1} \theta^-_{(i)} \right) \\ &\quad - \sum_{k=1}^{N} x_{(k)} \cdot \left(\max_{i=k+1}^{N} \theta^+_{(i)} + \min_{i=1}^{k} \theta^-_{(i)} \right), \end{aligned} \tag{5}$$

and

$$\begin{aligned} \underline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) &= \sum_{k=1}^{N} x_{(k)} \cdot \left(\min_{i=k}^{N} \theta^-_{(i)} - \max_{i=1}^{k-1} \theta^+_{(i)} \right) \\ &\quad - \sum_{k=1}^{N} x_{(k)} \cdot \left(\min_{i=k+1}^{N} \theta^-_{(i)} + \max_{i=1}^{k} \theta^+_{(i)} \right), \end{aligned} \tag{6}$$

where $(.)$ is the permutation that sorts the element of $\boldsymbol{x}$ in increasing order: $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}$.

There exists an equivalent form for the Macsum aggregation (Hmidy et al., 2022b) allowing its computation with sorting the parameter vector $\boldsymbol{\theta}$ instead of sorting the input vector $\boldsymbol{x}$:

$$\begin{aligned} \overline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) &= \sum_{k=1}^{N} \theta^+_{\lfloor k \rfloor} \cdot \left(\max_{i=1}^{k} x_{\lfloor i \rfloor} - \max_{i=1}^{k-1} x_{\lfloor i \rfloor} \right) \\ &\quad + \sum_{k=1}^{N} \theta^-_{\lceil k \rceil} \cdot \left(\min_{i=1}^{k} x_{\lceil k \rceil} - \min_{i=1}^{k-1} x_{\lceil k-1 \rceil} \right), \end{aligned} \tag{7}$$

and:

$$\begin{aligned} \underline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) &= \sum_{k=1}^{N} \theta^-_{\lfloor k \rfloor} \cdot \left(\min_{i=1}^{k} x_{\lfloor i \rfloor} - \min_{i=1}^{k-1} x_{\lfloor i \rfloor} \right) \\ &\quad + \sum_{k=1}^{N} \theta^+_{\lceil k \rceil} \cdot \left(\max_{i=1}^{k} x_{\lceil k \rceil} - \max_{i=1}^{k-1} x_{\lceil k-1 \rceil} \right), \end{aligned} \tag{8}$$

where $\lfloor . \rfloor$ is a permutation that sorts $\boldsymbol{\theta}$ in decreasing order ($\theta_{\lfloor 1 \rfloor} \geq \theta_{\lfloor 2 \rfloor} \geq \cdots \geq \theta_{\lfloor N \rfloor}$), and $\lceil . \rceil$ is a permutation that sorts $\boldsymbol{\theta}$ in increasing order ($\theta_{\lceil 1 \rceil} \leq \theta_{\lceil 2 \rceil} \leq \cdots \leq \theta_{\lceil N \rceil}$), with $\min_{i=1}^{0} x_{\lfloor i \rfloor} = 0 = \max_{i=1}^{0} x_{\lceil i \rceil}$ and $\max_{i=1}^{0} x_{\lfloor i \rfloor} = 0 = \min_{i=1}^{0} x_{\lceil i \rceil}$.

4 MACSUM NEURAL NETWORK FOR REGRESSION

In this section, we present a new learning model based on the Macsum aggregation to replace the classical neuron in a feedforward neural network. The motivation for introducing this model is to show that the flexibility of the aggregation function we propose can model a wide range of input/output relations and therefore allows to get rid of the activation function whose choice is often arbitrary. We define the Macsum perceptron as being the center of the Macsum aggregation. Getting rid of the activation function is possible thanks to the inherent non-linearity of the Macsum perceptron.

4.1 Macsum Perceptron

We propose to define the Macsum perceptron, denoted $\mathcal{S}_{\nu_{\boldsymbol{\theta}}}$, as being the following parametric function:

$$\forall \boldsymbol{\theta} \in \mathbb{R}^N, \; \mathcal{S}_{\nu_{\boldsymbol{\theta}}} \;:\; \begin{array}{ccl} \mathbb{R}^N & \rightarrow & \mathbb{R} \\ \boldsymbol{x} & \mapsto & \frac{1}{2}\left(\underline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) + \overline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})\right) \end{array}$$

As for any classical perceptron, the learning process of a Macsum perceptron uses the gradient descent method to update the parameters. To able a similar learning process, the derivatives of the Macsum perceptron w.r.t. its parameters and w.r.t. its inputs are required.

The derivative of $\underline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}$ and $\overline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}$ w.r.t. its parameters have been established in (Hmidy et al., 2022b) . The derivative of $\mathcal{S}_{\nu_{\boldsymbol{\theta}}}$ w.r.t. the k^{th} parameter is:

$$\begin{aligned} \frac{\delta \mathcal{S}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})}{\delta \theta_k} = \frac{1}{2} \cdot \Bigg(& \max_{i=1}^{l} x_{\lfloor i \rfloor} - \max_{i=1}^{l-1} x_{\lfloor i \rfloor} + \min_{i=1}^{u} x_{\lceil i \rceil} \\ & - \min_{i=1}^{u-1} x_{\lceil i \rceil} + \min_{i=1}^{l} x_{\lfloor i \rfloor} - \min_{i=1}^{l-1} x_{\lfloor i \rfloor} \\ & + \max_{i=1}^{u} x_{\lceil i \rceil} - \max_{i=1}^{u-1} x_{\lceil i \rceil} \Bigg), \end{aligned}$$

with l and u being the index such that $\lfloor l \rfloor = k$ and $\lceil u \rceil = k$.

The derivates of $\mathcal{S}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})$ w.r.t. its inputs can be easily obtained by considering Equations (5) and (6):

$$\begin{aligned} \frac{\delta \mathcal{S}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})}{\delta x_k} = \frac{1}{2} \cdot \Bigg(& \max_{i=l}^{N} \theta^+_{(i)} - \min_{i=1}^{l-1} \theta^-_{(i)} - \max_{i=l+1}^{N} \theta^+_{(i)} \\ & + \min_{i=1}^{l} \theta^-_{(i)} + \min_{i=l}^{N} \theta^-_{(i)} - \max_{i=1}^{l-1} \theta^+_{(i)} \\ & - \min_{i=l+1}^{N} \theta^-_{(i)} + \max_{i=1}^{l} \theta^+_{(i)} \Bigg), \end{aligned}$$

with $(.)$ being the permutation that sorts the element of $\boldsymbol{x}$ in increasing order: $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}$ and l the index such that $(l) = k$.

4.2 A Non-Linear Model with no Activation Function

In a classical feedforward neural network each perceptron is composed of a linear aggregation function followed by an activation function. In the case of a Macsum neural network the aggregation function is non-linear which rids the model of activation functions. Let us give a simple example:
Let $\boldsymbol{\theta} = (-1, 2, -3)$, $\boldsymbol{x} = (1, 0, 3)$ and $\boldsymbol{y} = (-2, 3, 9)$: then $\mathcal{S}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x} + \boldsymbol{y}) = 4$ while $\mathcal{S}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) + \mathcal{S}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{y}) = 5$.

Consequently, each perceptron of a Macsum neural network is non-linear as for a classical neural network.

4.3 An Interval Model for Regression

A way to learn the parameters of a Macsum aggregation based on a dataset have been proposed in (Hmidy et al., 2022b). The experiment associated to this work shed light on a correlation between the width of the interval-valued output and the prediction error, i.e. the distance between the target value and the center of the interval-valued output. This property is interesting since it allows to have information on the quality of a prediction. We propose to keep this prediction ability by using a Macsum aggregation instead of a Macsum perceptron for the last neuron. This will provide an interval valued output. The idea is to change this classical structure: To obtain

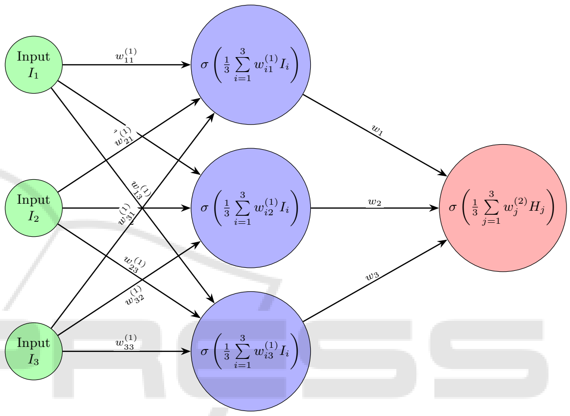

Figure 1: Classical neural network.

a structure in which neuron on the hidden layers are Macsum perceptron and the output neuron is the Macsum aggreation and is therefore interval-valued.

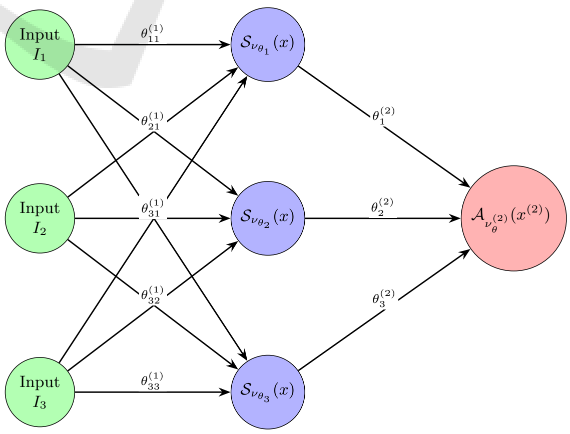

Figure 2: Macsum neural network.

5 EXPERIMENTS

We present two experiments. The aim of the first experiment was to show the relevance of using a Macsum neural network as a learning model compared to a Macsum aggregation. The aim of the second experiment was to demonstrate the ability of a Macsum neural network to compete with a classical neural network on a regression task.

To determine the performances of a learning model we relied on the following criteria:

- the cost value which is the Euclidean distance between the middle of the predicted output $\frac{1}{2}(\overline{y}+\underline{y})$ and the actual output (target) for the Macsum neural network and the Euclidean distance between the predicted output and the actual output for the classical networks,
- the resulting criterion R^2 (Lewis-Beck, 2015),
- the belonging rate that is the percentage of targets that belongs to the interval-valued output $[\underline{y},\overline{y}]$,
- the mean width of the predicted output $\frac{1}{2}(\overline{y}-\underline{y})$ for the Macsum neural network,
- the Pearson correlation between the predicted error $\frac{1}{2}(\overline{y}-\underline{y})$ and the prediction error for the Macsum neural network.

5.1 Comparison with a Macsum Aggregation

In these experiments we first compared a Macsum neural network to a Macsum aggregation on their ability to learn a linear relation. Secondly, we compared them on their ability to learn a non-linear relation in order to shed light on the relevance of using a Macsum neural network instead of a simple Macsum aggregation.

We implemented these experiment on noiseless synthetic data.

We built two dataset of 700 examples such that $\forall i \in \{1,...,700\}$ the i^{th} target was a real value $y^i = f(X^i,\psi)$ with the parameter $\psi \in \mathbb{R}^{13}$ randomly picked according to the standard normal distribution and the elements of the input vectors $X^i \in \mathbb{R}^{13}$ were uniformly picked between -10 and 10.
The function f was linear in the first experiment and non-linear in the second.

We split the dataset into 500 examples for the training phase and 200 examples for the test phase.

To perform these experiments, we proposed to use a very simple fully connected 3-layer neural network with two neurons on the input layer, two neurons on the hidden layer and one neuron on the output layer.

The learning rate was arbitrarily set to 0.001 and the number of epochs to 2000.

5.1.1 Learning a Linear Relation

The linear relation had the following form:

$$\forall i \in \{1,...,700\},\ y^i = \sum_{j=1}^{13} X^i_j.\psi_j.$$

Table 1: Results on the linear dataset.

Criteria	Macsum Net	Macsum Neuron
	train / test	train / test
Cost value	0.33 / 0.43	0.24 / 0.57
Belonging	80.3% / 80.3%	100% / 100%
Width	6.34 / 6.04	652 / 703
R^2	0.992 / 0.992	1 / 1

Table 1 brought out the ability of the Macsum neural network to learn linear data since the cost value was quite low and R^2 was high. Nevertheless the Macsum perceptron outperformed it in terms of belonging rate. Although the Macsum neural network gave much smaller intervals than the Macsum aggregation. Thus, the interval-valued output given by the Macsum neural network was much more informative than this of the Macsum aggregation.

Furthermore, the similarity of the performances on both the train and the test dataset showed the generalization ability of both models.

5.1.2 Learning a Non-Linear Relation

As the Macsum aggregation is a convex set of linear functions, the aim here was to show that a composition of Macsum perceptron was non-linear enough to get rid of activation functions.

The non-linear relation had the following form:

$$\forall i \in \{1,...,700\},\ y^i = \sum_{j=1}^{13} \sin(X^i_j).\psi_j.$$

We chose a sinusoidal function because of its strong non-linearity.

Table 2: Results on the non-linear dataset.

Criteria	Macsum Net	Macsum Neuron
	train / test	train / test
Cost value	12.27 / 9.36	52.09 / 55.36
Belonging	52% / 50.9%	49.6% / 49.6%
Width	7.31 / 8.35	18.52 / 18.30
R^2	0.94 / 0.95	0.053 / 0.029

In Table 2, the significant difference between the R^2 of the Macsum neural network and the Macsum perceptron showed the relevance of using a network of Macsum perceptron instead of a simple Macsum perceptron to learn non-linear relations. As in the previous experiment, the neural network gave a more specific i.e. more informative interval. Also the R^2 was the same on both training and test dataset.

5.2 Comparison Between Macsum Neural Network and Classical Neural Network Across Multiple Datasets

The goal of this experiment was to compare the performance of a Macsum neural network with a classical neural network across multiple regression tasks using different datasets. Each dataset involves predicting a variable based on others, with neural networks structured identically across tasks.

In all experiments, the networks were composed of 8 fully connected layers (6 hidden layers) with 9 neurons in the first layer, 18 in the second, 36 in the third, 72 in the fourth, 36 in the fifth, 18 in the sixth, 9 in the seventh, and 1 neuron in the last layer. This structure was chosen arbitrarily for consistency across datasets.

In a classical neural network, each neuron is composed of an affine function (a linear aggregation with a bias) and an activation function, except for the last perceptron, which has no activation function. For the Macsum neural network, each neuron is composed of a Macsum perceptron, which is a non-linear aggregation function with no bias and no activation function.

The learning rate was initialized at 10^{-12} for the classical neural network and 10^{-5} for the Macsum neural network in all experiments. Learning rates were decreased gradually in steps of 1000 iterations to encourage convergence.

Below, we describe the datasets and the tasks, followed by a comparison of results.

5.2.1 Datasets

- **Dataset 1: CalCOFI Dataset (Oceanographic Data) (Dane, 2018)**
  Objective: Predict water temperature based on salinity.
  Inputs: Water salinity and other environmental factors.
  Output: Water temperature.
  Samples: 1000 (randomly selected for training and testing).
- **Dataset 2: Szeged Weather Data (Budincsevity, 2017)**
  Objective: Predict apparent temperature based on humidity.
  Inputs: Humidity and other weather-related variables.
  Output: Apparent temperature.
  Samples: 800 (randomly selected).
- **Dataset 3: World War Two Weather Data (Smith, 2019)**
  Objective: Predict maximum temperature based on minimum temperature.
  Inputs: Daily minimum temperature.
  Output: Daily maximum temperature.
  Samples: 700 (randomly selected).
- **Dataset 4: Montreal Bike Lane Usage (Monleon, 2020)**
  Objective: Predict the number of bicyclists on a specific bike path based on counts from other paths.
  Inputs: Counts of bicyclists on different bike paths.
  Output: Number of bicyclists on a selected path.
  Samples: 600 (randomly selected).
- **Dataset 5: New York City East River Bicycle Crossings (of New York, 2021)**
  Objective: Predict the number of bicyclists on one bridge based on counts from other bridges.
  Inputs: Counts of bicyclists on different bridges.
  Output: Number of bicyclists on a specific bridge.
  Samples: 800 (randomly selected).
- **Dataset 6: UK Road Safety Data (Tsiaras, 2020)**
  Objective: Predict the number of casualties in road accidents based on the number of people in the car.
  Inputs: Number of people in the car and other contextual factors.
  Output: Number of casualties.
  Samples: 900 (randomly selected).

5.3 Results Across Datasets

The following tables summarize the results of the experiments on all datasets for both the Macsum neural network and the classical neural network.

5.3.1 Euclidean Distance (Train/Test)

The classical neural network learning curve often shows variability in the learning process depending on the dataset used, with performance improvements occurring at different rates and sometimes with significant fluctuations. In contrast, with the Macsum neural network, the learning process tends to be more uniform across different datasets. This indicates that Macsum networks may generalize better, with more stable convergence patterns, reducing the impact of

Table 3: Euclidean Distance across datasets for both models.

Dataset	Classical NN	Macsum NN
CalCOFI	12.532 / 13.145	6.314 / 6.765
Szeged	14.120 / 15.812	7.821 / 8.335
WW2	11.431 / 12.009	5.157 / 5.634
Mt. Bike	7.215 / 7.897	3.712 / 3.987
NYC Bike	8.635 / 9.324	4.012 / 4.563
UK Road	5.135 / 6.023	2.985 / 3.235

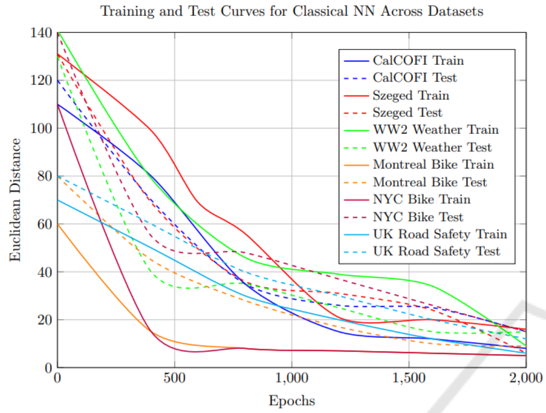

Figure 3: Euclidean Distance across datasets for the classical neural network.

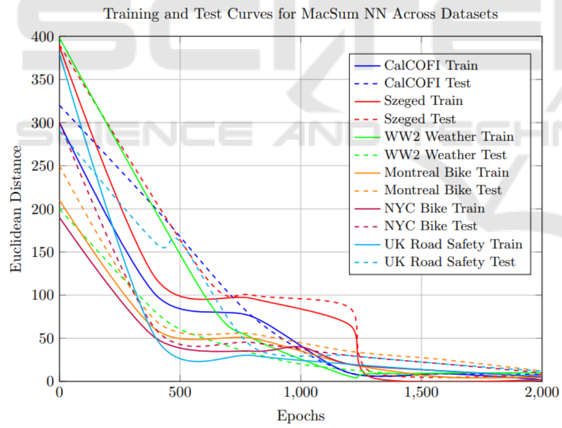

Figure 4: Euclidean Distance across datasets for the Macsum neural network.

dataset-specific characteristics on learning stability and speed.

5.3.2 R^2 Coefficient (Train/Test)

5.3.3 Mean Width of Predicted Output

As seen in table 5, the correlation between the predicted and actual errors steadily increases with more training epochs, indicating that both networks are becoming more capable of reducing their error margins.

Table 4: R^2 Coefficient across datasets for both models.

Dataset	Classical NN	Macsum NN
CalCOFI	0.121 / 0.223	0.102 / 0.245
Szeged	0.341 / 0.315	0.253 / 0.325
WW2	0.212 / 0.290	0.221 / 0.301
Mt. Bike	0.174 / 0.223	0.202 / 0.285
NYC Bike	0.153 / 0.211	0.235 / 0.301
UK Road	0.042 / 0.125	0.123 / 0.201

Table 5: Mean Width and Pearson correlation across datasets.

Dataset	Mean Width	Correlation
CalCOFI	14.7 / 13.2	0.68
Szeged	5.41 / 5.32	0.73
WW2	4.91 / 5.10	0.72
Mt. Bike	16.1 / 16.05	0.80
NYC Bike	5.31 / 5.45	0.66
UK Road	14.72 / 14.85	0.85

6 CONCLUSION AND FUTURE WORK

In this work, we proposed a neural network whose perceptrons are based on a new aggregation method called the Macsum aggregation. Given the Macsum perceptron is non-linear there is no need for activation functions. A comparison with a classical neural network with a ReLu activation function confirmed the potential competitiveness of the Macsum neural network with regard to the state of the art. On top of that, the Macsum neural network goes toward the current trend of trustworthy learning model. Indeed, as the length of the interval-valued output is correlated to the error made by the model we have information on the quality of the prediction that can be crucial in decision making. Therefore not only the Macsum neural network has satisfying learning performances but also it gives insights of the prediction error.
Yet, many improvement can be done for increasing the performances of the Macsum neural network. An important issue to tackle is the computation cost of the Macsum perceptron since it is much higher that this of a classical perceptron. We could also think of an architecture that combines Macsum perceptrons with classical perceptrons to alleviate the computation cost. Further work could go towards trying to tighten the size of the interval output for them to be more informative. The potential of this new model is far from being fully exploited and several directions might lead to interesting results. For instance we could use an extension of the Macsum aggregation to interval-valued input (Hmidy et al., 2022a) to learn from interval-valued data.

REFERENCES

Beheshti, M., Berrached, A., de Korvin, A., Hu, C., and Sirisaengtaksin, O. (1998). On interval weighted three-layer neural networks. In *Proceedings of the 31st Annual Simulation Symposium*, pages 188–194.

Bresson, R., Cohen, J., Hüllermeier, E., Labreuche, C., and Sebag, M. (2020). Neural representation and learning of hierarchical 2-additive choquet integrals. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1984–1991.

Budincsevity, N. (2017). Weather in szeged 2006-2016. https://www.kaggle.com/datasets/budincsevity/szeged-weather. Kaggle Dataset.

Chiang, J. (1999). Choquet fuzzy integral-based hierarchical networks for decision analysis. *IEEE Transactions on Fuzzy Systems*, 7(1):63–71.

Dane, S. (2018). Calcofi. https://www.kaggle.com/datasets/sohier/calcofi. Kaggle Dataset.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7:1–26.

Garczarczy, Z. A. (2000). Interval neural networks. In *2000 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 3, pages 567–570.

Grabisch, M. (2015). Fuzzy measures and integrals: Recent developments. In *Fifty Years of Fuzzy Logic and Its Applications*, pages 125–151.

Grabisch, M. (2016). *Set Functions, Games and Capacities in Decision Making*. Springer.

Grabisch, M., Sugeno, M., and Murofushi, T. (2000). *Fuzzy Measures and Integrals: Theory and Applications*. Physica, Heidelberg.

Hisao, I. and Manabu, N. (2000). Neural networks for soft decision making. *Fuzzy Sets and Systems*, 115(1):121–140.

Hmidy, Y., Rico, A., and Strauss, O. (2022a). Extending the macsum aggregation to interval-valued inputs. In *15th International Conference on Scalable Uncertainty Management*, volume 13562, pages 338–347.

Hmidy, Y., Rico, A., and Strauss, O. (2022b). Macsum aggregation learning. *Fuzzy Sets and Systems*, 24.

Khosravi, A., Nahavandi, S., Srinivasan, D., and Khosravi, R. (2014). Constructing optimal prediction intervals by using neural networks and bootstrap method. *IEEE Transactions on Neural Networks and Learning Systems*, 26:1810–1815.

Kowalski, P. A. and Kulczycki, P. (2017). Interval probabilistic neural network. *Neural Computing and Applications*, 28.

Lewis-Beck, C. (2015). *Applied Regression: An Introduction*, volume 22. Sage Publications.

Machado, M., Flavio, L., Jusan, D., and Caldeira, A. (2015). Using a bipolar choquet neural network to locate a retail store. In *3rd International Conference on Information Technology and Quantitative Management*, volume 55, pages 741–747.

Mancini, T., Calvo-Pardo, H., and Olmo, J. (2021). Prediction intervals for deep neural networks. *arXiv preprint arXiv:2010.04044v2*.

Monleon, P. (2020). Montreal bike lanes. https://www.kaggle.com/datasets/pablomonleon/montreal-bike-lanes. Kaggle Dataset.

Oala, L., Heiß, C., Macdonald, J., März, M., Samek, W., and Kutyniok, G. (2020). Interval neural networks: Uncertainty scores. *arXiv preprint arXiv:2003.11566*.

of New York, C. (2021). New york city - east river bicycle crossings. https://www.kaggle.com/datasets/new-york-city/nyc-east-river-bicycle-crossings. Kaggle Dataset.

Rossi, F. and Conan-Guez, B. (2002). Multi-layer perceptron on interval data. In *Classification, Clustering, and Data Analysis*, pages 427–434.

Shi-Hong, Y. and Zheng-You, W. (2005). Toward accurate choquet integral-based neural network. In *International Conference on Machine Learning and Cybernetics*, volume 8, pages 4621–4624.

Smith, S. (2019). Weather conditions in world war two. https://www.kaggle.com/datasets/smid80/weatherww2. Kaggle Dataset.

Strauss, O., Rico, A., and Hmidy, Y. (2022). Macsum: A new interval-valued linear operator. *International Journal of Approximate Reasoning*, 145:121–138.

Tsiaras, T. (2020). Uk road safety: Traffic accidents and vehicles. https://www.kaggle.com/datasets/tsiaras/uk-road-safety-accidents-and-vehicles. Kaggle Dataset.