



UNIVERSITY OF PADOVA

DEPARTMENT OF MATHEMATICS "TULLIO LEVI-CIVITA"

MASTER THESIS IN DATA SCIENCE

BEYOND MATERIAL IMPLICATION: AN EMPIRICAL STUDY OF RESIDUUM IN KNOWLEDGE ENHANCED NEURAL NETWORKS

SUPERVISOR

LUCIANO SERAFINI
UNIVERSITY OF PADOVA

CO-SUPERVISOR

ALESSANDRO DANIELE
FONDAZIONE BRUNO KESSLER

MASTER CANDIDATE

ANDREA MASCARI
2058072

ACADEMIC YEAR

2022-2023

TO THOSE WHO HOLD A SPECIAL PLACE IN MY HEART.

Abstract

Knowledge Enhanced Neural Networks (KENN) is a neuro-symbolic architecture that exploits fuzzy logic for injecting prior knowledge, codified by first-order logic formulas, into a neural network. It works by adding a new layer at the end of a generic neural network that further elaborates the initial predictions according to the knowledge. In the existing KENN, prior knowledge is represented as a set of conjunctive normal form formulas, and the representation of conditional statements is done according to material implication rule. The following work extends this interpretation of the implication by using the fuzzy logic's residuum semantic and shows how it has been integrated into the original KENN architecture. The residuum integration allowed to evaluate KENN on MNIST Addition and Visual Sudoku, examples of tasks that couldn't be approached by the original architecture, and the results obtained were comparable to others state of the art neuro-symbolic methods. The extended architecture of KENN has subsequently been evaluated also on visual relationship detection, showing that it could improve the performance of the original underlying neural network.

Contents

ABSTRACT	v
LIST OF FIGURES	ix
LIST OF TABLES	xi
LISTING OF ACRONYMS	xiii
1 INTRODUCTION	1
2 RELATED WORK	3
3 BACKGROUND	5
3.1 Fuzzy logic	5
3.1.1 T-norm	5
3.1.2 Residuum	6
3.2 Knowledge Enhanced Neural Network	8
3.2.1 T-conorm boost function	10
3.2.2 Knowledge Enhancer	12
3.2.3 Clause enhancer	12
3.2.4 Relational KENN architecture	13
3.2.5 Experimental results of KENN	16
4 RESIDUUM ENHANCED NEURAL NETWORKS	19
4.1 Residuum boost function	19
4.2 R-implication knowledge enhancer	24
4.3 Residuum clause enhancer	25
5 EXPERIMENTS	27
5.1 Citeseer	28
5.1.1 Dataset	28
5.1.2 Experiment	29
5.1.3 Results discussion	34
5.2 Visual relationship detection	36
5.2.1 Dataset	36
5.2.2 Experiment	38

5.2.3	Results discussion	41
5.3	MNIST addition	43
5.3.1	Dataset	43
5.3.2	Experiment	44
5.3.3	Results Discussion	47
5.4	Visual sudoku	48
5.4.1	Dataset	48
5.4.2	Experiment	51
5.4.3	Results discussion	55
6	CONCLUSION	57
6.1	Future work	58
	REFERENCES	59
A	SOURCE CODE	63
	ACKNOWLEDGMENTS	69

Listing of figures

3.1	An overview of the complete architecture of KENN presented by Daniele and Serafini [1]. The figure shows in green the structure of the Knowledge Enhancer while in pink it is shown the Clause Enhancer. In this example, the KE has two clauses c_1, c_2 to apply on the preactivation $z = [z_A, z_B, z_C, z_D]$. On the right is shown only the CE that handles clause $c_1 : A \vee \neg B$ involving only predicates A, B corresponding to the first two columns of z . At the end of computations the CE proposes changes $\delta_A^{c_1}$ and $\delta_B^{c_1}$ to be added to columns z_A and z_B	16
5.1	KENN improvements over underlying neural network	30
5.2	The histograms showing the result densities obtained on transductive learning for each training dimension. In the left columns are printed the accuracy scores for each of the three models while on the right are the improvements in accuracy carried out by KENN models w.r.t. the NN. In green are depicted the deltas computed using material implication, while in yellow are the ones computed from mixed knowledge and NN.	33
5.3	Improvements over the underlying neural network using the new set of logic rules	34
5.4	Within-image example on the left, across-image on the right. With relative annotations.	37
5.5	Confusion matrix on the MNIST classification for a local minimum	46
5.6	Example of KMNIST images	49
5.7	Example of FMNIST images	49

Listing of tables

5.1	1% of training set, 100 runs	39
5.2	5% of training set, 100 runs	39
5.3	10% of training set, 100 runs	40
5.4	20% of training set, 100 runs	40
5.5	The table compares ILR, residuum KENN, and material implication KENN average results obtained in the MNIST Addition experiment. 10 runs	45
5.6	Results of the same experiment showed in table 5.5, but this time computed on 30 runs.	46
5.7	Residuum KENN and CNN results on the Basic task on 10 runs with two different train sets	54
5.8	Residuum KENN and CNN results on PerSplit task, 20 runs	54

Listing of acronyms

CE	Clause Enhancer
CNN	Convolutional Neural Network
CNF	Conjunctive Normal Form
DNF	Disjunctive Normal Form
ILR	Iterative Local Refinement
KE	Knowledge Enhancer
KENN	Knowledge Enhanced Neural Network
LTN	Logic Tensor Network
NN	Neural Network
RBF	Residuum Boost Function
RCE	Residuum Clause Enhancer
TBF	T-conorm Boost Function
VRD	Visual Relationship Detection

1

Introduction

The thesis resumes the work carried out during the internship in the Data and Knowledge Management Research Unit (DKM)[2] of Fondazione Bruno Kessler. Data and knowledge management is an interdisciplinary research field between computer science, artificial intelligence, and mathematics. It comprises a range of practices used to create, represent, share, and exploit knowledge in any type of domain. A field of particular interest in DKM is Neuro-symbolic AI (NeSy). It includes the methods that combine the strengths of connectionist AI, like neural architectures, with the capabilities of human-like symbolic knowledge and reasoning.

During the internship, I worked on a neuro-symbolic model called Knowledge Enhanced Neural Network (KENN)[1][3]. KENN exploits fuzzy logic to inject prior knowledge into the predictions of a generic neural network. My responsibilities were to extend the model by implementing a new semantic representing implication rules. Also, to test those implementations on different tasks, in order to evaluate new capabilities and compare them with the ones of starting architecture.

A given knowledge can be taken as input by KENN under first-order logic formulas written in conjunctive normal form (CNF). It is then applied to the neural network predictions in order to make them follow the specified logic rules. More precisely, if some of the preactivation values caused the neural network to make predictions that don't respect a logic rule, KENN would either increase or decrease them in such a way that the truth value of all logic rules can be satisfied. The original architecture handles a set of disjunctive clauses, where the truth value of each formula is computed by the Gödel t-conorm operation. To augment the truth value of

each disjunctive clause, KENN makes use of a *t-conorm boost function*.

To represent a conditional statement with disjunctive clauses, we can follow the material implication rule, which states that $A \rightarrow B$ is equivalent to $\neg A \vee B$. Although such a rule is in general valid in classical logic, the same is not true for fuzzy logic where the implication is often interpreted using a residuum function. In standard t-norm based fuzzy logic, the residuum function is the unique binary operation that gives the implication truth value equal to that of the consequent, if the latter is lower than that of the antecedent. Otherwise, it gives the implication truth value equal to 1.

In the thesis, I present an extension of KENN which implements a *residuum boost function*. Similarly to the t-conorm boost function, the residuum boost function is thought to increase the truth value of implicational logic rules interpreted by residuum of the t-norm. Experimental results showed how the use of residuum boost function allows to perform further reasoning starting from neural network predictions as evidenced by the satisfactory results obtained in *MNIST addition* and *Visual Sudoku* puzzle-classification experiments. The same results were unreproducible with the as-is version of KENN, meaning that the reasoning capabilities were a successful achievement made possible by the integration of residuum semantics. The two versions of KENN were also tested on the experiments that require modifications to neural network predictions. It was the case of scientific publications topic prediction (Citeseer) and Visual Relationship Detection. In these scenarios, the residuum boost function is not always the better option for improving the quality of predictions, and a further analysis was done to understand in which situations the use of a specific boost function instead of the other is more appropriate.

The thesis is organized as follows. Related works are presented in Chapter 2. Fuzzy logic and related notions of t-norm, t-conorm, and residuum are explained in Chapter 3, as well as the as-is functioning of KENN. Chapter 4 describes the integration in the existing architecture of implication rules, alongside the implementation of residuum boost function. Experimental results are in Chapter 5. Here, the first two sections, 5.1 and 5.2, present the experiments done on Citeseer and Visual Relationship Detection datasets. The last two sections regard the experiments requiring reasoning: MNIST addition 5.3 and Visual Sudoku 5.4. Concluding remarks and future works are reported in Chapter 6.

2

Related work

Examples of recent works exploiting first-order logic in Neural-Symbolic systems are *Logic Tensor Network* (LTN) (Serafini and d'Avila Garcez, 2016)[4] and *Semantic Based Regularization* (SBR) (Diligenti *et. al*, 2017)[5]. In these methods, the satisfaction of logic rules is maximized during training. In KENN, logic rules become part of the classifier, of which predictions are manipulated according to the constraints. The idea of a system injecting prior knowledge into the structure of a neural network was first defined in *Knowledge Enhanced Neural Networks* (Daniele and Serafini, 2019)[1]. In this first version of KENN, the architecture worked just with propositional logic and was tested in multi-classification tasks, among which was Visual Relationship Detection (VRD). A further extension of KENN (Daniele and Serafini, 2022)[3] made it adaptable for relational data. The relational version exploited first-order logic to represent prior knowledge using also binary predicates, which are particularly useful for modeling links on graph-like domains. Relational KENN was tested on the *Citeseer* dataset, by representing publications as nodes and citations as links of a graph. The *residuum KENN* presented in this thesis extends the interpretation of conditional statements by adding the residuum semantic alongside the material implication rule used by existing relational KENN.

Another algorithm that implements the idea of having logic constraints to be part of the classifier, is *Iterative Local Refinement* (ILR) (Daniele, Van Krieken *et. al*, 2022)[6] which exploits *refinement functions* to find refined predictions for given logical formulas. ILR has been evaluated on the task of *MNIST addition*, setting a baseline for the evaluation of residuum KENN. Also *Probabilistic Soft Logic* (PSL) (Pryor *et al.*, 2022)[7] incorporates results from neural net-

works into a declarative templating language that uses first-order logical rules to perform logical reasoning. In (Augustine *et. al*, 2022)[8], PSL was combined with a neural image classifier (neuPSL), and then evaluated on Visual Sudoku puzzle classification.

The addition of the new residuum semantic allowed KENN to be evaluated on these last two tasks (MNIST addition and Visual Sudoku), initially unapproachable with relational KENN using material implication. Residuum KENN was also evaluated on the tasks carried out in the works presenting previous versions of KENN, in order to have a comparison with the architecture exploiting material implication rule.

3

Background

3.1 FUZZY LOGIC

Fuzzy logic is a form of many-valued logic in which the truth values may be any real number in the interval $[0, 1]$ (this range could be scaled for taking values outside), in such a way it can handle the concept of partial truth. Basic fuzzy logic is closely related to continuous t-norms, which play the role of truth functions of conjunction. Each continuous t-norm determines a semantics of fuzzy logic. Below are summarized the notions of t-norms, and more specifically the Gödel t-norm, from which is possible to define the related *t-conorm* and *residuum*, playing the role of disjunction and implication in Gödel semantics.

The basic reference is [9].

3.1.1 T-NORM

Definition 1. *A triangular norm (t-norm) is a function $\top: [0, 1]^2 \rightarrow [0, 1]$ such that for all $x, y, z \in [0, 1]$ the following properties are satisfied:*

- *Commutativity:* $\top(x, y) = \top(y, x)$,
- *Associativity:* $\top(x, \top(y, z)) = \top(\top(x, y), z)$,
- *Monotonicity:* $y \leq z \Rightarrow \top(x, y) \leq \top(x, z)$,
- *Boundary condition:* $\top(x, 1) = x$.

From Definition 1, it follows the definition of t-conorm.

Definition 2. If \top is a t-norm, then its dual t-conorm $\perp: [0, 1]^2 \rightarrow [0, 1]$ is given by:

$$\perp(x, y) = 1 - \top(1 - x, 1 - y).$$

\perp maintains all the properties of the t-norm, with the exception of *Boundary condition* which becomes: $\perp(x, 0) = x$.

In fuzzy logic, t-norms are the functions that build the semantics of the conjunctive operator \wedge , while the corresponding dual t-conorm represents the disjunctive operator \vee .

KENN refers to Gödel logic's semantics to interpret logic formulas. In Gödel logic the connective truth function \wedge is defined as:

$$t_{x \wedge y} = \top(x, y) = \min(x, y)$$

$\min(x, y)$ is a t-norm since satisfies all the properties listed above.

Then the definition of disjunctive operator \vee is the t-conorm of the *min* function:

$$t_{x \vee y} = 1 - \min(1 - x, 1 - y) = \max(x, y)$$

It is easy to see that the Gödel semantics, like other fuzzy semantics, agree with classical logic on the values 0, 1. For instance, consider the conjunction: $t_{0 \wedge 1} = \min(0, 1) = 0$ and $t_{0 \vee 1} = \max(0, 1) = 1$.

3.1.2 RESIDUUM

In two-valued logic, the implication is true iff the truth value of the antecedent is less then or equal to the truth value of the consequent. To generalize this concept in fuzzy logic, it can be said that the truth value of implication should be large when the truth value of the antecedent is not "too much larger" than the consequent. In *Metamathematics of Fuzzy Logic* (Hajek, 1998)[10], the truth function $x \Rightarrow y$ of fuzzy implication is defined by requiring it to be non-increasing in x and non-decreasing in y , moreover, to keep guaranteed the idea of *modus ponens* it is also required that, given truth degree of antecedent x and of function $x \Rightarrow y$, one should be able to compute a lower bound of truth degree of the consequent y .

More formally, considering a t-norm \top :

$$IF \quad z \leq t_{x \Rightarrow y}, \quad THEN \quad \top(x, z) \leq y$$

In this constraint, z is a possible candidate for $t_{x \Rightarrow y}$, and to make it as large as possible (to get a more powerful rule), also the opposite should be requested, and the whole would result in:

$$\top(x, z) \leq y \quad IFF \quad z \leq t_{x \Rightarrow y}$$

Saying that $t_{x \Rightarrow y}$ is the maximal z satisfying $\top(x, z) \leq y$.

From those assumptions, it follows the definition of *residuum*:

Definition 3. Let \top be a continuous t-norm. The residuum of \top is the unique operation $x \Rightarrow y$ such that $\forall x, y, z \in [0, 1] \quad \top(x, z) \leq y$ iff $z \leq t_{x \Rightarrow y}$, namely:

$$t_{x \Rightarrow y} = \max\{z \mid \top(x, z) \leq y\}.$$

It is easy to see that for each continuous t-norm \top and its residuum \Rightarrow :

- $x \leq y$ iff $t_{x \Rightarrow y} = 1$,
- $t_{1 \Rightarrow y} = x$.

Focusing on Gödel logic semantics, the Gödel implication is the residuum of Gödel t-norm. Considering $t_X, t_Y, t_Z \in [0, 1]$ truth values of predicates X, Y, Z , it results to be:

$$t_{X \Rightarrow Y} = \begin{cases} 1 & \text{if } t_X \leq t_Y \\ t_Y & \text{otherwise} \end{cases} \quad (3.1)$$

Proof: Assuming $t_X > t_Y$, if we consider the Gödel conjunction we have that $t_{X \Rightarrow Y} = \max\{t_Z \mid \min(t_X, t_Z) \leq t_Y\}$, then $\min(t_X, t_Z) = t_Y$ iff $t_Z = t_Y$.

3.2 KNOWLEDGE ENHANCED NEURAL NETWORK

Knowledge Enhanced Neural Networks (KENN) is a neural network model originally defined in (Daniele and Serafini, 2019[1]) that exploits prior knowledge about the domain of interest. The idea that inspired its creation was that having prior knowledge related to a specific domain could make predictions of a generic neural network more solid, in the sense of reducing the errors that would be unreasonable to the eyes of a domain-expert supervisor. For instance, if it's common knowledge that an animated screen means that the television is turned on, if an AI model spots that the screen is animated but wrongly predicts that it is turned off, then it should change its prediction relying on the fact that every animated screen is turned on. Since neural networks learn from experience, they are forced to have a knowledge of the domain restricted to the training data from which their experience is built, in this way, a poorly distributed training set could lead a neural network to unreasonable and unexplainable predictions. Even though we are living in the era of big data where it is much easier to access and elaborate enormous quantities of data and that facilitates the construction of enough big and well-distributed training sets, the problem presented of having consistent neural network predictions is still a case of interests in scenarios that require few-shot learning or, even more, in zero-shot learning problems. It is in those situations that KENN could be employed to boost the quality of the predictions of any kind of neural architecture that learns through empirical data.

As told in the Introduction, KENN is a Neural-Symbolic architecture, so, it is important to understand which are the bonds between neural and symbolic parts. As hinted by the name, the "knowledge", which is modeled in logical rules, is what should enhance the neural network predictions, and it does that without directly interfering with the neural network's computations. All the operations specific to the KENN architecture are carried out after the neural network computes its outcomes, so the structure as a whole can be seen as an underlying neural network with the logical components attached to its output layer, where the "enhancement" operations are performed according to the knowledge.

The first version of Knowledge Enhanced Neural Networks used propositional logic to model prior knowledge and turned out to have good empirical results on multi-label classification tasks. However, a strong limitation emerged when dealing with relational domains, where the translation of prior knowledge into propositional logic requests the usage of binary predicates. An updated version was then implemented by generalizing its functioning in such a way that it could deal with relational data (Daniele and Serafini 2023[3]). The new relational KENN

could then approach also tasks that involved relationships between data samples. In the following, we will refer to this version when talking about KENN.

In the KENN architecture, prior knowledge is translated into first-order logic formulas with at most binary predicates, which are given in input to the model. Logic rules must be written in Conjunctive Normal Form (CNF), i.e. a conjunction of one or more clauses, where a clause is a disjunction of literals.

Given the set of predicates $P = \{p_1, p_2, \dots, p_n\}$ associated to a clause c , then:

$$c = \bigvee_{i=1}^n l_i$$

Where l_i is either $p_i(x)$ or $\neg p_i(x)$.

Each clause composes a logic rule. Since it is assumed that the clauses are in CNF, every clause that defines the prior knowledge has to be satisfied. Also, all variables in each clause are assumed to be universally quantified.

As a logic rule example, consider:

$$\neg TVScreenAnimated(x) \vee TelevisionON(x)$$

stating that all televisions with an animating screen are turned on.

This is a unary clause and its groundings can be denoted by $c[a]$, which refers to the clause obtained by substituting the x variable with a constant a , meaning that $c[a]$ can now be assigned a truth value based on the traits of constant a .

A binary clause would instead appear as:

$$\neg Smoker(x) \vee \neg Friends(x, y) \vee Smoker(y)$$

This would state that if a person x is a smoker and he is a friend of another person y , then y is also a smoker. In this case, a grounding of this clause would need two constants a, b resulting in $c[a, b]$. The two constants allow the predicates composing the clause to assume values. For instance, in Boolean logic, if we know that a is a smoker and he's friend with b , but b is not a smoker then the clause assumes truth value $\neg 1 \vee \neg 1 \vee 0 = 0$ and therefore it would result false.

The smoker example supplied gives an idea of how to compute clause's truth values from their assigned groundings, and for simplicity, it was computed using Boolean algebra. However KENN operates with neural network preactivations (the values obtained before the output layer's activation function is applied, it will be explained later why this detail is important), so it deals with values belonging to \mathbb{R} making Boolean algebra inappropriate to this scenario. That's the reason why KENN makes use of Fuzzy Logic to compute truth values of clauses.

As previously said, KENN uses its neural network preactivations as clause groundings. To have a better overview on how it does it, suppose we have a NN for a multi-label classification task with classes $Q = \{q_1, q_2, \dots, q_m\}$ which returns an output $y \in [0, 1]^{n \times m}$ computed from preactivations $z \in \mathbb{R}^{n \times m}$, where n is the number of samples s_i . The knowledge K is defined by a set P of predicates and a set C of clauses. Consider the set of predicates P where each predicate $p_i(x)$ means " x belongs to class q_i ", and define a unary clause c containing a non-empty subset of them. The groundings of c : $\{c[s_1], c[s_2], \dots, c[s_n]\}$ are obtained by substituting the x variable with constants s_i , i.e. the samples taken in input by the NN. For every grounded clause c , the truth value of the predicates in c can be computed by substituting the preactivation values of the classes that are represented by those predicates. For instance, if c is composed by p_i and p_j ($i, j \leq m$), then the truth value of $c[s_1]$ would be computed from z_{1i} and z_{1j} .

The goal of KENN is to increment the truth value of every clause belonging to C , and it does so by calculating a residue δ , that is summed to NN's preactivation. The component that takes care of this operation is called *Knowledge Enhancer*.

Before describing the functioning of the Knowledge Enhancer (KE), it is essential to define the set of functions that increase the values of a t-conorm called *t-conorm boost function*, which are responsible for computing the residue δ . In the following, we report the theory of boosting functions as defined in (Daniele and Serafini, 2019[1]).

3.2.1 T-CONORM BOOST FUNCTION

Definition 4. A function $\delta : [0, 1]^n \rightarrow [0, 1]^n$ is a *t-conorm boost function (TBF)* if and only if $\forall n \in \mathbb{N}, \forall \mathbf{t} \in [0, 1]^n$:

$$0 \leq t_i + \delta(\mathbf{t})_i \leq 1, \quad 0 \leq i < n$$

From the *Monotonicity* property of t-conorm follows the following proposition:

Proposition 1. For every t-conorm \perp and every TBF δ , $\perp(\mathbf{t}) < \perp(\mathbf{t} + \delta(\mathbf{t}))$.

Given that TBFs are generic functions that keep values inside the space $[0, 1]^n$ while increasing the value of a t-conorm (which translates into augmenting the truth value of a disjunction), they are the perfect tool for proposing changes on the groundings of a clause.

Considering the set of all possible TBFs, KENN uses the TBF that increases the outcome of a t-conorm by applying the minimum modification that guarantees the given improvement.

In a more formal way:

Definition 5. Consider Δ the set of all TBFs, a norm $\|\cdot\|$ and a t-conorm \perp , a function $\delta \in \Delta$ is minimal iff:

$$\begin{aligned} \forall \delta' \in \Delta \quad \forall n \in \mathbb{N} \quad \forall \mathbf{t} \in [0, 1]^n \\ \|\delta'(\mathbf{t})\| < \|\delta(\mathbf{t})\| \quad \rightarrow \quad \perp(t_i + \delta'(\mathbf{t})) < \perp(t_i + \delta(\mathbf{t})). \end{aligned}$$

Proposition 2. Every $\delta^f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ defined as:

$$\delta^f(\mathbf{t})_i = \begin{cases} f(\mathbf{t}) & \text{if } i = \operatorname{argmax}_{j=1}^n t_j \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

where f is a function such that $0 \leq f(\mathbf{t}) \leq 1 - \max_{j=1}^n t_j$, is a minimal TBF for the Gödel t-conorm and l_p norm.

Proof:

Gödel t-conorm $\perp(\cdot)$ is defined as $\perp(\mathbf{t}) = \max_{i=1}^n (t_i)$; and l_p - norm is defined as $\|\mathbf{t}\|_p = (\sum_{k=1}^n |t_k|^p)^{\frac{1}{p}}$. Suppose that $\delta \in \Delta$ is such that:

$$\|\delta(\mathbf{t})\|_p < \|\delta^f(\mathbf{t})\|_p$$

If $j = \operatorname{argmax}_{k=1}^n (t_k + \delta(\mathbf{t})_k)$, we can derive:

$$\perp(\mathbf{t} + \delta(\mathbf{t})) = t_j + \delta(\mathbf{t})_j$$

and, if $i = \operatorname{argmax}_{k=1}^n t_k$, we have that

$$\perp(\mathbf{t} + \delta^f(\mathbf{t})) = t_i + f(\mathbf{t})$$

Since $t_i \geq t_j$, we just need to demonstrate that $\delta(\mathbf{t})_j < f(\mathbf{t})$. Notice that:

$$\delta(\mathbf{t})_j = (|\delta(\mathbf{t})_j|^p)^{\frac{1}{p}} \leq \|\delta(\mathbf{t})\|_p < \|\delta^f(\mathbf{t})\|_p$$

Since $\delta^f(\mathbf{t})$ changes only the value of the i^{th} component of \mathbf{t} we have that $\|\delta^f(\mathbf{t})\| = f(\mathbf{t})$.

3.2.2 KNOWLEDGE ENHANCER

The Knowledge Enhancer is the component that implements the minimum TBF introduced in the previous section. It is responsible for enhancing the outcomes received from the neural network by proposing changes δ computed from the set of clauses belonging to K . Differently from the theory proposed in Section 3.2.1, the KE operates on the preactivation z of the final layer.

The final residue δ is the result of the sum of all δ_c computed individually for each clause:

$$y = \sigma(z + \delta), \quad \delta = \sum_{c \in C} \delta_c$$

here, y is the matrix containing the final predictions for each sample, z is the preactivation matrix and C is the set of clauses. To compute each δ_c , KE generates further submodules called *Clause Enhancers* (CE), one for each clause: these components are the ones that actually evaluate the clauses and decide which predicate to modify in order to increase their truth value. Inside each CE, a differentiable approximation of the t-conorm boost function, defined in section 3.2.1, is computed on the reference clause.

3.2.3 CLAUSE ENHANCER

In the Clause Enhancer, the notion of the t-conorm boost function has been adapted to handle NN's predictions. Given that from the definition of TBF, the increased value should stay in the interval $[0, 1]$, it is impossible to apply a liner function f directly to the output $\mathbf{y} \in \mathbb{R}^n$ of the NN. Note that \mathbf{y} is computed from preactivations $\mathbf{z} \in \mathbb{R}^n$ of the last layer through the non-linear function $\sigma: \mathbb{R}^n \rightarrow [0, 1]^n$:

$$y_i = \sigma(z_i) = \frac{1}{1 + e^{-z_i}}$$

If the values of \mathbf{z} are never decreased, then the constraint of Definition 4 holds.

So, as we know from 3.2, the minimal increment for the Gödel t-conorm is δ^f , we can use that

function to increase the value of \mathbf{z} , obtaining that

$$y = \sigma(z) \leq \sigma(z + \delta^f(z))$$

since function σ is monotonic. And considering the function δ^g such that:

$$\delta^g(y) = \sigma(z + \delta^f(z)) - \sigma(z)$$

We obtain

$$y \leq y + \delta^g(y) = \sigma(z + \delta^f(z)) \in [0, 1]^n$$

showing that $\delta^g(y)$ is a minimal TBF.

The CE doesn't directly compute $\delta^g(y)$ but instead computes $\delta^f(z)$ which are the proposing changes δ_c related to a single clause. Since the clause c is a disjunction of literals, the truth value of each of its groundings can be computed from \mathbf{z} by applying the Gödel t-conorm \perp which can be augmented by increasing the maximum value z_{ij} for each row z_i , where j is the index of a predicate of the clause. After applying the activation function σ , the described operations would be equal to apply the TBF $\delta^g(y)$ directly to the predictions \mathbf{y} .

3.2.4 RELATIONAL KENN ARCHITECTURE

Now that has been explained the functioning of Clause Enhancer, the process carried out by KENN can be described as a whole. Since KENN can also operate in relational domains, the Knowledge Enhancer must perform a distinction between unary and binary predicates, in such a way it can elaborate and update their predicate values separately. The theoretical notions of t-conorm and boosting functions don't require any adjustment in order to be extended to a relational domain. The split between unary and binary logic formulas is done before calling the Clause Enhancer which stays unaware of the kind of predicates it is working on.

To understand how the whole process is performed, let's consider the matrix $\mathbf{z} \in \mathbb{R}^{n \times m}$ computed by the underlying NN, which contains the class preactivation values for each of the n samples s_i . From a logical perspective, \mathbf{z} contains the truth values of predicates (classes) for each grounding (sample).

The boosting function used by the CE isn't the optimal TBF for the Gödel t-conorm (Daniele and Serafini, 2019[1]), but instead, the CE applies the *softmax* function, which is a continu-

ous and differentiable approximation of the minimal TBF. Indeed, $\text{softmax}(z)$ still increases the highest value more than the others. However, all the literals are increased, and when two literals have close values, it produces similar improvements to both.

The important thing to underline here is that the TBF approximation is differentiable, allowing for the application of back-propagation to the entire architecture. Moreover, KENN can learn the truthfulness of a logic rule by assigning it a learnable weight w_c . In this way, if it turns out that a clause isn't respected in most of the target's groundtruths, then the model will learn to ignore that specific clause by dropping its w_c to 0.

Asserted that, the computation of a $\delta_{c[s_i]}$ relative to the grounded clause $c[s_i]$ containing predicates $P_j(s_i)$ results to be:

$$\delta_{c[s_i], P_j(s_i)} = \begin{cases} w_c \cdot \text{softmax}(z_i)_j & \text{if } P_j(s_i) \in c[s_i] \\ -w_c \cdot \text{softmax}(z_i)_j & \text{if } \neg P_j(s_i) \in c[s_i] \end{cases} \quad (3.3)$$

Before computing deltas, the relational KENN performs a pre-elaboration of the \mathbf{z} taken from the NN which is divided in \mathbf{z}_U and \mathbf{z}_B containing unary and binary predicates preactivations respectively. Note that, it is not strictly necessary that all predicates must be taken from the NN's output. In Chapter 5, it will be shown how some predicates can be provided directly as features of the model. Since relational KENN handles both unary clauses and binary clauses, it is also necessary to consider them separately. For the unary clauses in K_U the changes calculated for a single grounded predicate $P_j(s_i)$ will result in:

$$\delta_{K_U, P_j(s_i)} = \sum_{\substack{c \in K_U \\ P_j(x) \in c}} \delta_{c[s_i], P_j(s_i)} \quad (3.4)$$

To improve binary clauses, i.e. the ones that contain at least one binary predicate, the KE has a slightly different behavior. Indeed, alongside preactivations matrixes \mathbf{z}_U and \mathbf{z}_B , it needs two lists of integers i_x, i_y for each binary predicate indicating which samples are related through the binary relations. This information is used to perform a join operation between \mathbf{z}_U and \mathbf{z}_B , resulting in the joined matrix \mathbf{z} , where the groundings of the pairs of predicates related by a binary predicate are considered in the same line \mathbf{z}_i , alongside the grounding of the binary predicate referring to them.

This resulting matrix \mathbf{z} is fed into the CE, which will treat \mathbf{z} as if it contains just unary predicates.

When δ_c is returned by CE, the KE does a group-by operation that gathers back the deltas for unary predicates and binary predicates. Note that, in this situation, unary predicates' deltas are doubled since they refer both to x and y variables that appear on the binary clause. The proposed changes for a unary predicate $P_j([s_i])$, result in:

$$\delta_{K_B, P_j(s_i)} = \sum_a \left(\sum_{\substack{c \in K_U \\ P_j(x) \in c}} \delta_{c[s_i, a], P_j(s_i)} + \sum_{\substack{c \in K_U \\ P_j(y) \in c}} \delta_{c[a, s_i], P_j(s_i)} \right) \quad (3.5)$$

The deltas for a binary predicate $P_j(x, y)$ are instead easier to compute since any possible grounded atom can be found in only one corresponding grounding $[s_{i_1}, s_{i_2}]$ of each clause:

$$\delta_{K_B, P_j(s_{i_1}, s_{i_2})} = \sum_{\substack{c \in K_B \\ P_j(x, y) \in c}} \delta_{c[s_{i_1}, s_{i_2}], P_j(s_{i_1}, s_{i_2})}$$

After the computations of changes, all deltas are summed together and the resulting prediction $y_{P_j(s_i)}$ for the grounded unary predicate P_j are:

$$y_{P_j(s_i)} = \sigma(z_{ij} + \delta_{K_U, P_j(s_i)} + \delta_{K_B, P_j(s_i)})$$

Following the same reasoning, for a grounded binary predicate P_j the resulting prediction would be $y_{P_j(s_{i_1}, s_{i_2})}$

$$y_{P_j(s_{i_1}, s_{i_2})} = \sigma(z_{ij} + \delta_{K_B, P_j(s_{i_1}, s_{i_2})}).$$

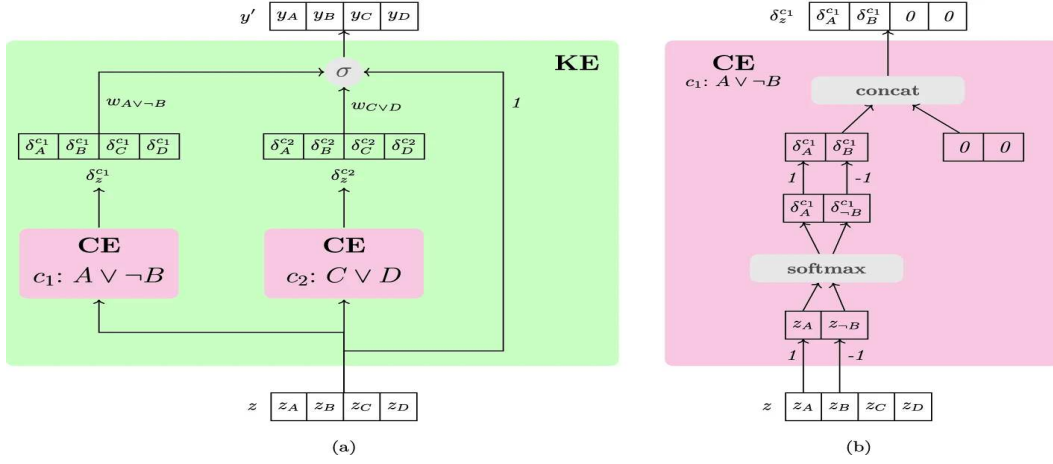


Figure 3.1: An overview of the complete architecture of KENN presented by Daniele and Serafini [1]. The figure shows in green the structure of the Knowledge Enhancer while in pink it is shown the Clause Enhancer. In this example, the KE has two clauses c_1, c_2 to apply on the preactivation $z = [z_A, z_B, z_C, z_D]$. On the right is shown only the CE that handles clause $c_1 : A \vee \neg B$ involving only predicates A, B corresponding to the first two columns of z . At the end of computations the CE proposes changes $\delta_A^{c_1}$ and $\delta_B^{c_1}$ to be added to columns z_A and z_B

3.2.5 EXPERIMENTAL RESULTS OF KENN

The presented KENN model has been tested on different multi-class classification datasets. The experiments carried out by (Daniele, Serafini 2019[1]) on Yeast (Elisseeff and Weston 2001), Emotions (Trohidis et al. 2008), and VRD (Lu et al. 2016) datasets showed that the prior knowledge inserted in KENN could improve the predictions of the underlying Logistic Regression model.

Further, in (Daniele, Serafini 2022[3]) the extended KENN for relational domains was tested on Citeseer dataset [11] on top of a multi-layer perceptron. This experiment deserves a deeper explanation, since it is not only an example of how to shape graph-like relations into binary clauses, but also is an important baseline that will be approached again in the following chapters by the new *Residuum* KENN architecture.

The Citeseer dataset consists of 3312 scientific publications classified into one of six classes, while the citation network consists of 4732 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary.

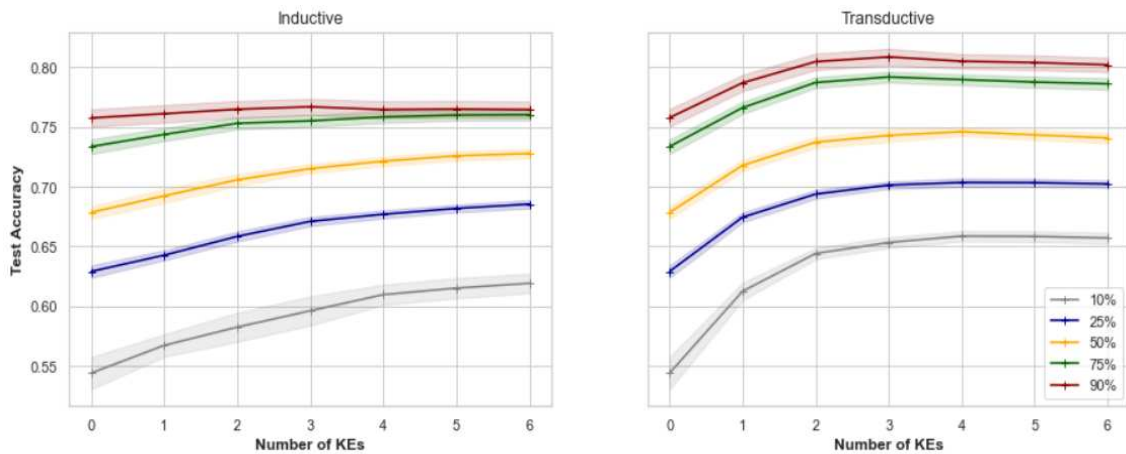
In the experiment the NN used was a dense network with 3 hidden layers, each with 50 hidden nodes and ReLU activation function. The knowledge consisted of six rules obtained by substi-

tuting the topic T in $\neg T(x) \vee \neg Cite(x, y) \vee T(y)$ with all the classes. This codifies the idea that papers cite works that are related to them. From a graph point of view, the predicates $T(x)$ represent the class of each node, while the binary predicate $Cite(x, y)$ assumes true value 1 if there is an edge between nodes x and y and 0 otherwise.

Two slightly different experiments were made, one called "Inductive" where the edges that go from samples belonging to different data sets (e.g. one in the training set and the other in validation) were removed, while the other called "Transductive" meaning that all edges of the original graph are kept in the training and evaluation of the model.

The efficacy of KENN was evaluated on different dimensions of training data and by changing the amount of KE layers.

The results obtained are the following:



We can see how just by adding a single KE layer, KENN outperforms the NN in each subset of the training set, and that three KE layers were a sufficient amount for having good results in relation to the complexity of the model. The importance of having multiple KE layers was discussed in [3].

Another important thing to notice here is that the injected prior knowledge could, in some sense, play the role of training data. Indeed, the results in terms of accuracy score carried out by the model trained on 25% of the training set with 0 KE layers (i.e. just the NN) were outperformed by the KENN model made of 3 KE using just the 10% of the knowledge. This means that adding prior knowledge into an NN, beyond improving its results, also facilitates its learning procedure by making it able to learn the same task even if a smaller number of samples is seen during training.

4

Residuum enhanced neural networks

The functioning of KENN explained in the previous chapter focused on increasing truth values of disjunctive clauses, assuming that to represent an implication rule it would be sufficient to replace it by the disjunction between the negated antecedent rule and the consequent one. This replacement in classical logic is equivalent to the actual implication. However, since KENN makes use of fuzzy logic semantics, using the material implication isn't always the best way to represent conditional statements. Indeed, fuzzy implications are distinguished in two sets, the so-called *S-implications* are derived by generalizing the material implication in t-conorms which is the current interpretation carried out in KENN, while *R-implications* are induced by the *residuum* of t-norms. It turned out that in many scenarios the use of R-implications semantics could allow to inject a "direction" on conditional statement rules, Daniele *et. al* (2023)[6]. This chapter explains the implementation of residuum semantics and the related residuum boost function in KENN architecture. By showing how the functioning of Knowledge Enhancer and Clause Enhancer modules have been adapted to handle the residuum boost function alongside the current TBF. This new architecture of KENN opened the way for approaching new tasks and in some cases to improve the results obtained with the old version.

4.1 RESIDUUM BOOST FUNCTION

For Gödel residuum operation we define as follows the *residuum boost function* (RBF). By considering the case $x \Rightarrow y$, where x and y are two predicates, and t_x, t_y their respective truth

values.

Definition 6. A function $\delta : [0, 1]^2 \rightarrow [-1, 1]^2$, such that:

- $-1 \leq \delta(t_x, t_y)_x \leq 0$
- $0 \leq \delta(t_x, t_y)_y \leq 1$

is a residuum boost function.

We set the following notation: $R(t_x, t_y) = t_{x \Rightarrow y}$, $\delta(t_x, t_y)_x = \delta_x$

Proposition 3. For every RBF δ , $\forall t_x, t_y \in [0, 1]$:

$$R(t_x + \delta_x, t_y + \delta_y) \geq R(t_x, t_y)$$

Proof:

$$\begin{aligned} R(t_x + \delta_x, t_y + \delta_y) &= \begin{cases} 1 & \text{if } t_x + \delta_x \leq t_y + \delta_y \\ t_y + \delta_y & \text{otherwise} \end{cases} & (4.1) \\ &= \begin{cases} 1 & \text{if } t_x - t_y \leq \delta_y - \delta_x \\ t_y + \delta_y & \text{otherwise} \end{cases} \\ &\geq \begin{cases} 1 & \text{if } t_x - t_y \leq 0 \\ t_y & \text{otherwise} \end{cases} = R(t_x, t_y) \end{aligned}$$

Note that, if $R(t_x, t_y) = 1$ than $\delta_y - \delta_x \geq 0 \geq t_x - t_y$, and $R(t_x + \delta_x, t_y + \delta_y) = 1$. Moreover, if $R(t_x, t_y) < 1$, then $R(t_x, t_y) = t_y$ and $R(t_x + \delta_x, t_y + \delta_y)$ is either 1 or $t_x + t_y$, both greater t_y .

In the other case: $\delta_y \geq 0 \implies t_y + \delta_y \geq t_y$

In paragraph 3.1.1, it has already been said that KENN uses the TBF that increases the outcome of formulas by applying the minimum modification that guarantees a given improvement. In

the original case of disjunctive clauses the *minimal* TBF was shown in 3.2. The same reasoning was adopted for residuum, in this case the RBF to look for is the one that minimizes the modifications to be done on the Gödel residuum function to refine its outcome on a given quantity.

Definition 7. Consider Δ the set of all RBFs, Gödel t -norm inducted residuum R . Given norm $\| * \|$, a function $\delta \in \Delta$ is a minimal Gödel residuum boost function iff:

$$\forall \delta' \in \Delta \quad \forall t_x \in [0, 1] \quad \forall t_y \in [0, 1]$$

$$\|\delta'(t_x, t_y)\| < \|\delta(t_x, t_y)\| \quad \rightarrow \quad R(t_x + \delta'_x, t_y + \delta'_y) < R(t_x + \delta_x, t_y + \delta_y)$$

To find minimal Gödel residuum boost functions we consider separately three cases:

1. $t_x \leq t_y$
2. $t_x > t_y$ and $t_x + \delta_x > t_y + \delta_y$
3. $t_x > t_y$ and $t_x + \delta_x \leq t_y + \delta_y$

Case 1:

The rule is already satisfied. The truth value of implication (residuum) is already 1 so the obvious solution is to not change the prediction: $\delta_x = \delta_y = 0$.

Proposition 4. The RBF δ is minimal if $\delta_x = \delta_y = 0$ when we are in Case 1.

Proof:

Note that $\|\delta\| = 0$. Meaning that it can't exist $\|\delta'\| < \|\delta\|$, so

$$\|\delta'(t_x, t_y)\| < \|\delta(t_x, t_y)\| \quad \rightarrow \quad R(t_x + \delta'_x, t_y + \delta'_y) < R(t_x + \delta_x, t_y + \delta_y)$$

is always satisfied because the assumption is always false.

Case 2:

Since $t_x > t_y$ and also $t_x + \delta_x > t_y + \delta_y$, we have that the truth value of implication is equal to the truth value of the consequent before and after summing δ , and it does not depend on the antecedent. As such, the δ_x should be equal to zero.

Proposition 5. For every minimal RBF δ it holds $\delta_x = 0$ in Case 2.

Proof:

Suppose otherwise: $\delta_x < 0, \delta_y \geq 0$.

In such a case, it is always possible to define δ' such that $\delta'_x = 0, \delta'_y = \delta_y$. Note that, $\|\delta'\| < \|\delta\|$, while the truth value of implication is $t_y + \delta_y$ in both cases. Hence, δ is not minimal.

Case 3:

Since $t_x + \delta_x \leq t_y + \delta_y$, the truth value of the implication after applying the RBF is 1. In order to be minimal, we need a value of δ such that by reducing the norm of δ we obtain a value smaller than one. As a consequence, we require $t_x + \delta_x$ to be equal to $t_y + \delta_y$ (i.e. $t_x + \delta_x$ cannot be strictly smaller than $t_y + \delta_y$).

Proposition 6. Every minimal RBF, in Case 3., is a δ such that $t_x + \delta_x = t_y + \delta_y$

Proof:

Assume otherwise: $t_x + \delta_x < t_y + \delta_y$, meaning that $t_x - t_y < \delta_y - \delta_x$

Note that, by assumption of Case 3., $t_x > t_y$. We can always define δ' such that:

$$\begin{aligned} 0 \geq \delta'_x \geq \delta_x, \quad 0 \leq \delta'_y \leq \delta_y \\ \delta'_y - \delta'_x = t_x - t_y \end{aligned}$$

Note that the truth value of the implication is still equal to one, while the norm is reduced:

$$\|\delta\|_p - \|\delta'\|_p = |\delta_x|^p - |\delta'_x|^p + |\delta_y|^p - |\delta'_y|^p \geq 0$$

As a consequence, δ cannot be minimal.

Proposition 6 can be intuitively explained by saying that, to bring implication truth value to 1 using a minimal RBF, it is sufficient to stop when antecedent and consequent truth values are equal, without further modifications.

At this point, we know that in Case 3 we need to reduce t_x and t_y in such a way that the two become equal. We need to find such a change in order to have the minimal norm. This choice depends on the selected norm.

For instance, with L_2 - norm, the optimal solution would consist on increasing/decreasing both equally: $\delta_x = -\delta_y$. However, such a solution would not be continuous for all the cases, since in Case 2 we increase only the consequent.

Following (Daniele, van Krieken *et al*, 2022)* [6], we choose L_1 - norm, where every

*In their case, on the definition of a different concept consisting in minimal refinement functions.

solution which implies $\delta_y - \delta_x = t_x - t_y$ has the same value for the norm. As a consequence, the solution $\delta_x = 0, \delta_y = t_x - t_y$ is minimal for $L_1 - norm$.

Note that, with this definition, the RBF behaves exactly as in Case 2, making the function continuous.

Putting all together, we defined the minimal RBF $\delta^r : [0, 1]^2 \rightarrow [-1, 1]^2$ for $L_1 - norm$ and first-order Gödel logic as:

$$\delta^r(t_x, t_y) = (0, r(t_x, t_y))$$

where function $r(t_x, t_y) : [0, 1]^2 \rightarrow [0, 1]$ is such that:

$$\begin{cases} 0 \leq r(t_x, t_y) \leq t_x - t_y & \text{if } t_x \geq t_y \\ 0 & \text{otherwise} \end{cases} \quad (4.2)$$

In our extension of KENN, conditional formulas are in general written in the form of a conjunction of literals implying a disjunction of literals:

$$\bigwedge_i^n X_i \Rightarrow \bigvee_j^m Y_j$$

The truth values of the set of predicates X, Y are $\mathbf{x} \in [0, 1]^n$ and $\mathbf{y} \in [0, 1]^m$ respectively. Having the truth value of the formula to be:

$$R(\top(\mathbf{x}), \perp(\mathbf{y})) = R(t_{\mathbf{x}}, t_{\mathbf{y}})$$

By applying a minimal RBF δ^r to the formula we obtain:

$$R(t_{\mathbf{x}}, t_{\mathbf{y}} + r(t_{\mathbf{x}}, t_{\mathbf{y}}))$$

Note that our goal is to increase the satisfaction of the consequent, which is now defined as a disjunction of literals, while our definition of RBF considers only the case with a single literal as a consequent. However, the minimal change to increase a disjunction is already managed by

KENN, since it corresponds to the output of a TBF. So we can increase the truth value t_y by applying the TBF $\delta^f : \mathbb{R}^m \rightarrow \mathbb{R}^m$ defined as follows:

$$\delta^f(\mathbf{y})_i = \begin{cases} r(t_x, t_y) & \text{if } i = \operatorname{argmax}_{j=1}^m y_j \\ 0 & \text{otherwise} \end{cases} \quad (4.3)$$

since $r(t_x, t_y) \leq t_x - t_y$ and $t_x - t_y \leq 1 - \max_{j=1}^m y_j$ [†], from Proposition 2 we know that δ^f is a minimal TBF.

Finally, note that our new implementation is a generalization of classical KENN, since a disjunction rule $\bigvee_i l_i$, can always be expressed as $T \rightarrow \bigvee_i l_i$ with T corresponding to true (one).

4.2 R-IMPLICATION KNOWLEDGE ENHANCER

Before getting into the changes made on Knowledge Enhancer, it is worth underlining the mathematical differences derived by interpreting the implication as a t-norm induced residuum instead of applying the material implication rule on the t-conorm. First of all, in the case of $x < y$, i.e. the antecedent having less truth degree than the consequent, the S-implication computed from the t-conorm assigns to $x \Rightarrow y$ truth degree $\max((1 - x), y)$. While the R-implication assigns always a truth value of 1 in such cases. Moreover, when $x > y$, the S-implication takes values higher than y (take $x = 0.6, y = 0.3$, the S-implication $x \Rightarrow y$ truth value would be 0.4, while residuum $x \Rightarrow y$ would be 0.3). The conclusion is that given the truth values of antecedent x and $x \Rightarrow y$ using S-implication, is not always possible to compute a lower bound for consequent y , hence not respecting deductive *modus ponens* argument of classical logic.

Inside the KE, logic rules containing R-implication statements are treated alongside disjunctive clauses, the main difference resides in the way the Clause Enhancer increases the truth value of those rules. Indeed, it has been implemented an ad hoc *Residuum Clause Enhancer* (RCE), which makes use of RBF to increase the truth values of R-implication formulas.

It is important to note that, for every implication formula $\phi \Rightarrow \omega$, having ω to be written in CNF leads to rewriting the formula as $\bigwedge_i (\phi \Rightarrow \omega_i)$ where ω_i is the i -th clause of ω . On the other hand, having ϕ written in disjunctive normal form (DNF) allows us to equivalently write $\phi \Rightarrow \omega$ as $\bigwedge_i (\phi_i \Rightarrow \omega)$ where ϕ_i is the i -th conjunction. As it has been explained in the

[†]note that $t_y = \max_{j=1}^m y_j$ by definition of Gödel t-conorm

first chapter, KENN aims to validate all logic rules injected during the definition of the model, meaning that, given a conjunction of implication formulas $\bigwedge_i (\phi_i \Rightarrow \omega_i)$ each $\phi_i \Rightarrow \omega_i$ can be seen by the architecture as different formulas to be independently manipulated. This means that the irreducible implication formulas that the KE has to handle are written in the form $\bigwedge_i \phi_i \Rightarrow \bigvee_i \omega_i$, such that on the antecedent appears just a conjunction of literals while in the consequent a disjunction of literals.

4.3 RESIDUUM CLAUSE ENHANCER

In this new architecture, the KE sums together all the modifications suggested by all clause enhancers created for each logic rule specified in the definition of the knowledge. At this level of computation, KE doesn't treat the deltas differently based on whether they come from implication formulas or not. Indeed, the updating rules 3.4 and 3.5 don't change in the new *Residuum* KENN, where the different treatment reserved to the two kinds of logic rules, disjunctive clauses and implication statements, is carried out at CE level. It is the task of the Residuum Clause Enhancer to improve the truth degree of implication formulas, (even though it doesn't handle clauses anymore, the word clause has been kept in the name to underline the similarity with original clause enhancer), and it is the duty of KE to create a dedicated CE for each type of logic rule in the injected knowledge. As done for disjunctive clauses, also RCEs exploit boost functions to improve the validity score of implication formulas, by using an RBF.

To show how modifications are proposed by the RCE, consider a matrix $\mathbf{z} \in \mathbb{R}^{n \times m}$ representing the m classes preactivations computed by the underlying NN for the n samples s_i , the computation of a $\delta_{c[s_i]}$ relative to the grounded implication formula $c[s_i]$ having antecedent predicates $A \subseteq \{P_1, \dots, P_m\}$ and consequent predicates $Q \subseteq \{P_1, \dots, P_m\}$, is defined as:

$$\delta_{c[s_i], Q_j(s_i)} = \begin{cases} \max\left(\min\left(w_c, \min_{w=1}^{|A|} (A_w(s_i)) - Q_j(s_i)\right), 0\right) & \text{if } j = \operatorname{argmax}_{i=1}^{|Q|} Q_i(s_i), Q_j \in c \\ -\max\left(\min\left(w_c, \min_{w=1}^{|A|} (A_w(s_i)) - Q_j(s_i)\right), 0\right) & \text{if } j = \operatorname{argmax}_{i=1}^{|Q|} Q_i(s_i), \neg Q_j \in c \\ 0 & \text{otherwise} \end{cases} \quad (4.4)$$

where w_c is the learnable parameter relative to formula c .

Note that, this definition respects the notion of the residuum boost function, since when the truth value of antecedent predicates (computed by minimum function, i.e. Gödel t-norm for

conjunction) is lower than the maximum of the consequent (Gödel t-conorm), then $\min_{w=1}^{|A|} (A_w(s_i)) - Q_j(s_i) < 0$ and the proposed modification is 0. Otherwise, the modification is the minimum between the difference of their truth values and the weight w_c , which by definition respects the constraint exposed when defining minimal residuum boost function (4.2). In other words, the task of RCE is to bring the truth degree of the consequent side of implication closer to that of the antecedent side, if it is not already greater.

To conclude, the new architecture exploits the residuum instead of material implication to interpret conditional statements, and it does so by adding a new component, the Residuum Clause Enhancer. This component implements a different boost function specific to improving residuum outcome. The KE behavior doesn't change in terms of computations, it continues to sum up all modifications returned by the whole set of clause enhancers without concern about which type of boost function they carry out, the only difference is that now KE must create the specific sub-components according to which logic rule it has to validate.

The following chapter will focus on the experiments done to empirically verify what are the advantages of adding this interpretation for implication rules and in which situations it is particularly useful. Since this kind of interpretation makes use of a boost function that modifies just the consequence, it is possible to tell the model which predicates are the ones that could be modified and which ones can instead be assumed. It is going to be shown how this innovation made the model applicable to tasks that couldn't be approached before, presenting results comparable to other neuro-symbolic methods.

5

Experiments

In this chapter will be discussed the experiments done to test the capabilities of the implemented *Residuum KENN* model. The chapter is divided into four sections. The first two sections are dedicated to experiments already approached by KENN, consisting of the Citeseer dataset and Visual Relationship Detection for spatial predicates. These first two tasks are on the set of experiments that require prior knowledge to correct predictions. Here, the aim of KENN is to supervise NN predictions by applying modifications to the predicates according to the logic rules.

Going further, in the following two sections are presented experiments not yet approached by KENN: *MNIST addition* and *Visual Sudoku*. These tasks are distinct from the previous two. The objective is to perform further reasoning starting from the information returned by the NN, which remains unchanged by the knowledge. Here the neural and symbolic components of KENN focus on different parts of the task: the underlying NN takes care of low-level perception, while the KE layers perform the reasoning.

The results have shown that the residuum semantics is particularly beneficial compared to material implication when it comes to approaching the tasks requested by the latter type of experiments.

5.1 CITESEER

The experiment presented is carried out on Citeseer dataset[11]. Section 3.2.5 has already briefly introduced the task and showed some results obtained by applying material implication KENN.

The results obtained with the old architecture showed that exploiting logic helped to get a significant improvement, especially when the comparison with a simple multi-layer perceptron was done reducing the training set size, meaning that, prior logic could reduce the model's need to elaborate a significant amount of training samples in order to reach acceptable classification scores.

5.1.1 DATASET

The dataset contains a selection of papers, where each of them is represented by bag-of-words vectors indicating the absence or presence of a word in the paper's text (with the paper we indicate all the information regarding it, which comprises the header, the abstract, citations and full text). The word vocabulary was built by gathering all the words present in the papers composing the dataset, removing stopwords and words with document frequency less than 10. The result of these procedures is a vocabulary of size 3703 unique words. A document is encoded by a vector $\mathbf{d} \in \{0, 1\}^{3703}$:

$$\mathbf{d} = [d_1, d_2, \dots, d_{3703}]$$

where each d_i gets value 1 if vocabulary word i is in document \mathbf{d} and 0 otherwise.

The dataset contains, alongside documents encoding, the representation of the graph indicating citations among them. In the graph, the nodes represent documents while the edges indicate the citation relationship between them. The papers are a total of 3312 and were selected by asserting that each of them would be cited by at least one other paper, meanwhile, the directed edges going from paper A to paper B modeling the relation *paper A is cited by paper B* are a total of 4732.

The papers are distinguished in 6 classes: Artificial intelligence (AI), Information retrieval (IR), Machine learning (ML), Agents, Human-computer interaction (HCI), and Database (DB), and the task is to map each paper to the correct class.

5.1.2 EXPERIMENT

The experiment was firstly carried out by modifying the one presented in the work Knowledge Enhanced Neural Network for Relational Domains[3], converting material implication logical rules into residuum semantics.

The original experiment aimed to predict the paper classes by merging the knowledge of their word composition and citations graph. The idea behind the logic inserted into KENN model was that papers belonging to the same class cite works that are related to them. In first-order logic, this idea translates into:

Given a topic $T \in \{Agents, AI, ML, DB, HCI, IR\}$ then:

$$\neg T(x) \vee \neg Cite(x, y) \vee T(y)$$

The same reasoning was applied when testing residuum KENN on the same dataset. While the experimental setup was kept the same, the knowledge was simply translated in such a way it could be handled by Residuum Clause Enhancers. The modified prior knowledge to be injected into the new architecture appears as:

$$T(x) \wedge Cite(x, y) \rightarrow T(y)$$

representing the same idea if considering the material implication rule of classical logic, but leading to different interpretations in a scenario that exploits fuzzy logic semantics.

As previously said, the experiment was initially carried out by the same setup as in the relational KENN paper. The dataset has been split into training, validation, and test sets. The performance of the model was analyzed by varying the dimensions of the training set. Since the whole dataset was split into training and testing, it came up the opportunity to test the model performance with different approaches of considering train and test sub-graphs. Indeed, when creating dataset subsets, each containing exclusive documents, the edges that go from documents belonging to different sets could be either removed or kept during training and inference processes. The procedure performed on the dataset resulted from removing edges that connect documents that don't belong to the same set has been called *inductive* learning, while the opposite is named *transductive*.

The performance of residuum KENN has been evaluated on both tasks separately, using a

dense multi-layer perceptron (MLP) as underlying NN with 3 hidden layers, each with 50 hidden nodes and ReLU activation function. The NN takes in input the bag-of-words vectors representing each document, while it doesn't elaborate the edges (i.e. citations information) among them. The edges are injected into the relational KENN model by giving as input the indexes of documents (nodes) that cite each other, as explained in section 3.2.4. The KE creates the groundings by considering the pairs of connected nodes and improving the truth value of the logic rules where the binary predicate $Cite(x, y)$ assumes truth value 1, so only when the two nodes are linked in the graph.

The experiment was run by selecting as training set the 10%, 25%, 50%, 75%, and 90% of dataset's nodes and using three layers of KEs, since in the original experiment it empirically turned out to be a sufficient amount. For each training subset the learning and evaluation procedures were repeated 50 times and the average between them was taken as a reference result in order to better estimate the expected value of evaluation accuracy, which is a random variable given the random nature of the training procedure.

The outcome is summarized in the following figures:

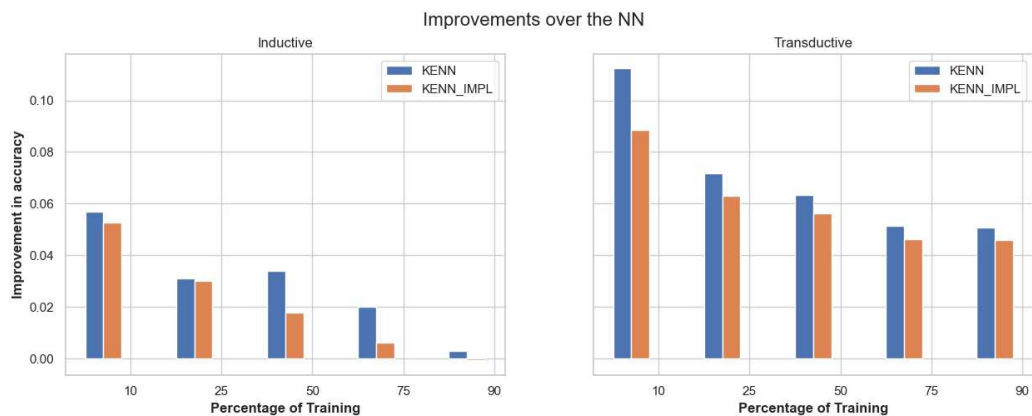


Figure 5.1: KENN improvements over underlying neural network

It is easy to see that both material implication and residuum KENN improve the performance of the simple NN. The improvements are much more significant when it comes to transductive learning, due to the fact that the number of edges is higher and, unlike KE layers, the NN doesn't exploit information given by them.

The model using residuum couldn't improve the performance of KENN using material implication or at least reproduce its results. The plot shows indeed a worsening in the accuracy of predicting documents topic. What the TBF-based KE does in this experiment is to modify doc-

ument preactivations only when they are linked in the graph. Note that, if two nodes x, y of the graph are not linked it means that $Cite(x, y)$ assumes the minimum possible value. Since the $Cite$ predicate is always negated, all logic rules are automatically satisfied when $Cite(x, y)$ is false. In such a case, the truth value of $\neg Cite(x, y)$ is always the highest. Hence, if $Cite(x, y)$ is false the KE changes only its value, keeping the document class predictions unchanged. On the other hand, when two nodes are linked, the binary predicate is the one having the lowest value, and the truth value of one between $\neg T(x)$ and $T(y)$ is increased, in particular the one corresponding to the greater preactivation returned by the NN. Then, what actually happens is that the KE increases the truth value of the predicate corresponding to $T(y)$ (the cited paper) only when it is higher than the truth value of $\neg T(x)$. On the contrary, it decreases the truth value of $T(x)$ only when the truth value of $\neg T(x)$ is higher than $T(y)$. In other words, a TBF-based KE improves the document's predicate corresponding to the preactivation that is more likely to be the true one, regardless of which is the cited and which is the citing.

When the KE uses the residuum boost function it also modifies the unary predicates only when their respective documents are linked in the graph, but the way it decides the improvements to apply is quite different compared to TBFs. As explained in the previous chapter, logic rules with implication inject a direction on the predicates to be modified, so, in this experiment, only $T(y)$ (the preactivation of document y being of topic T) is modified, and the cited document's preactivation value is increased to get it as high as the truth value of predicate $T(x)$, while preactivations of $T(x)$ are never adjusted. In brief, an RBF can adjust the prediction only for samples y , while a TBF can adjust the prediction for both samples x and y . Note that both antecedent and consequent predicates are generated by the NN. As a consequence, there is no reason to trust the antecedent more than the consequent, so applying a residuum boost function instead of a t-conorm boost function doesn't have an advantage in this scenario.

Since the first results suggested that material implication fits better than residuum interpretation in this task, another approach was tried to improve existing results by mixing the two types of logical rules in order to exploit both of their strengths.

The idea has been applied in transductive learning and consists of applying the material implication rule when linked nodes belonged to the same set (either training or evaluation sets) while using residuum implication for nodes linked across sets. More precisely, if two documents belonged to different sets, the property of RBF to decide which one would be modified gave the opportunity to change just preactivations of nodes belonging to the training set, with the goal of facilitating the training procedure. The prior knowledge for this adjusted exper-

iment consisted of both types of rules, but this time the binary predicate $Cite(x, y)$ was distinguished on the three available possibilities: $Cite_{Same}(x, y)$ if x and y belong to the same set, $Cite_{TrTs}(x, y)$ if the edge connecting the nodes goes from training to test sets and $Cite_{TsTr}(x, y)$ if it goes from test to train. The logic rules composing knowledge K would now appear as:

$$\neg T(x) \vee \neg Cite_{Same}(x, y) \vee T(y)$$

$$T(y) \wedge Cite_{TrTs}(x, y) \rightarrow T(x)$$

$$T(x) \wedge Cite_{TsTr}(x, y) \rightarrow T(y)$$

The experiment has been carried out with the same setting presented previously, but this time the results are shown using a plot that contains all single run's results, in order to have a better overview of the distribution of accuracy scores and deltas resulting from the comparison with NN performance. The mixed-knowledge KENN outcomes are compared again with the ones carried out by the model taken as baseline, i.e. material implication KENN.

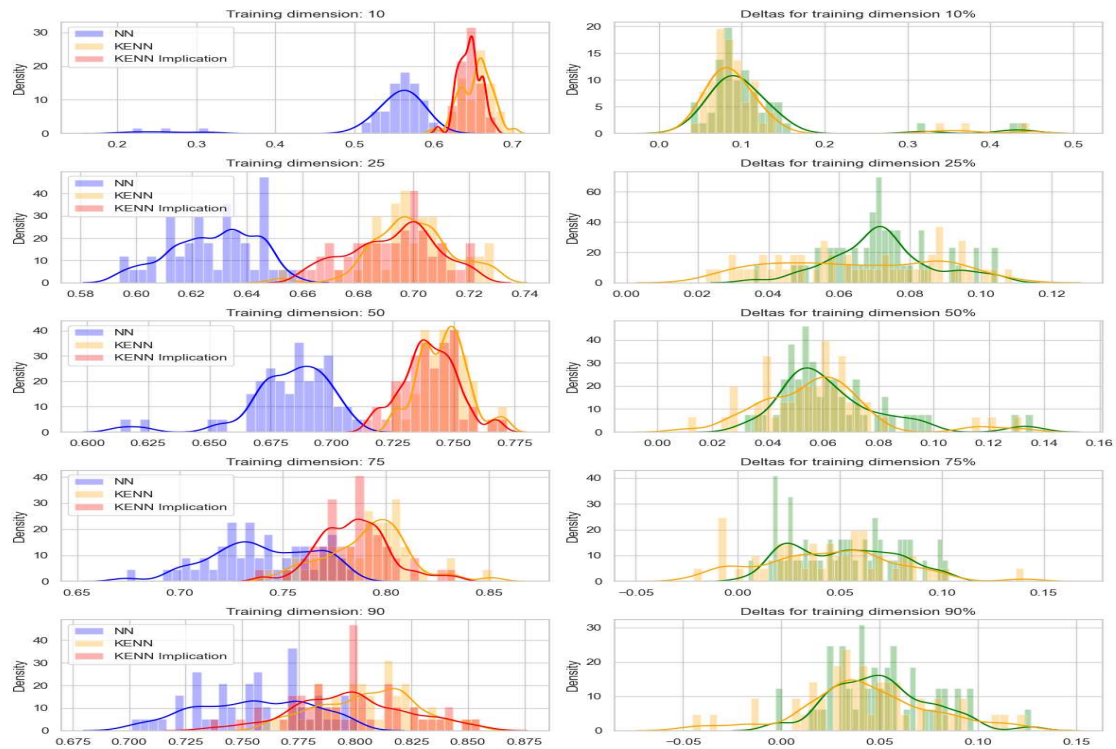


Figure 5.2: The histograms showing the result densities obtained on transductive learning for each training dimension. In the left columns are printed the accuracy scores for each of the three models while on the right are the improvements in accuracy carried out by KENN models w.r.t. the NN. In green are depicted the deltas computed using material implication, while in yellow are the ones computed from mixed knowledge and NN.

In the figure emerges that the distribution of improvements compared to NN is clearer when the training dimension is very small. At the same time, it is not that clear whether the use of the two different sets of prior knowledge makes a difference. Focusing on the delta's columns, the curves approximating values distribution are strongly overlapping suggesting that there isn't a significant difference in using a knowledge instead of the other, even though in each row the improvements given from mixed-knowledge KENN seem to have a larger variance.

Since from that plot isn't clear which model performed better on average, it is useful to take a look at the same plot used for the initial experiment:

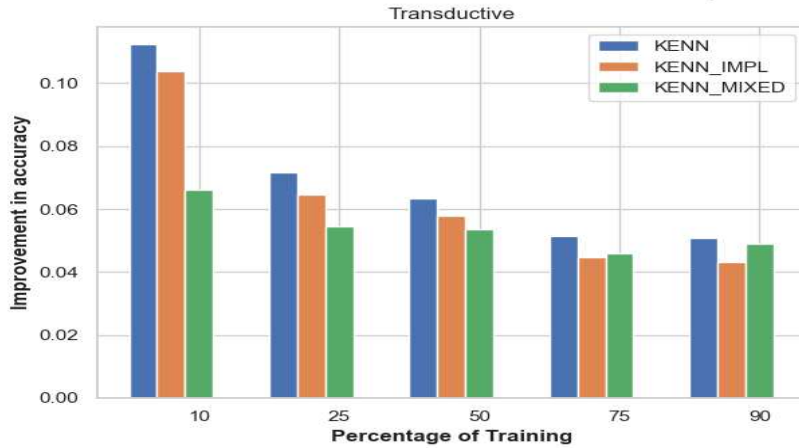


Figure 5.3: Improvements over the underlying neural network using the new set of logic rules

Here we can see how material implication KENN still performs better than knowledge-mixed one. Even though the new approach shows some improvements particularly when using 90% of the training set if compared to the results in figure 5.1 (that is what was expected to be seen since it made use of material implication for a subset of nodes) it still hasn't been enough to reach baseline KENN performance.

5.1.3 RESULTS DISCUSSION

In this first experiment, the results obtained weren't the hoped ones, given that the modifications implemented in the KENN architecture during this work didn't turn out to be useful for improving the results. However, it is worth noticing that the new architecture of KENN doesn't actually perform worse since it can still reproduce the results of classic KENN by injecting just material implication rules. Hence, it would be more correct to affirm that residuum logic rules, and not residuum KENN, make Citeseer predictions worse. Moreover, the Citeseer experiment was useful for giving a hint on which kinds of scenarios are more adapted for material implication rules instead of residuum ones and inspired thinking on practical differences in the behaviors of these two approaches. The outcomes suggested that in situations where the classes to predict appear as multiple predicates in the same logic rule, then the idea of increasing the truth value of the most "probable" one without distinguishing between antecedent and

consequent predicates is a better approach, due to the fact that there isn't a more reliable prediction that can play the role of a trusted assertion. Therefore, when the classifier's preactivations are all of the same level of quality, it is more advantageous to improve them through TBFs.

5.2 VISUAL RELATIONSHIP DETECTION

Visual Relationship Detection is a computer vision task regarding the understanding of spatial relations or interactions between objects in images or video. Visual relationships are not a new concept. The task of predicting relationships from detected objects was formalized in Lu *et al.* (2017) [12], which introduces a dataset that presents a variety of relationships per object type. Before that, the use of visual phrases describing relations between entities was already used for improving other computer vision tasks such as object recognition [13], or image retrieval [14]. From then on, visual relationship detection (VRD) has seen a rapid evolution. To tackle the problem many neuro-symbolic approaches has been defined. Donadello and Serafini [15] used Logic Tensor Network for the detection of unseen visual relationships, by assuming that logical knowledge allows to explicitly state relations between subjects/objects and relation predicates. KENN has been evaluated on a VRD dataset by Daniele and Serafini [1].

Continuing on this trend, in this work residuum KENN is evaluated on 2.5D visual relationship detection dataset (Su *et al.* [16]). This dataset extends the classic formalization of visual relationship detection by adding predicates regarding depth (that is why 2.5D). The reason for the choice of this dataset is that logic knowledge working along visual perception could be helpful in understanding depth relations (for example, a further object appears smaller than a closer one).

5.2.1 DATASET

The 2.5D-VRD dataset is made from images of Open Images V4 Dataset (OID)(Kuznetsova *et al.*, 2020) [17]. The dataset contains 110.894 images for 2.5VRD annotation, each image contains bounding boxes annotating the objects of 600 classes from OID. A bounding box is referred to by 4 coordinates, indicating the corners inside which the detected object is contained. For each image, the dataset contains information regarding bounding box coordinates alongside the class of the object contained in them. Separately are indicated the labels of predicates for each pair of objects. The spatial predicates introduced in this VRD dataset are depth and occlusion. For each pair of objects a predictor is expected to guess which one is closer to the camera's point of view and, if they are overlapped, which are the occluded and the occluding. The dataset proposes two sub-tasks, *within-image* and *across-image* prediction. In within-image, the relationship's predicate prediction is done on objects that are in the same image. In the across-image task, the objects compared come from two different images. Here

the occlusion predicate loses meaning, so the only predicate to be predicted is the distance relationship in relation to the respective points in which the photos were taken.

For each predicate a model should tell which is the closer and further object, or if they are at the same distance. The dataset contains also "unsure" labels, but those kinds of samples were removed to simplify the classification task. The same thing applies to occlusion predicate: if the model detects an occlusion it has to specify which object occludes the other or if they mutually occludes each other, otherwise it should predict "no occlusion". Here is an example of an image taken from both within-image and across-image tasks:

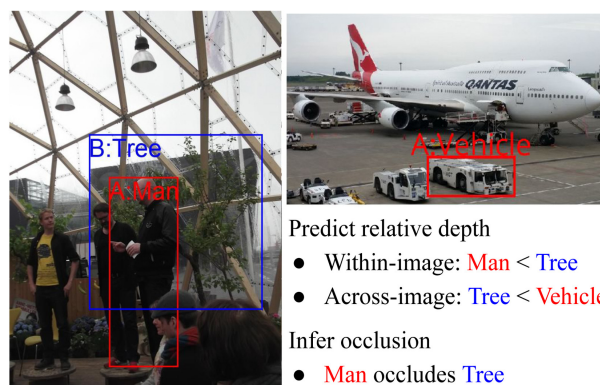


Figure 5.4: Within-image example on the left, across-image on the right. With relative annotations.

For this task, the article collecting the dataset presents a two-stage approach: object detection and predicate prediction. The dataset contains row images and labels either for objects bounding boxes and relationships. One could either use an end-to-end approach that starts from detecting the object and later predicts the relations from the detected objects. A simpler task assumes labeled bounding boxes to be given as input to the model, which would then perform just predicate prediction.

In this experiment, since the matter of interest was using prior knowledge to understand spatial relationships, I followed the second approach, using bounding box truth coordinates as input as well as their ground truth labels. Also, it was taken a subset of the images composing the 2.5D-VRD training set, while validation and test sets were kept the same. For the within-image task, the train set contains 10.559 visual relationships, which means 21.123 object bounding boxes, while in the across-image it is smaller and contains 520 pairs of images and 10.562 objects.

5.2.2 EXPERIMENT

During the learning and evaluation procedures, the only information made available to the predictors were objects bounding box coordinates. Since the across-image task didn't achieve worthy results, the results showed are the ones that occurred in the within-image task, which turned out to be more adapted for prior knowledge definition. The idea was to exploit information about bounding boxes to correct occlusion predictions, affirming that object bounding boxes that overlap (i.e. the intersection area is higher than a threshold parameter) imply that there is occlusion between the two objects, even though it doesn't explain which object occludes the other, so there are used together intersection information computed from the input and distances prediction returned by underlying NN. The logic rules designed for this experiment are:

- $Intersection(x) \wedge Object1Closer(x) \rightarrow Object1Occludes(x)$
- $Intersection(x) \wedge Object2Closer(x) \rightarrow Object2Occludes(x)$
- $Intersection(x) \wedge SameDistance(x) \rightarrow MutualOcclusion(x)$
- $\neg Intersection(x) \rightarrow NoOcclusion(x)$

where x represents a pair of objects. $Intersection(x)$ is the predicate manually computed from the bounding box coordinates. All the others instead, assume truth values according to NN predictions. For material implication KENN, the adjusted rules are:

- $\neg Intersection(x) \vee \neg Object1Closer(x) \vee Object1Occludes(x)$
- $\neg Intersection(x) \vee \neg Object2Closer(x) \vee Object2Occludes(x)$
- $\neg Intersection(x) \vee \neg SameDistance(x) \vee MutualOcclusion(x)$
- $Intersection(x) \vee NoOcclusion(x)$

In the same way, as it was done for Citeseer experiments, the models were evaluated on different amounts of the training set. Once again, symbolic information turns out to be an effective ingredient when the amount of data available for learning is reduced. By evaluating the whole training set, we allow the NN to learn the logic behind data, making it redundant to inject prior knowledge.

The baseline NN, and also the one underlying KENN prediction, used for this experiment is

a basic MLP made of two linear layers and dropout. The KENN model is evaluated both with residuum semantics and material implication rules.

The experiments were performed using 1%, 5%, 10%, and 20% percent of the training set. This time, the metric used to assert the quality of the results has been F1-score, due to the un-balanced nature of the dataset. All the experiments have been running for 100 runs. To assert the significance of the experiment a t-test was performed by comparing the populations consisting of the results returned by the MLP and KENN models. The metric used to test the significance of obtained outcomes was the p-value computed from the t-test which refers to the probability that the results occurred by chance. Since they are a probability, p-values are in interval $[0, 1]$, so low p-values indicate the data did not occur by chance. For our experiments, we consider them reliable if the related p-value took a value below 0.05. The p-values computed are all less than 10^{-5} , so the results are considered significant.

The results obtained are summarized in the following four tables, each for one percentage of the training set.

Models	F1-score distance	F1-score occlusion
MLP	59.1%	71.3%
residuum	59.8%	77.0%
material implication	59.9	77.1%

Table 5.1: 1% of training set, 100 runs

Models	F1-score distance	F1-score occlusion
MLP	62.8%	72.7%
residuum	61.3%	77.4%
material implication	62.7%	78.1%

Table 5.2: 5% of training set, 100 runs

Models	F1-score distance	F1-score occlusion
MLP	66.8%	74.4%
residuum	66.3%	79.0%
material implication	66.5%	78.1%

Table 5.3: 10% of training set, 100 runs

Models	F1-score distance	F1-score occlusion
MLP	68.2%	75.5%
residuum	68.1%	79.5%
material implication	66.9%	78.0%

Table 5.4: 20% of training set, 100 runs

In the table are reported f_1 -scores of distance and occlusion prediction. Even though logic rules defined for this problem only attempted to improve occlusion prediction, also distance f_1 -scores are printed. That is because, unlike residuum KENN, in the material implication model the Clause Enhancers could possibly decide to modify preactivations of the antecedent side, which involves distance predictions. Hence, there was a reason to check whether the material implication model could perform better on distance predicates.

Getting into the results achieved, it is possible to see how both KENN models could give a boost to the MLP performance for the occlusion predicate. In the first evaluation involving just 1% of the training set (which is about one hundred samples), neuro-symbolic models reached an f_1 -score of 77% that significantly improves the one carried out by the MLP which is just a 71.3%. By increasing the quantity of training sets the pattern remains the same. The MLP improves its results together with KENN models, but its performance grows slightly faster if compared to KENN. As expected to be seen, the neural-only MLP is more data-demanding, it reaches 75.5% f_1 -score with 20% of the training set, for a total improvement of 4.2%. While residuum KENN already starts by 77% and increases its score to 79.5% with a smaller improvement (but a better final result) of 2.5%. This shows again how adding prior knowledge removes the need to see data. This is proven by the fact that neuro-symbolic models using 1% of the training set performed better on occlusion prediction than an MLP trained on the 20%.

About the comparison between the KENN models, what emerges from the tables is that material implication doesn't seem to improve the results for the distance predicate. Remember that KENN modifies predictions returned by the underlying NN. So, if the NN classification on the distance predicate is too unreliable, the errors are spread also on the final enhanced pre-

dictions. For occlusion prediction, the two models stabilize on similar performances, with residuum KENN showing better f_1 -scores on evaluations with 10% and 20% of the training set, with a difference of 0.9% and 1.5% respectively. It possibly means that using a boost function only working on the consequent side of the implication arrow, could have helped in occlusion predicate predictions.

5.2.3 RESULTS DISCUSSION

Visual relationship detection experiment confirmed the hypothesis also suggested by the Cite-seer experiment, regarding the fact that in KENN residuum semantics aren't a revolutionary tool compared to material implication interpretation if exploited in a scenario where the goal is to correct neural network predictions. The results showed that residuum KENN can improve neural network predictions, especially in situations where the amount of data available for learning is not big enough. This is not an upgrade from the original KENN using the material implication rule, which also in this experiment could confirm the same ability. However, it is worth noting that in this situation, differently from the Citeseer experiment, the model performing slightly better here is residuum KENN. An explanation comes from the reasons presented in the introduction of chapter 4 for the implementation of residuum semantics: the way that the residuum boost function allows to "direct" changes in the preactivations. Having residuum KENN focusing on changing only occlusion predictions, causes it to have better performance on them, without losing "quality" on distance predicates which were already low. As we can see, when the task requires working on NN predictions, and not using them for further reasoning, it is impossible to definitively tell which kind of implication interpretation is the best one.

Visual relationship detection was a task that opened many other possibilities for evaluating KENN. During the work, many ideas have been attempted to reach better results in distance predicate and in the across-image task. Even though those attempts didn't lead to good results, either due to the nature of the dataset or for current limits of KENN architecture, they could possibly be an incentive for future works or architecture upgrades. Logic rules that could simplify prediction were added also for the distance predicate. The first idea was that, at parity of object class, a smaller bounding box could indicate a further object. Another one was that, for perspective reasons, a bounding box starting lower in the image, could possibly mean that the object is closer to the camera.

In the across-image task, given that it is impossible to define logic rules inspired by geometric

reasoning, it was made an attempt to insert in the model the logic of transitive property. In the across-image dataset, the same object is compared to the other ones multiple times, so the idea was to make predictions that respect the transitive relations in distance comparisons. Since transitive relation requires three variables, which isn't (yet) supported by KENN, a trick was to use as a domain triplets of objects, interpreting the distance relation as an unary predicate. For instance, the predicate $AB(x)$ would mean "the object A of triplet x is closer than the object B of triplet x ". The results weren't as good as expected. The weak predictions on the distance predicate caused a high propagation of errors in the KE layers. However, this attempt could possibly motivate further implementations in KENN architecture that could permit it to handle logic rules involving more than two variables.

5.3 MNIST ADDITION

The following experiments were designed to be more adaptable for a residuum approach in order to better highlight its properties. The next two sections show how the focus moves from modifying NN predictions to using those predictions as a means for carrying out further deductions that aren't directly faced by the NN. The third experiment regarded the evaluation of KENN on MNIST addition. The idea was conceived to test Residuum KENN's ability to emulate logical reasoning skills. This task was originally formulated in (Manhaeve et al. 2018)[18] and is an extension of the classic learning task on the MNIST dataset (Lecun et al. 1998)[19] to a more complex problem that requires reasoning. The MNIST dataset is composed of 70.000 examples of handwritten digits data each provided by 28×28 pixel grayscale image. In the reformulated task, instead of using labeled single digits, the training procedure is done on a pair of images labeled with the sum of individual labels. This demands further reasoning compared to the low-level perception required to recognize digits, which is already well handled by deep learning models. Even though digit sum can be directly learned by a standard neural classifier, addressing the higher-level reasoning with logic rules defining the addition operation leads to a better generalization. For example, if we also add logic rules for the sum of two-digit numbers, the model would continue to function without necessitating retraining. The same is not true for a neural classifier.

The implementation of residuum opened the way for KENN to approach the addition task, bringing it to the same level as other neuro-symbolic methods. Here, residuum semantics allows defining addition rules by using sum predicates as consequences of the perceived digits.

5.3.1 DATASET

The experiment's dataset was built by customizing the original MNIST dataset, where each sample would now consist of a pair of digits images labeled by their sums. In our dataset, there are 3000 samples for the training set and 5000 for testing. Each sample was built by randomly choosing two images from the MNIST dataset and computing their sum from the digits labels. Note that, in this experiment digits labels are assumed to be unknown, so a single MNIST addition example would look like a triplet (x, y, z) :

$$(\mathbf{5}, \mathbf{6}, 11)$$

where x and y are images of MNIST handwritten digits, and z is a label representing an inte-

ger in the range $\{0, \dots, 18\}$.

The results carried out by residuum KENN were compared with ones obtained by another neuro-symbolic method called *iterative local refinement* (ILR) introduced in (Daniele et al. 2023)[6], that integrates refinement functions for different fuzzy logic operators as a neural network layer to efficiently find refinements for logical formulas of any complexity.

5.3.2 EXPERIMENT

To perform the experiment it was firstly necessary to feed into the KE a prior knowledge defining addition. Consider a triplet $t = (x, y, z)$, where x, y are images of the MNIST dataset and z is the sum of their labels. We can define predicates representing digits and their sum. The list of predicates injected in the model is:

$$\{X_0(t), \dots, X_9(t), Y_0(t), \dots, Y_9(t), S_0(t), \dots, S_{18}(t)\}$$

where $X_i(t)$ stands for "digit depicted in image x of triplet t is i " and similarly $Y_i(t)$ refers to image y . The predicates $S_i(t)$ stand for "sum of digits depicted in images x and y of triplet t is i ". Given those predicates, each rule was then built by combining each possible pair of digits and specifying on the consequent side the respective sum. The whole set of logic rules would result in:

$$\forall i, j \in \{0, \dots, 9\} \\ X_i(t) \wedge Y_j(t) \rightarrow Sum_{i+j}(t)$$

This time the KE doesn't work, as previously done, on preactivation space, but instead manipulates directly class values returned by the softmax activation function. In this way digits predicates $X_i(t)$ and $Y_i(t)$ can only assume values in interval $[0, 1]$. The truth values concerning sum predicates are manually initialized to 0 (False). Thanks to the definition of the residuum boost function, they can increase until they reach the minimum value between predicates $X_i(t), Y_i(t)$. In this way, only when it happens that both image labels are correctly predicted then the predicate corresponding to their sum would be the higher value. Differently from Citeseer, here the predicted values are computed directly and only by the symbolical side of KENN. In this task, the NN's job only regards perception on images while the reasoning behind sum computation is carried out exclusively by the KE layers using a residuum boost function.

In this experiment, the role of perception was played by a convolutional neural network (CNN) made by three convolutional layers interpolated by two maximum pooling layers using ReLu activation function, followed by a sequential classifier consisting of two layers also using ReLu activation. The whole architecture KENN was completed by adding three KE layers on top of the CNN. Data samples were elaborated separately by the CNN which returned two vectors $\mathbf{x}, \mathbf{y} \in [0, 1]^9$, which were then fed into the KEs and concatenated to the zero-vector of sums. The model then returned the "enhanced" sums vector from which the loss is computed. The evaluation was run 10 times with batch size 128 and number of epochs 40, using Adam optimizer and learning rate 0.01. There were evaluated the quality of predictions regarding sums and single digits. As evaluation metrics were used accuracy score for sums and f1-score for digits prediction. The obtained results are the following:

Model	Sum accuracy	Digits F1-score
ILR	92,4% \pm 3.5	96.1% \pm 1.9
residuum KENN	91.1% \pm 1.7	95.7% \pm 0.8
material implication KENN	10.4% \pm 0.0	2.6% \pm 0.0

Table 5.5: The table compares ILR, residuum KENN, and material implication KENN average results obtained in the MNIST Addition experiment. 10 runs

In the table, we can see how KENN using material implication couldn't solve the task. On the other hand, residuum KENN could reach metrics scores roundly close to ILRs, meaning that residuum semantics could accomplish the task of MNIST addition. Since the results of ILR and residuum KENN are pretty close to each other, a significance test was carried out to make further conclusions on which of the two models performed better.

A t-test was used to compare if the differences between averages of results obtained in each evaluation run arose from random chance or were reliable. The p-values were computed by comparing the results obtained by ILR and residuum KENN in each run. The p-value for the accuracy scores is 0.313, while for f1 scores is 0.630. Meaning that it was concretely probable that the averages obtained were given by random chance. This doesn't mean that the results aren't significant, but simply that it cannot be said with certainty that ILR performance was better just because it presented higher means.

To have lower p-values and consequently more significant results, I decided to increment the number of runs for evaluating the performance, bringing it to 30 runs. This time, the obtained

averages were:

	Sum accuracy	Digits F1-score
ILR	85.0% ± 20.8%	83.8% ± 31.9%
residuum KENN	85.3% ± 20.3%	89.3% ± 23.1%
p-value	0.403	0.323

Table 5.6: Results of the same experiment showed in table 5.5, but this time computed on 30 runs.

This table presents very different results compared to before. Indeed, we can see a strong decrement in both accuracies and f1-scores, followed by an even higher increase in the standard deviation. This consequently brought the p-values to result higher than we expected, given the high standard deviations.

By analyzing every single run can be noted few outliers that highly drop qualities of digits classification (and so decreases also sum classification) to be around 1%. This behavior regarded both ILR and residuum KENN models, and to better understand its nature, it is here shown the related confusion matrix.

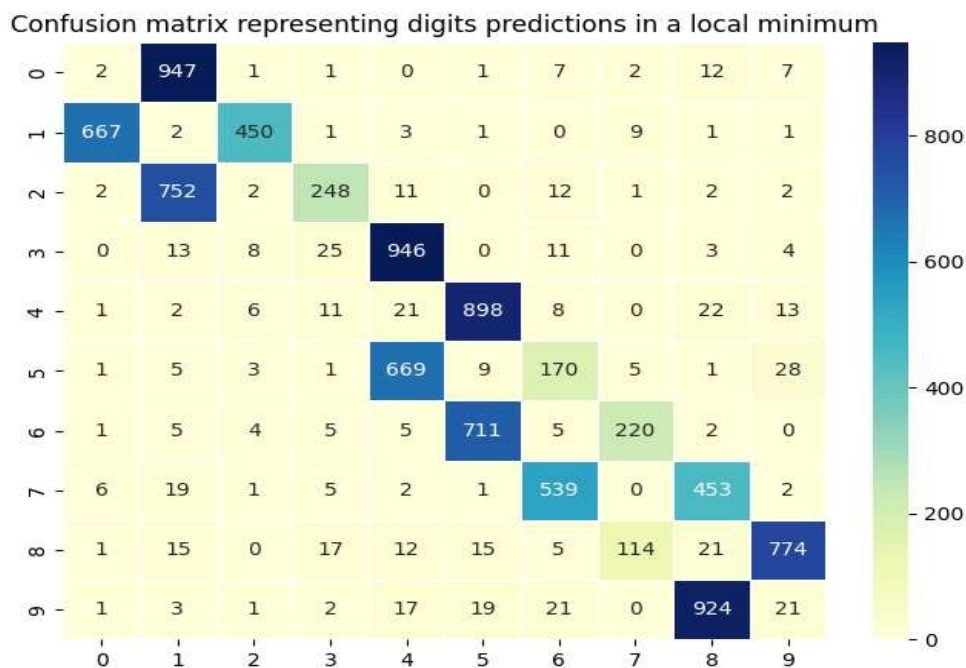


Figure 5.5: Confusion matrix on the MNIST classification for a local minimum

During the learning process, the underlying CNN gets stuck on a local minimum, where it recognizes each digit either as the correct digit minus one or plus one. Then, the model obtains

the correct prediction in close to 50% of the cases. For example, suppose the digits are a 3 and a 5. The 3 is classified as a 2 or a 4, while the 5 is classified as a 4 or a 6. If the model predicts 2 and 6 or 4 and 4, it returns the correct sum (8), otherwise, it does not. This problem was already pointed out in (Daniele et al. 2023)[6] for ILR. It was interesting to see that also residuum KENN presents the same issue with a similar frequency. In 30 runs ILR got stuck in a local minima three times while two times for residuum KENN. The sample size is still too small to conclude that KENN is less prone to these local minima, even though the standard digits f1-score in the table presents a higher value and a less standard deviation for residuum KENN, the p-value is still too high (0.32) to draw definite conclusions.

5.3.3 RESULTS DISCUSSION

During this experiment the performance of residuum KENN reached that of ILR, confirming the belief that adding residuum semantics in KENN could make it able to perform further reasoning starting from low-level perception. Even though it doesn't remove local minimum issues, it is further interesting to see if on a larger enough sample size which of the two models is less prone to getting stuck in them. This test wasn't carried out due to the amount of computational time required to have a sufficient amount of runs that could empirically prove the existence of a significant difference between KENN and ILR. This experiment could be a new goal for future work.

The presented task was tried to be approached also by the original KENN architecture, but its results weren't shown given its inability to learn the addition operation by defining it just using material implication. This experiment showed therefore in what kind of scenarios residuum semantic is a great graft for KENN. The satisfactory results obtained in this experiment by the application of the residuum boost function suggested further analysis and testing of those new capabilities of KENN.

5.4 VISUAL SUDOKU

Visual Sudoku Puzzle Classification (ViSudo-PC)[8] is a neuro-symbolic task, which consists of determining whether a completed Sudoku puzzle is correctly solved. Sudoku is a puzzle game in which there is a 9×9 grid, called a “puzzle” or “board”, in which each cell is populated with numbers 1–9. A puzzle is correct if no row, column, or non-overlapping 3×3 sub-grid (or “block”) contains the same number.

The puzzle is built from four canonical visual datasets, that are MNIST, KMNIST[20], FMINST[21], and EMNIST[22]. For a 9×9 puzzle, the first nine characters are taken from these datasets and used to fill the puzzle. As in MNIST addition, the model that approaches the problem firstly has to correctly recognize character and then exploit logic reasoning to determine the correctness of the puzzle, which is the only property given as supervision during the learning phase (digits true labels are hidden to the model).

This experiment was used to test the abilities of residuum KENN on approaching a more complex task compared to addition.

5.4.1 DATASET

ViSudo-PC dataset contains completed Sudoku puzzles along with their classification: “correct” if solved, “incorrect” otherwise. Each cell of the puzzle instead of containing symbolic information (e.g. digits labels) is provided in the form of an image depicting digits or characters based on the reference dataset from which puzzles were built. The data sources that were used to create the puzzles are:

MNIST Same dataset used for addition, composed by 70.000 samples of 28×28 pixel grayscale images depicting handwritten digits.

EMNIST Stands for extended MNIST. It introduces additional examples of handwritten digits as well as examples of handwritten English letters. The dataset contains a total of 814.255 characters. The digits are more frequent than letters and the number of images per letter is uneven since they roughly equate the frequency of use in the English language. The dataset is available in ByClass and ByMerge splits. Which have a different amount of classes because in the ByMerge version capital and lowercase letters are merged in the same label, while in ByClass the same letter has a lowercase and a capital label.

KMNIST Kuzushiji-MNIST contains 70.000 28×28 grayscale images, perfectly balanced like the original MNIST dataset, spanning 10 classes taken from Hiragana, a syllabary part of the Japanese writing system.

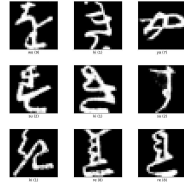


Figure 5.6: Example of KMNIST images

FMNIST Fashion-MNIST is a dataset containing articles taken from an online retailer’s shopping list. Images consist of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28×28 grayscale image, associated with a label from 10 classes.

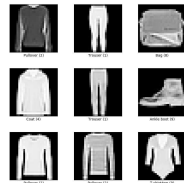


Figure 5.7: Example of FMNIST images

As previously said, those datasets are just the foundations of the ViSudo-PC dataset, which deserves a thorough description. No matter which kind of dataset is chosen as a character provider, ViSudo-PC presents different splits and subtasks that are independent of the type of images.

In the first place, the dataset offers also a tinier version of the original 9×9 version, made of 4×4 puzzles and thought to relieve the computational load that 9×9 requires.

To construct each puzzle, a set of cell labels respecting all Sudoku’s constraints is randomly generated, and then the images corresponding to those labels are assigned to each cell (cell labels are kept hidden). The pools of images for each cell label are never shared between train, validation, and test splits. To generate incorrectly solved puzzles existing correct puzzles are corrupted (a random generation of incorrect puzzles would have many mistakes). Puzzles are corrupted via

substitution or replacement, which means that puzzles can be corrupted either by swapping two random cells in the same puzzle or by randomly choosing a location in the puzzle and an alternate label. The number of corruptions made into a correctly solved puzzle is determined by a geometric distribution.

Puzzles can possibly share the same images with each other. In such a case, the connectivity between puzzles is determined by an *overlap* parameter ω . Considering a pool of images I , $\omega|I|$ images are randomly sampled from I and added to it, meaning that the new collection would be $I + \omega|I|$. Adding overlap allows the predictor to recognize the same entity being used in the same or different puzzles.

The dataset provides five different tasks to provide increasingly difficult problems. In each of them, the goal remains to classify Sudoku puzzles as correct or incorrect, but it varies the exact method of choosing data sources and cell labels.

Basic In the basic task a single data source is specified, and the first d characters of that data source are used to fill the puzzles. The images replacing cell labels are chosen randomly from the pool of images taken from the data source selected.

PerSplit This task generalizes the Basic method by randomly selecting, from the specified data source, the d cell labels to use inside a single split. In this way, each split uses a different set of labels, so the challenge is now that any architecture approaching this task must be effective on several types of images, and not specific to one label set.

PerPuzzle Here the idea is the same as the PerSplit method, but this time the cell labels (and the corresponding images) are randomly selected for each puzzle. The difficulty here is that, if the pool of cell labels is sufficiently large each cell label may be represented by even fewer examples than in PerSplit. Meaning that for this task, a predictor should rely mostly on symbolic information instead of learning an image classifier on the train split

PerCell In this task instead of limiting in each puzzle the use of d characters, the cell label is randomly chosen for each cell in each puzzle, meaning that it can use up to d^2 cell labels. Even though more than d characters are used, the constraints to respect remain the same, but this task takes the idea of having fewer examples of each cell label even further to complicate the role of an image classifier.

Transfer learning The last task is transfer learning, The same process used for Basic is used here, except that two disjoint sets of cell labels are chosen, one for training and another

one for test and validation, making symbolic reasoning a necessary component for approaching the problem.

The tasks refer to the method of building the puzzle, but they all share the same structure on which data are organized. Each split set row is a tensor in $\mathbb{R}^{d^2 \times 28 \times 28}$ where d^2 is the number of cells in the puzzle and 28×28 the dimension of the image depicting the character in the cell. So, a set containing n samples would be represented by a tensor in $\mathbb{R}^{n \times d^2 \times 28 \times 28}$.

5.4.2 EXPERIMENT

Residuum KENN was evaluated on Basic and PerSplit tasks on 4×4 puzzles. The data source considered for these experiments was only the MNIST dataset. Indeed, the main focus was on symbolic reasoning rather than low-level perception. The experiment was taken on these two tasks to investigate the order of improvements carried by the injection of symbolic reasoning and whether this gap would expand by increasing the difficulty of the problem.

Logical rules should codify that the same digit (character) cannot appear twice in the same row, column, and block. The most straightforward way to decide whether a Sudoku is correctly solved is to check for the existence of at least one occurrence that doesn't respect the constraints and classify it as incorrect. So, considering a 4×4 puzzle S , the predicates $P_{c_i}(x)$ with $c_i \in C = \{c_1, c_2, c_3, c_4\}$ states "cell x contains character c_i ". We have to check that for all pairs of cells in the puzzle that are on the same row, column, or block there isn't the same character C . Otherwise, the puzzle is incorrect. We can do that in the following way:

$$\forall x, y \text{ in } S$$

$$\bigwedge_{c \in C} P_c(x) \wedge P_c(y) \wedge Connected(x, y) \rightarrow Wrong(S)$$

with $Connected(x, y)$ meaning that x and y are in the same row, columns or block:

$$Connected(x, y) \leftrightarrow SameRow(x, y) \vee SameColumns(x, y) \vee SameBlock(x, y)$$

The formula contains the binary predicate $Connected(x, y)$ which can be handled by the relational version of KENN. However, it was decided to bypass the procedure of drawing a graph connecting cells that are in the same rows, columns, or squares. Moreover, since each sample of the dataset is a list of images for each cell of the Sudoku puzzle, the groundings for a variable x in the KE would represent a whole Sudoku puzzle and not a single cell in it. The

cells would instead be represented by the columns returned by an underlying CNN carrying the task of low-level perception. Each of them takes 4 possible values (the number of characters that could be in each cell), for a total of 16×4 predicates. Each column in the CNN prediction could then be interpreted with the predicate $P_{j,c_i}(x)$ stating "in puzzle x , the cell j contains character c_i ". Given the structure of the dataset and the consequent CNN output, a possible solution that could be handled by the KE layers was to compute all possible unique pairs of cells without repetition that are on the same row, column, or square. After that, rewrite the formula for validity check without using binary predicates, but by considering from the beginning just the pairs of cells for which $Connected(x, y)$ is true.

Calling A the set containing all possible pairs for which $Connected(x, y)$ is true, the rules would be:

$$\forall \text{ Sudoku puzzle } x$$

$$\bigwedge_{c_i \in C} \bigwedge_{j \in A} \left(P_{j_1, c_i}(x) \wedge P_{j_2, c_i}(x) \rightarrow Wrong(x) \right)$$

Note that in this formula, a single constraint $P_{j_1, c_i}(x) \wedge P_{j_2, c_i}(x) \rightarrow Wrong(x)$ constitutes a logic rule to be elaborated by the KE. To verify the validity of a Sudoku puzzle is necessary to verify the truth value of all those rules. As done for MNIST addition, $Wrong(x)$ predicate are manually defined and initially set to 0, and truth values of predicates $P_{j, c_i}(x)$ are in interval $[0, 1]$ (due to softmax activation function in the CNN). If the antecedent of the implication has a high truth value, meaning that a constraint isn't respected, then the KE would increase the value of $Wrong(x)$ to be as high as the antecedent. However, in the practical application of residuum KENN, putting all logic rules for each pair of cells in the same layer turned out to be a problem. Since the KE proposes the changes for all the logic rules simultaneously, the total increment for the predicate $Wrong(x)$ resulted to be much higher than the one it was supposed to be. For instance, consider just the constraints regarding the first row containing two 1, which are:

- $P_{1,1}(x) \wedge P_{2,1}(x) \rightarrow Wrong(x)$
- $P_{1,1}(x) \wedge P_{3,1}(x) \rightarrow Wrong(x)$
- $P_{1,1}(x) \wedge P_{4,1}(x) \rightarrow Wrong(x)$
- $P_{2,1}(x) \wedge P_{3,1}(x) \rightarrow Wrong(x)$
- $P_{2,1}(x) \wedge P_{4,1}(x) \rightarrow Wrong(x)$

$$\bullet P_{3,1}(x) \wedge P_{4,1}(x) \rightarrow Wrong(x)$$

Let's say that the underlying CNN predicts 1 to be in cell 2, resulting in a high truth value of predicate $P_{2,1}(x)$, let's say 0.9. At the same time the CNN is not sure of 1 being in the other cells of the first row, so predicates $P_{1,1}(x)$, $P_{3,1}(x)$, $P_{4,1}(x)$ would all get low truth values, such as 0.1. Relying on the predictions of the underlying CNN the first row of the puzzle would be correct, so after the elaboration carried out by the KE, $Wrong(x)$ should have a low truth value. Instead, what happens is that the KE computes deltas for each of the six constraints and then increases the value of the consequent predicate using all of them. Referring to the example, since the truth value of the antecedent conjunction would be 0.1 in each rule, the KE would propose to apply an increment of 0.1×6 on the $Wrong(x)$ truth value, bringing it to 0.6. Which is much higher than desired. To solve this problem, each logic rule was assigned to a specific KE layer. In this way, the constraints and the relative increment could be handled one at a time. If we consider the example, after applying the first constraint, $Wrong(x)$ would be incremented to 0.1. Now, the following constraints wouldn't affect its value since the predicate had already reached a truth value equal to the antecedent conjunction one.

Considering all pairs for a 4×4 puzzle, the KENN architecture designed for the experiment would have 288 KE layers (4 character times 6 pairs times 4 (number of rows), all multiplied by 3 to consider also columns and squares). The CNN underlying KE layers are made of 2 convolutional layers interspersed with max-pooling layers and ReLu activation function, followed by two sequential linear layers with 4 output neurons in which is applied softmax activation function.

In the Basic task, the experiment was carried out using different sizes of the training set. In one case all the splits were merged, making the training set contain 2200 samples as much as the test set, while the same evaluation was done using just one split, for a total of 200 samples for training and 200 for testing. The results obtained by residuum KENN were compared to the ones obtained just by the CNN underlying it. The results obtained have been computed by averaging the accuracy scores of 10 runs and are shown in the following table. The p-values have been computed by performing a t-test on the two populations consisting of accuracy values obtained from the CNN and residuum KENN models. The same test was repeated for both training sets.

At the bottom of the table is showed also the results of material implication KENN, which reaffirms its unsuitability for this type of task.

	Accuracy train set 200	Accuracy train set 2200
CNN	52.3% ± 3.0%	72.3% ± 14.7%
residuum KENN	76.4% ± 12.7%	93.2% ± 9.0%
p-value	2×10^{-4}	5×10^{-3}
material implication KENN	50.0%	50.0%

Table 5.7: Residuum KENN and CNN results on the Basic task on 10 runs with two different train sets

The p-values are sufficiently low so the results cannot be considered the result of chance. The table shows a significant improvement in the performance of residuum KENN compared to CNN not exploiting symbolic reasoning. As expected, adding prior knowledge modeling Sudoku constraints brought a great improvement in the accuracy scores. Looking merely at the difference between the two averages to understand the scope of this improvement is misleading, due to the high standard deviation in all evaluations. That is because, as in the MNIST addition experiment, both models are prone to stack on local minima. The accuracy scores, if we do not consider "bad runs", change drastically, growing on average to 56% for CNN and 91% for residuum KENN if trained on a single split, and to 84%, 97% if trained on all splits. Since residuum KENN reaches an almost perfect score in the Basic task when trained on a sufficiently large amount of puzzles (if it doesn't split into local minima), the next challenge was to evaluate it on a more difficult task, consisting of PerSplit. Here, differently from the Basic, the set of characters that appear in the puzzles is no longer limited to $\{1, 2, 3, 4\}$, but the digits are randomly selected in each split and the model ignores which ones have been used for building the puzzles. Given that, each digit has fewer images available for which the model can learn to recognize, complicating the task on the CNN. The KENN model used to approach this task was the same as the one designed for Basic, but this time all the splits were merged to perform the evaluation. The results obtained are the following:

	Average Accuracy
CNN	73.5% ± 10.5%
residuum KENN	77.1% ± 8.0%
p-value	0.11
material implication KENN	50.0%

Table 5.8: Residuum KENN and CNN results on PerSplit task, 20 runs

This table shows with confidence (p-value equals 0.11), that even by complicating the task

residuum KENN keeps working better than baseline CNN, even though the standard deviation is considerably high for the occurrence of local minima (if we ignore them, CNN has an average of 79% while KENN of 84%). An undesirable behavior is that, although the performance of CNN remains on average as bad as in the Basic task, the residuum KENN shows a significant decrement in its performance, about 16% worse than in Basic. The reason for this occurrence is explained in the next section.

5.4.3 RESULTS DISCUSSION

The experiment of ViSudo puzzle classification was initially thought to confirm the hints given by the experiment on MNIST addition, suggesting that residuum could add capabilities of symbolic reasoning into KENN architecture.

The logic behind checking the validity of Sudoku puzzles isn't very hard, but it still requires more complex reasoning if compared to simple addition. However, residuum KENN showed still better results compared to perceptive-only CNN. The improvement was more marked in the Basic task with respect to PerSplit, but in both tasks significant increments were carried out by residuum. From the two tables, we can see how the PerSplit results obtained by KENN are worse if compared to the ones in the Basic task, even though the CNN doesn't significantly decrease its performance from one to the other. An explanation for this behavior is that in the Basic task, the perception phase is more reliable, so the quality of prediction only depended on the ability of the model to understand the constraints for the validity of Sudoku puzzles. Since KENN had prior knowledge modeling the procedure for checking Sudoku's constraints, its accuracy was almost perfect. The CNN is instead more likely to get stuck on this high-level reasoning, explaining the huge gap in the models' performances in the Basic task.

On the other hand, in the PerSplit task the situation is quite different. Here there are fewer images available for each character to allow the underlying CNN to accurately distinguish them. Those kinds of errors propagate further during the KE elaboration and may sometimes taint final predictions. This entails that CNN didn't suffer from the increasing difficulty of the task, because its performance already struggled in Basic, while residuum KENN got into a great worsening (although still better than CNN) because in this task it cannot completely rely upon CNN character predictions.

For this reason, it wasn't decided to further evaluate residuum KENN on the more challenging tasks presented in subsection 5.4.1, keeping them open for evaluations with a further improved architecture. Despite that, the obtained results in Basic and PerSplit tasks have been considered

sufficiently satisfying to prove the logical reasoning abilities of residuum KENN. Even more, if we consider the improvements that the developed architecture has been able to make from the starting KENN, where material implication semantics isn't suitable for solving ViSudo puzzle classification, the achievements of residuum turn out to be a success.

6

Conclusion

The empirical analysis was performed to test the advantages of interpreting conditional statements using fuzzy logic's residuum semantic instead of the material implication rule. The experiments carried out suggested that the newly implemented architecture of KENN, which includes a residuum boost function, can generalize the interpretation done by material implication. It also opens prospects of a KENN architecture able to perform reasoning beyond neural network predictions. In the experiments where the task was to correct neural network prediction, the use of residuum instead of material implication does not boost, in general, the quality of predictions with respect to classic KENN architecture. The characteristics of the t-conorm boost function are generally more suitable for refining tasks. If needed, a t-conorm boost function can modify also the truth value of the antecedent side of a conditional statement interpreted using material implication. That was a feature especially useful in the Citeseer experiment. However, the new implementations do not prevent KENN from continuing to interpret conditional statements as it used to do, meaning that all results obtained in previous works are reproducible in the renewed architecture.

The greatest benefits obtained by residuum interpretation were shown in the last two experiments of MNIST Addition and Visual Sudoku, both requiring reasoning abilities. The residuum boost function allows the manipulation of the neural network perceptions which are used to assign a truth value to predicates beyond the ones caught by the NN, simulating a reasoning process. This capability made it possible to achieve results comparable to the state-of-the-art in these kinds of experiments. Moreover, those tasks couldn't be approached with KENN us-

ing material implication, meaning that residuum semantics has been a great addition to the pre-existing KENN architecture for adding reasoning-like abilities.

6.1 FUTURE WORK

In future works, it may be useful to investigate residuum semantics inducted by t-norms other than the Gödel t-norm. Focusing on the current residuum architecture, it could be made suitable for handling ternary predicates, as emerged in the VRD experiment, where such a feature could be useful for modeling transitive relations in prior knowledge. Another improvement could be made by ensuring the occurrence of minimal increments even when the same predicates appear multiple times in the set of logic rules. In the Visual Sudoku experiment, we work around the problem by assigning a specific KE layer for each implication formula, even though it makes the whole system slower. An efficient solution would allow the evaluation of residuum also on the more complex tasks of Visual Sudoku, as well as in other challenging experiments.

References


- [1] A. Daniele and L. Serafini, “Knowledge enhanced neural networks,” in *PRICAI 2019: Trends in Artificial Intelligence*, A. C. Nayak and A. Sharma, Eds. Cham: Springer International Publishing, 2019, pp. 542–554.
- [2] Dkm | fbk. [Online]. Available: <https://dkm.fbk.eu/>
- [3] A. Daniele and L. Serafini, “Knowledge enhanced neural networks for relational domains,” in *AIXIA 2022 – Advances in Artificial Intelligence*, A. Dovier, A. Montanari, and A. Orlandini, Eds. Cham: Springer International Publishing, 2023, pp. 91–109.
- [4] L. Serafini and A. d. Garcez, “Logic tensor networks: Deep learning and logical reasoning from data and knowledge,” *arXiv preprint arXiv:1606.04422*, 2016.
- [5] M. Diligenti, M. Gori, and C. Saccà, “Semantic-based regularization for learning and inference,” *Artificial Intelligence*, vol. 244, pp. 143–165, 2017, combining Constraint Solving with Mining and Learning. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0004370215001344>
- [6] S. v. H. Daniele, van Krieken, “Refining neural network predictions using background knowledge,” 2023. [Online]. Available: <https://doi.org/10.1007/s10994-023-06310-3>
- [7] E. Augustine, P. Jandaghi, A. Albalak, C. Pryor, C. Dickens, W. Wang, and L. Getoor, “Emotion recognition in conversation using probabilistic soft logic,” 2022.
- [8] E. Augustine, C. Pryor, C. Dickens, J. Pujara, W. Wang, and L. Getoor, “Visual sudoku puzzle classification: A suite of collective neuro-symbolic tasks,” in *International Workshop on Neural-Symbolic Learning and Reasoning (NeSy)*, 2022.
- [9] P. Hájek, “Basic fuzzy logic and bl-algebras,” 1998. [Online]. Available: <https://doi.org/10.1007/s005000050043>

- [10] P. Hájek, *Product Logic, Gödel Logic (and Boolean Logic)*. Dordrecht: Springer Netherlands, 1998, pp. 89–107. [Online]. Available: https://doi.org/10.1007/978-94-011-5300-3_4
- [11] C. L. Giles, K. D. Bollacker, and S. Lawrence, “Citeseer: An automatic citation indexing system,” in *Proceedings of the third ACM conference on Digital libraries*, 1998, pp. 89–98.
- [12] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei, “Visual relationship detection with language priors,” in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 852–869.
- [13] M. A. Sadeghi and A. Farhadi, *Recognition using visual phrases*. IEEE, 2011.
- [14] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei, “Image retrieval using scene graphs,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
- [15] I. Donadello and L. Serafini, “Compensating supervision incompleteness with prior knowledge in semantic image interpretation,” in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.
- [16] Y.-C. Su, S. Changpinyo, X. Chen, S. Thoppay, C.-J. Hsieh, L. Shapira, R. Soricut, H. Adam, M. Brown, M.-H. Yang, and B. Gong, “2.5d visual relationship detection,” *Computer Vision and Image Understanding*, vol. 224, p. 103557, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1077314222001357>
- [17] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Mallocci, A. Kolesnikov *et al.*, “The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale,” *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.
- [18] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt, “Deep-problog: Neural probabilistic logic programming,” in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/dc5d637ed5e62c36ecb73b654b05baza-Paper.pdf

- [19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665>
- [20] V. U. Prabhu, "Kannada-mnist: A new handwritten digits dataset for the kannada language," *arXiv preprint arXiv:1908.01242*, 2019.
- [21] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
- [22] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.



Source code

The following appendix shows part of the code written to implement residuum semantics in the sub-components of KENN. The implementations have been carried out using Pytorch* .

The Forward method in Relational KENN class:

```
1 def forward(self, unary, binary, index1, index2)
2     -> (torch.Tensor, torch.Tensor):
3
4 if len(self.unary_clauses) != 0:
5     deltas_sum, deltas_u_list = self.unary_ke(unary)
6     u = unary + deltas_sum
7 else:
8     u = unary
9
10 if len(self.binary_clauses) != 0 and len(binary) != 0:
11     joined_matrix = self.join(u, binary, index1, index2)
12     deltas_sum, deltas_b_list = self.binary_ke(joined_matrix)
13
14     delta_up, delta_bp = self.group_by(u, deltas_sum, index1, index2)
15 else:
16     delta_up = torch.zeros(u.shape)
17     delta_bp = torch.zeros(binary.shape)
```

*PyTorch, the PyTorch logo and any related marks are trademarks of The Linux Foundation.

```

18
19 if len(self.implication_unary_clauses) != 0:
20     deltas_sum, deltas_u_list = self.implication_unary_ke(unary)
21     u_impl = u + deltas_sum
22 else:
23     u_impl = u
24
25 if len(self.implication_binary_clauses) != 0:
26     joined_matrix = self.join(u, binary, index1, index2)
27     deltas_sum, deltas_b_list = self.implication_binary_ke(joined_matrix)
28     delta_impl_up, delta_impl_bp = self.group_by(
29         u, deltas_sum, index1, index2)
30 else:
31     delta_impl_up = torch.zeros(u.shape)
32     delta_impl_bp = torch.zeros(binary.shape)
33
34 return self.activation(u_impl + delta_up + delta_impl_up),
35 self.activation(binary + delta_bp + delta_impl_bp)

```

Listing A.1: When *relationalKENN* object is called, it needs as parameters unary and binary predicates truth values. For the latter, two lists of indexes indicating which samples are connected in the binary predicates are needed. Based on the clauses it received as input, *relationalKENN* takes care of calling the dedicated KE. If there are binary clauses, it is at this level that the join operation between unary and binary matrixes is performed before calling the *KnowledgeEnhancer* instance. The returned deltas are then summed into the unary matrix by grouping by the sample of interest.

The Forward method in Knowledge Enhancer class:

```

1 def forward(self, ground_atoms, using_max=False) -> (torch.Tensor, [torch.Tensor,
2     torch.Tensor]):
3     scatter_deltas_list: [torch.Tensor] = []
4     light_deltas_list = []
5     weights = []
6     for enhancer in self.clause_enhancers:
7         scattered_delta, delta = enhancer(ground_atoms)
8         scatter_deltas_list.append(scattered_delta)
9         if self.save_training_data:
10            light_deltas_list.append(delta)
11            weights.append(enhancer.clause_weight.numpy()[0][0])
12
13 deltas_data = [light_deltas_list, weights]
14 if using_max:

```

```

14 stacked_deltas = torch.stack(scatter_deltas_list)
15 _, indexes = torch.abs(stacked_deltas).max(dim=0)
16 return torch.gather(stacked_deltas, 0, indexes.unsqueeze(0)), deltas_data
17 else:
18 return torch.stack(scatter_deltas_list).sum(dim=0), deltas_data

```

Listing A.2: When a *KnowledgeEnhancer* object is invoked, it is responsible for collecting the deltas returned by all the *ClauseEnhancer* instances inside it.

The Forward method in Clause Enhancer class:

```

1
2 def forward(self, ground_atoms) -> (torch.Tensor, torch.Tensor):
3     # [b, 1]
4     selected_predicates = self.select_predicates(ground_atoms)
5
6     delta = self.conorm_boost(selected_predicates, self.signs)
7     # [b, 2|U|+|B|]
8     scattered_delta = torch.zeros_like(ground_atoms)
9     scattered_delta[..., self.gather_literal_indices] = delta
10
11 return scattered_delta, delta

```

Listing A.3: When a *ClauseEnhancer* object is called, it gathers the truth values of the predicates (that are in the parameter *ground_atoms*) and calls the t-conorm boost function for computing the deltas.

The Forward method in the Residuum Clause Enhancer class:

```

1
2 def forward(self, ground_atoms) -> (torch.Tensor, torch.Tensor):
3
4     antecedent_predicates, consequent_predicates = self.select_predicates(
5         ground_atoms)
6     delta = self.conorm_boost([antecedent_predicates, consequent_predicates], self.
7         signs)
8
9     scattered_delta = torch.zeros_like(ground_atoms)
10    scattered_delta[..., self.consequent_literal_indices] = delta

```

```
10     return scattered_delta, delta
```

Listing A.4: When a *ResiduumClauseEnhancer* object is called, it gathers the truth values of the predicates (that are in the parameter *ground_atoms*) and calls the residuum boost function for computing the deltas. Differently from standard CE, it distinguishes between antecedent and consequent predicates.

T-conorm boost function class:

```
1
2 class GodelBoostConormApprox(BoostFunction):
3
4     def forward(self, selected_predicates: torch.Tensor, signs: torch.Tensor):
5         self.clause_weight.data = torch.clip(self.clause_weight, self.min_weight,
6         self.max_weight)
7
8         clause_matrix = selected_predicates * signs
9
10        return signs * softmax(clause_matrix, dim=-1) * self.clause_weight
```

Listing A.5: Implementation of the approximation of a minimal t-conorm boost function

Residuum boost function class:

```
1
2 class GodelBoostResiduum(BoostFunction):
3
4     def forward(self, selected_predicates: [torch.Tensor, torch.Tensor], signs: [
5     torch.Tensor, torch.Tensor]):
6
7         self.clause_weight.data = torch.clip(self.clause_weight, self.min_weight,
8         self.max_weight)
9
10        antecedent_matrix = selected_predicates[0] * signs[0]
11        consequent_matrix = selected_predicates[1] * signs[1]
12
13        conjunction_val = torch.min(antecedent_matrix, 1)[0]
14        disjunction_val = torch.max(consequent_matrix, 1)[0]
15        indices_consequent = torch.argmax(consequent_matrix, 1)
16
17        formula_unsatisfied = disjunction_val < conjunction_val
18        delta = torch.zeros(consequent_matrix.size()[0], consequent_matrix.size()
19        [1])
20        delta[np.arange(delta.size()[0]), indices_consequent] = \
```



```
17         torch.minimum(self.clause_weight, conjunction_val - disjunction_val) *  
formula_unsatisfied  
18         delta *= signs[1]  
19         return delta
```

Listing A.6: Implementation of a minimal residuum boost function

Acknowledgments

I would like to express my gratitude to Professor Serafini for providing me with the opportunity to work on this project and to Alessandro Daniele for his assistance and support throughout my experience.