# RIDDLE: Rule Induction with Deep Learning

Cosimo Persia* and Ricardo Guimarães

Department of Informatics, University of Bergen, Norway

## Abstract

Numerous applications rely on the efficiency of Deep Learning models to address complex classification tasks for critical decisions-making. However, we may not know how each feature in these models contributes towards the prediction. In contrast, Rule Induction algorithms provide an interpretable way to extract patterns from data, but traditional approaches suffer in terms of scalability. In this work, we bridge Deep Learning and Rule Induction and define the RIDDLE (**R**ule **Ind**uction with **D**eep **Le**arning) architecture. We show that RIDDLE has state-of-the-art performance in Rule Induction via an empirical evaluation.

## 1 Introduction

The adoption of Rule Induction algorithms supports decisions in medicine [30, 27], fault and fraud detection [33, 7], and brings benefits to chemical, oil [4], and energy industries [32], among others. These algorithms express patterns found in data in the form of associative ('if-then') *rules* effectively aiding users in decision-making [5, 6, 31, 22].

Rule Induction approaches have to solve hard combinatorial problems, as the symbols available for constructing the rules form a discrete space [5, 6, 20]. Hence, their scalability suffers when compared to classification methods that can rely on techniques tailored for the optimisation of differentiable functions, such as Deep Leaning with gradient-based optimisation [15, 22]. Indeed, Deep Learning approaches have excelled in many tasks, including image segmentation [24] and text generation [16]. The success of Deep Learning approaches have two well-known reasons: the flexibility to handle many different forms of data and scalability powered by technological advances. Yet, the lack of interpretability of Deep Learning models limits their application in systems that directly or indirectly influence critical decisions [6, 29].

Our main contribution is an original Deep Learning architecture dubbed RIDDLE (**R**ule **Ind**uction with **D**eep **Le**arning) which learns rules using a differentiable error function. This means that our approach can employ the efficient optimisation methods, while providing interpretable rules. We show with formal arguments the reasons that led the development of the RIDDLE architecture. Since it is based on the formal framework of Possibility Theory, RIDDLE builds rules equipped with certainty degrees which express the reliability of each rule. This has two main advantages over most traditional Rule Induction methods: (1) they avoid having 'sharp' decision boundaries and (2) the orders of the rules is irrelevant, that is, they yield rule sets instead of lists [20]. Furthermore, we show that our method has state-of-the-art performance (accuracy) on well-known datasets, especially those with uncertain or missing information.

**Related work.** The most prominent Rule Induction algorithms belong to the decision tree family such as RIPPER [5] that outputs binary (or crisp) rules. FURIA is an extension that generates fuzzy rules [20]. To solve the issue of scalability of Rule Induction systems in Big Data settings, Elkano et al. [15] proposed CFM-BD, a distributed system for fuzzy Rule Induction using a MapReduce paradigm. While CFM-BD has shown promising results, it still solves a search problem on a large discrete space.

We can also find works on neuro-symbolic approaches for Rule Induction. DR-Net [28] employs a simple 2-layer neural network architecture to learn rule sets. DR-Net also controls the complexity of the rules learned via a sparsity term. In contrast, RIDDLE automatically prioritises simpler

*Corresponding Author: cosimo.persia@uib.no



https://doi.org/10.7557/18.6801

rules over complex ones as a consequence of weight-decay. Kusters et al. [22] define a 3-layer neural network architecture for rule induction, named R2N, that can also identify potential new terms. R2N integrates neural networks and Rule Induction with a differentiable function, but it can only learn positive DNF, a restricted class of rules where variables cannot be negated [1]. RIDDLE instead can learn any propositional formula, while also providing a measure of the 'reliability' for each rule.

Glanois et al. [18] propose HRI, a hierarchical approach to Rule Induction designed for Inductive Logic Programming (ILP) [25]. The language of rules differs considerably from propositional rules. Their method also relies on pre-defined rule templates that determine the types of rules that can be learned, while we do not impose such restrictions with our method.

In Section 2, we present the theoretical foundations of our contribution. Then, we introduce RIDDLE in Section 3. In Section 4, we provide an empirical comparison between RIDDLE and FURIA, a prominent fuzzy Rule Induction algorithm. We conclude and mention future steps in Section 5.

## 2 Preliminaries

In this section, we present relevant notions regarding propositional logic [2], Possibility Theory [11, 12], and neural networks [19].

**Propositional Logic**. Let $\mathsf{V}$ be a finite set of *boolean variables*. *A literal* over $\mathsf{V}$, denoted with the symbol $l$, is either a *variable* $\mathsf{v} \in \mathsf{V}$ or its negation, in symbols, $\neg\mathsf{v}$. The former is also called a *positive literal*, the latter a *negative literal*. A *clause* is a disjunction ($\vee$) of literals. A *formula*[1] is a conjunction ($\wedge$) or set of clauses. A clause $\mathsf{c}$ can also be expressed as a *rule* $\mathsf{r}$ of the form $\mathsf{ant}(\mathsf{r}) \rightarrow \mathsf{con}(\mathsf{r})$ where $\mathsf{ant}(\mathsf{r})$ (the *antecedent* of $\mathsf{r}$) is a conjunction of all but one negated literals in $\mathsf{c}$ and $\mathsf{con}(\mathsf{r})$ (the *consequent* of $\mathsf{r}$) is a single literal. Example 1 illustrates these concepts.

**Example 1.** *Let* $\mathsf{V} = \{\mathsf{v}_1, \mathsf{v}_2, \mathsf{t}\}$. *For instance,* $(\neg\mathsf{v}_1 \vee \mathsf{v}_2 \vee \mathsf{t})$ *is a clause,* $(\mathsf{v}_1 \wedge \mathsf{v}_2)$ *or* $\{(\mathsf{v}_1), (\mathsf{v}_2)\}$ *is a formula and* $((\mathsf{v}_1 \wedge \mathsf{v}_2) \rightarrow \mathsf{t})$ *is a rule.*

[1] In general, formulas in propositional logic can combine $\neg$, $\vee$, and $\wedge$ arbitrarily. However, we will assume that every formula is in conjunctive normal form (CNF).

A (partial) interpretation $\mathcal{I}$ over $\mathsf{V}$ is a function $\mathcal{I} : \mathsf{V} \rightarrow \{\top, \bot, ?\}$ that states which variables are regarded as 'true', 'false', and 'unknown'. $\mathcal{I}$ *falsifies* a variable $\mathsf{v} \in \mathsf{V}$ if $\mathcal{I}(\mathsf{v}) = \bot$, otherwise it *satisfies* it. $\mathcal{I}$ satisfies a negative literal $\neg\mathsf{v}$ iff $\mathcal{I}(\mathsf{v})$ is equal to '$?$' or '$\bot$'. $\mathcal{I}$ satisfies a clause $\mathsf{c}$ if it satisfies at least one literal in $\mathsf{c}$. Intuitively, $\mathcal{I}$ satisfies a clause if there is a way of replacing 'unknown' values ($?$) with 'true' ($\top$) or 'false' ($\bot$) such that the clause is satisfied. Also, $\mathcal{I}$ satisfies a formula $\phi$, in symbols $\mathcal{I} \models \phi$, if every clause in $\phi$ is satisfied by $\mathcal{I}$. We write $\mathcal{I} \not\models \phi$ instead, if $\mathcal{I}$ does not satisfies $\phi$. We clarify these notions with the following example.

**Example 2.** $\mathcal{I}_1 = \{(\mathsf{v}_1, \top), (\mathsf{v}_2, \top), (\mathsf{t}, \top)\}$, *and* $\mathcal{I}_2 = \{(\mathsf{v}_1, \top), (\mathsf{v}_2, ?), (\mathsf{t}, ?)\}$ *satisfy the clause in Example 1.* $\mathcal{I}_3 = \{(\mathsf{v}_1, \top), (\mathsf{v}_2, \bot), (\mathsf{t}, \bot)\}$ *does not.*

For conciseness, we abuse the notation and write $\bot$ and $\top$ when referring to the empty disjunction and the empty conjunction, respectively.

**Possibility Theory**. A *possibilistic clause* over $\mathsf{V}$ is a pair $(\phi, \alpha)$, where $\phi$ is a clause over $\mathsf{V}$ and $\alpha$ is a real number with finite precision in the interval $(0, 1]$, called the *valuation* of $\phi$. A *possibilistic formula* $\mathcal{K}$ is a conjunction of possibilistic clauses. Given $\mathcal{K}$, and a set $\Omega$ of interpretations over $\mathsf{V}$, we define a *possibility distribution* $\pi_\mathcal{K} : \Omega \rightarrow [0, 1]$ as

$$\pi_\mathcal{K}(\mathcal{I}) \coloneqq \min_{\mathcal{I} \in \Omega}(\{1\} \cup \{1 - \alpha \mid \mathcal{I} \not\models \phi, (\phi, \alpha) \in \mathcal{K}\}).$$

The *possibility* degree $\Pi_\mathcal{K}(\phi)$ of $\phi$, indicates how much $\phi$ is coherent with $\pi_\mathcal{K}$ and $N_\mathcal{K}(\phi)$ expresses the *necessity* degree of $\phi$ being implied by $\pi_\mathcal{K}$. They are defined as follows

$$\Pi_\mathcal{K}(\phi) \coloneqq \sup_{\mathcal{I} \in \Omega}\{\pi(\mathcal{I}) \mid \mathcal{I} \models \phi\}$$
$$N_\mathcal{K}(\phi) \coloneqq 1 - \Pi(\neg\phi).$$

A possibility distribution $\pi_\mathcal{K}$ *satisfies* a possibilistic clause $(\phi, \alpha)$, written $\pi_\mathcal{K} \models (\phi, \alpha)$, if $N_\mathcal{K}(\phi) \geq \alpha$, and it satisfies a possibilistic formula $\mathcal{K}$ if it satisfies each $(\phi, \alpha) \in \mathcal{K}$. We have that $(\phi, \alpha)$ is *entailed* by $\mathcal{K}$, written $\mathcal{K} \models (\phi, \alpha)$, if all possibility distributions that satisfy $\mathcal{K}$ also satisfy $(\phi, \alpha)$. We recall key properties of this theory [14].

**Lemma 1** ([9])**.** *For every possibilistic formulas* $\mathcal{K}$ *and* $(\phi, \alpha)$*, we have that* $N_\mathcal{K}(\phi) = \max\{\alpha \mid \mathcal{K} \cup \{(\neg\phi, 1)\} \models (\bot, \alpha)\}$.

**P1** $\Pi(\phi_1 \vee \phi_2) = \max(\Pi(\phi_1), \Pi(\phi_2))$; and

**P2** $N(\phi_1 \wedge \phi_2) = \min(N(\phi_1), N(\phi_2))$.

**Neural Networks**. We assume an arbitrary but fixed ordering of the variables in $\mathsf{V}$: $(\mathsf{v}_1, \ldots, \mathsf{v}_n, \mathsf{t}_1, \ldots, \mathsf{t}_m)$. We will predict the certainty degree of the variables $\mathsf{t}_j$ with the information provided by $\mathsf{v}_i$. In this work, a *neural network* model is a function $g : [0,1]^{2n} \to [0,1]^m$. It takes as input a vector denoting the possibilities of each literal and it outputs the possibilities of each $\mathsf{t}_j$. The function $g$ contains parameters to be optimised by iterative updates of the backpropagation algorithm. To take advantage of the optimisation steps, we employ the function log-sum-exp: $\mathsf{LSE}_\alpha(x_1, \ldots, x_n) \coloneqq \frac{1}{\alpha}\ln(e^{\alpha x_1} + \cdots + e^{\alpha x_n})$, as a smooth approximation of the max (min) function when $\alpha \to \infty$ ($\alpha \to -\infty$). We write $\mathsf{LSE}_{\max}$ for $\mathsf{LSE}_{30}$ and $\mathsf{LSE}_{\min}$ for $\mathsf{LSE}_{-30}$.

# 3 Introducing RIDDLE

In this section, we describe the theoretical motivations that led the development of RIDDLE. Our goal is to predict $\Pi(\neg\mathsf{t})$ (or $\Pi(\mathsf{t})$), of a target variable $\mathsf{t}$, from the possibility degrees of literals $\mathsf{V} \coloneqq \{\mathsf{v}_1, \neg\mathsf{v}_1, \ldots, \mathsf{v}_n, \neg\mathsf{v}_n\}$. If needed, we can compute how necessary the target variable is according to our input with $N(\mathsf{t}) = 1 - \Pi(\neg\mathsf{t})$ (or $N(\neg\mathsf{t})$).

We assume the general setting in which we have a dataset of (partial) interpretations $\mathrm{I} \coloneqq \{\mathcal{I}_1, \ldots, \mathcal{I}_d\}$ where some rules of the form $\phi \to \mathsf{t}$ generally hold. We can convert the statements in $\mathcal{I} \in \mathrm{I}$ to possibilistic degrees via the method proposed by Joslyn [21] to estimate possibilities from imprecise data. We first define the set $\mathrm{I}_\mathcal{I} \coloneqq \{\mathcal{I}' \in \mathrm{I} \mid \forall \mathsf{v} \in \mathsf{V}, \mathcal{I}(\mathsf{v}) = ? \text{ or } \mathcal{I}(\mathsf{v}) = \mathcal{I}'(\mathsf{v})\}$ that contains precisely the interpretations in I that differ only on the unknown values of $\mathcal{I}$. Then, from $\mathrm{I}_\mathcal{I}$, we count the number of interpretations that satisfy a literal $l$ which is given by $\mathsf{c}_{\mathrm{I}_\mathcal{I}}(l) \coloneqq |\{\mathcal{I} \in \mathrm{I}_\mathcal{I} \mid \mathcal{I} \models l\}|$. Finally, the possibility associated to a literal $l$, according to the facts in $\mathcal{I}$ and I, is defined as $\Pi^{\mathrm{I}_\mathcal{I}}(l) \coloneqq \mathsf{c}_{\mathrm{I}_\mathcal{I}}(l)/\max(\mathsf{c}_{\mathrm{I}_\mathcal{I}}(l), \mathsf{c}_{\mathrm{I}_\mathcal{I}}(\neg l))$. Therefore, from I we can get the set of possibility degrees for each input literal $\mathsf{D} \coloneqq \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^d\}$, where for any $\mathcal{I}_i \in \mathrm{I}$, $\boldsymbol{x}^i \coloneqq (\Pi^{\mathrm{I}_{\mathcal{I}_i}}(\mathsf{v}), \Pi^{\mathrm{I}_{\mathcal{I}_i}}(\neg\mathsf{v}_1), \ldots, \Pi^{\mathrm{I}_{\mathcal{I}_i}}(\mathsf{v}_n), \Pi^{\mathrm{I}_{\mathcal{I}_i}}(\neg\mathsf{v}_n))$. The number $j$ in $\boldsymbol{x}^i_j$ corresponds to the value at the $j$-th position. We denote the formula associated to $\boldsymbol{x}^i$ w.r.t. $\mathcal{I}_i \in \mathrm{I}$ by $\mathcal{X}^i \coloneqq \{(l, 1 - \Pi^{\mathrm{I}_{\mathcal{I}_i}}(\neg l)) \mid \Pi^{\mathrm{I}_{\mathcal{I}_i}}(\neg l) < 1\}$. Example 3 provides a clarification.

**Example 3.** *Let* $\beta = \{\mathcal{I}_1, \mathcal{I}_2\}$ *as in Example 2.* $\boldsymbol{x}^1 = (1,0,1,0,1,0)$*, and* $\boldsymbol{x}^2 = (1,0,1,1/2,1,1/2)$*.* $\boldsymbol{x}^2_4$ *is* $1/2$ *and* $\mathcal{X}^2 \coloneqq \{(\mathsf{v}_1, 1), (\mathsf{v}_2, 1/2), (\mathsf{t}, 1/2)\}$.

**Theoretical Motivation**. We assume that the unknown formula $\mathcal{K} = \{(\phi_i \to \mathsf{t}, \alpha_i) \mid 1 \le i \le k\}$ holds in the dataset of possibility degrees $\mathsf{D}$, and that $N_\mathcal{K}(\mathsf{t}) = 0$.

**Lemma 2.** *Given* $\mathcal{K} = \{(\phi_i \to \mathsf{t}, \alpha_i) \mid 1 \le i \le k\}$*, and* $\mathcal{F} = \mathcal{K} \cup \{(l_j, \alpha_j) \mid 1 \le j \le s\}$*, we get* $\Pi_\mathcal{F}(\neg\mathsf{t}) = \min\{\max\{1 - \alpha_i, \Pi_\mathcal{F}(\neg\phi_i) \mid 1 \le i \le k\}\}$.

*Proof.* By Lemma 1 and definition of $\mathcal{K}$, we have

$$\begin{aligned} N_\mathcal{F}(\mathsf{t}) &= \max\{\alpha \mid \mathcal{F} \cup \{(\neg\mathsf{t}, 1) \models (\bot, \alpha)\} \\ &= \max\{N_\mathcal{F}(\phi_i \wedge (\phi_i \to \mathsf{t})) \mid 1 \le i \le k\}. \end{aligned}$$

From **P2**, we get $N_\mathcal{F}(\mathsf{t}) = \max\{\min(N_\mathcal{F}(\phi_i), \alpha_i) \mid 1 \le i \le k\}$. Also, for any $\boldsymbol{x} \in [0,1]^n$, it holds that $(1 - \max(\boldsymbol{x})) = \min(1 - \boldsymbol{x})$, hence by the relation between possibility and necessity:

$$\begin{aligned} &\Pi_\mathcal{F}(\neg\mathsf{t}) = 1 - \max\{\min(N_\mathcal{F}(\phi_i), \alpha_i) \mid 1 \le i \le k\} \\ &= \min\{\max\{1 - \alpha_i, \Pi_\mathcal{F}(\neg\phi_i) \mid 1 \le i \le k\}. \end{aligned}$$ □

For convenience, we denote by $\psi_i \coloneqq l_{i,1} \vee \ldots \vee l_{i,s_i}$ the clause $\neg\phi_i$ for any rule $(\phi_i \to \mathsf{t}, \alpha_i) \in \mathcal{K}$. By Lemma 2 and **P1**, for any formula $\mathcal{X}$ we can compute $\Pi_{\mathcal{K}\cup\mathcal{X}}(\neg\mathsf{t})$ with

$$\begin{aligned} &\min\{\max\{1 - \alpha_i, \Pi_{\mathcal{K}\cup\mathcal{X}}(\psi_i) \mid 1 \le i \le k\}\} \qquad (1) \\ &\Pi_{\mathcal{K}\cup\mathcal{X}}(\psi_i) = \max(\Pi_{\mathcal{K}\cup\mathcal{X}}(l_{1,i}), \ldots, \Pi_{\mathcal{K}\cup\mathcal{X}}(l_{1,s_i})) \end{aligned}$$

By definition of $\mathcal{K}$, $\Pi_{\mathcal{K}\cup\mathcal{X}}(\psi_i) = \Pi_\mathcal{X}(\psi_i)$, so we can propagate the known uncertainty of input $\boldsymbol{x} \in \mathsf{D}$ to obtain the certainty degrees of the unknown target variable $\mathsf{t}$ with min-max operations. In practice, we do not know what rules $\phi_i \to \mathsf{t}$ hold in $\mathcal{K}$ and their necessity degree $\alpha_i$. But, such rules constrain every possibility degree in $\mathsf{D}$ that we can use to induce $\phi_i$ and $\alpha_i$, with $1 \le i \le k$. Now, we describe RIDDLE, a novel neural network architecture for Rule Induction leveraging the uncertainty propagation properties of Possibility Theory.

**Architecture**. We can alternatively compute $\Pi_{\mathcal{K}\cup\mathcal{X}}(\psi_i)$ as a parametrised combination of product and maximum operators:

$$\max(w_1^{\psi_i}\Pi_{\mathcal{X}}(\mathsf{v}_1), w_2^{\psi_i}\Pi_{\mathcal{X}}(\neg\mathsf{v}_1), \ldots, w_{2n}^{\psi_i}\Pi_{\mathcal{X}}(\neg\mathsf{v}_n))$$

where for odd (even) $1 \leq t \leq n$, $w_t^{\psi_i} \in [0,1]$ selects to what degree $\mathsf{v}_t$ ($\neg\mathsf{v}_t$) appears in $\psi_i$. In matrix notation with input $\boldsymbol{x} \in \mathsf{D}$, this operation becomes

$$f_\psi(\boldsymbol{x}) \coloneqq \boldsymbol{x} \star \boldsymbol{w}^\psi = \boldsymbol{x} \star [w_1^\psi, w_2^\psi, \ldots, w_{2n}^\psi]^T,$$

where $\star$ denotes the matrix dot product with the sum replaced by the $\mathsf{LSE}_{\max}$ operator. Lemma 3 states that for any clause $\psi$ and input $\boldsymbol{x}$, we can find $f_\psi : [0,1]^n \to [0,1]$ such that $\Pi_{\mathcal{X}}(\psi) = f_\psi(\boldsymbol{x})$.

**Lemma 3.** *For any clause $\psi$ and formula $\mathcal{X}$ over $\mathsf{V}$, there is $\boldsymbol{w}^\psi \in [0,1]^{|\mathsf{V}|}$, s.t. $f_\psi(\boldsymbol{x}) = \Pi_{\mathcal{X}}(\psi)$.*

*Proof.* Let $\psi \coloneqq l_1 \vee \cdots \vee l_s$. By **P1**, we have $\Pi_{\mathcal{X}}(\psi) = \max(\Pi_{\mathcal{X}}(l_1), \ldots, \Pi_{\mathcal{X}}(l_s))$. We can assign for odd $1 \leq t \leq n$, the value $\boldsymbol{w}_t^{\psi_i} = 1$ ($\boldsymbol{w}_{2t}^{\psi_i} = 1$) if $\mathsf{v}_t$ ($\neg\mathsf{v}_t$) appears as a top-level literal in the disjunct $\psi_i$, otherwise the value 0. By definition, we get $\max(\boldsymbol{w}_1^\psi\Pi_{\mathcal{X}}(l_1), \ldots, \boldsymbol{w}_{2n}^\psi\Pi_{\mathcal{X}}(\neg\mathsf{v}_n)) = f_\psi(\boldsymbol{x})$. □

As a consequence, we can approximate the computation of $\Pi_{\mathcal{K}\cup\mathcal{X}}(\neg\mathsf{t})$ in Eq. (1) with

$$\begin{aligned}&\mathsf{LSE}_{\min}(\mathsf{LSE}_{\max}(\boldsymbol{\beta}, \boldsymbol{x} \star [\boldsymbol{w}^{\psi_1}, \cdots \boldsymbol{w}^{\psi_k}]))\\ &= \mathsf{LSE}_{\min}(\mathsf{LSE}_{\max}(\boldsymbol{\beta}_i, f_{\psi_i}(\boldsymbol{x})) \mid 1 \leq i \leq k).\end{aligned} \tag{2}$$

The vector $\boldsymbol{\beta} \in [0,1]^k$ is the parameter that approximates $1 - \alpha_i$. Therefore, the rule induction problem of finding rules in $\mathcal{K}$ is reduced to selecting the right value of each parameter $\boldsymbol{w}^{\psi_i}$ and $\boldsymbol{\beta}_i$.

We can improve this method by exploiting the associative property of the $\mathsf{LSE}_{\max}$ operator and compute $f_\psi(\boldsymbol{x})$ as $\mathsf{LSE}_{\max}(f_{\Psi_1}(\boldsymbol{x}), \ldots, f_{\Psi_l}(\boldsymbol{x}))$, where each $\Psi_j$ is a subformula of $\psi$ (Example 4). Moreover, some rule antecedents $\phi_i, \phi_j$ (with $i \neq j$) in $\mathcal{K}$ may share subformulas, so we can decrease the number of parameters by stratifying each $f_{\psi_i}(\boldsymbol{x})$.

**Example 4.** *Given $\psi_1 \coloneqq \mathsf{v}_1 \vee \neg\mathsf{v}_2 \vee \mathsf{v}_3$, $\psi_2 \coloneqq \mathsf{v}_1 \vee \neg\mathsf{v}_2 \vee \mathsf{v}_4$, and possibilities $\boldsymbol{x} \in [0,1]^8$, we can compute $f_{\mathsf{v}_1\vee\neg\mathsf{v}_2}(\boldsymbol{x}) = \boldsymbol{x}\star\boldsymbol{w}^{\mathsf{v}_1\vee\neg\mathsf{v}_2}$, $f_{\mathsf{v}_3}(\boldsymbol{x}) = \boldsymbol{x}\star\boldsymbol{w}^{\mathsf{v}_3}$, and $f_{\mathsf{v}_4}(\boldsymbol{x}) = \boldsymbol{x}\star\boldsymbol{w}^{\mathsf{v}_4}$ at first and then compute $f_{\psi_1}(\boldsymbol{x}) = ([f_{\mathsf{v}_1\vee\neg\mathsf{v}_2}(\boldsymbol{x}), f_{\mathsf{v}_3}(\boldsymbol{x}), f_{\mathsf{v}_4}(\boldsymbol{x})] \star [1,1,0]^T)$. Similarly, $f_{\psi_2}(\boldsymbol{x}) = ([f_{\mathsf{v}_1\vee\neg\mathsf{v}_2}(\boldsymbol{x}), f_{\mathsf{v}_3}(\boldsymbol{x}), f_{\mathsf{v}_4}(\boldsymbol{x})] \star [1,0,1]^T)$.*

For $l \geq 1$, let $\mathsf{HL}_i^l : [0,1]^{l_h} \to [0,1]$ be the layer that takes as input $l_h$ arguments and for $\boldsymbol{x} \in \mathsf{D}$ computes $f_{\psi_i}(\boldsymbol{x})$ as follows

$$\begin{aligned}&\mathsf{HL}_i^l(\boldsymbol{x}) \coloneqq \mathsf{LSE}_{\max}\{\boldsymbol{w}_{i,j}^l \mathsf{HL}_j^{l-1}(\boldsymbol{x}) \mid 1 \leq j \leq l_h\}\\ &\mathsf{HL}_j^0(\boldsymbol{x}) \coloneqq \boldsymbol{x}_j,\end{aligned}$$

where each $1 \leq s \leq l$, $w_{i,j}^t \in [0,1]$. In other words, for $1 \leq s \leq l$ and $1 \leq j \leq l_h$, the function $\mathsf{HL}_j^s(\boldsymbol{x})$ computes $\Pi_{\mathcal{X}}(\Psi)$ for a subformula $\Psi$ of $\psi_i$, in the same way $f_{\psi_i}(\boldsymbol{x})$ computes $\Pi_{\mathcal{X}}(\psi_i)$. Each layer $\mathsf{HL}_j^s : [0,1]^u \to [0,1]^v$, $1 \leq s \leq l$, is a function with $u, v \geq 1$ freely chosen (hyperparameters) that obey the constraint posed by the standard matrix dot product. Finally, we can define $\mathsf{RIDDLE}(\boldsymbol{x})$ as

$$\mathsf{LSE}_{\min}(\mathsf{LSE}_{\max}(\boldsymbol{\beta_i}, \mathsf{HL}_i^l(\boldsymbol{x})) \mid 1 \leq i \leq k)). \tag{3}$$

**Theorem 1.** *Given $\mathcal{K} = \{(\phi_i \to \mathsf{t}, \alpha_i) \mid 1 \leq i \leq k\}$ with $N_{\mathcal{K}}(\mathsf{t}) = 0$, we can find a configuration of the parameters in $\mathsf{RIDDLE}$ s.t. for any formula $\mathcal{X} = \{(l_j, \alpha_j) \mid 1 \leq j \leq s\}$, $\mathsf{RIDDLE}(\boldsymbol{x}) = \Pi_{\mathcal{K}\cup\mathcal{X}}(\neg\mathsf{t})$.*

*Proof.* For all literals $l$, by definition $N_{\mathcal{K}}(l) = 0$. Hence, for any antecedent $\psi$, $\Pi_{\mathcal{K}\cup\mathcal{X}}(\psi_i) = \Pi_{\mathcal{X}}(\psi)$. By Lemma 3 and associativity of max, we can set the values of the parameters in $\mathsf{HL}_i^l(\boldsymbol{x})$ so that it computes $\Pi_{\mathcal{X}}(\phi_i)$. By Eqs. (2) and (3), we get that $\mathsf{RIDDLE}(\boldsymbol{x})$ computes $\Pi_{\mathcal{K}\cup\mathcal{X}}(\neg\mathsf{t})$ as in Eq. (1). □

Multi-class tasks are modelled with many output nodes or with multiple $\mathsf{RIDDLE}$ instances in parallel. Theorem 1 shows the generality of our approach but it relies on Lemma 3 which requires parameters in $[0,1]$. Thus, after updating the parameters with SGD [3], we replace negative values with 0 and values greater than 1 with 1.

**Rule Extraction and Injection**. When the optimisation procedure terminates, we can track the literals that are used to discriminate the target possibility value by inspecting the value of each parameter. Indeed, every parameter lies in the interval $[0,1]$ and by the semantics given to their values, we can apply the argument in Lemma 3, to extract the literals included in the clause of each layer $\mathsf{HL}_i^l$. We observed that the parameters always collapse to either 0 or 1 after a sufficient number of updates (Section 4). Also, the introduction of hidden layers can be considered a way of having predicate invention as in ILP settings [25].

We can also manually inject rules of the form $\phi \rightarrow \mathsf{t}$ to a RIDDLE instance before or after training. Due to Lemma 3, we just need to append to the operation $\mathsf{LSE}_{\min}$ in Eq. (3) a layer $\mathsf{HL}' : [0,1]^{2n} \rightarrow [0,1]$, corresponding to the function $f_{\neg\phi}$.

# 4 Experimental Results

We implemented the RIDDLE model in Python 3.9 that is fully integrated in the PyTorch [26] ecosystem. The implementation is freely available at the following link: `https://git.app.uib.no/Cosimo.Persia/riddle`. The gradient of the model parameters are computed with PyTorch's automatic differentiation package and after the update, they are 'clamped' to the range $[0,1]$ to preserve the correctness of the model. We conduct the experiments on an Ubuntu 18.04.5 LTS server with i9-7900X CPU at 3.30GHz, 32 physical cores, 8 GPUs NVIDIA A100 with 80GB, and 32GB RAM.

**Test settings**. Often, the features in the considered datasets include a mixture of nominal, continuous, and integer fields. Using feature discretization, we divide continuous or integer values in 8 bins such that all bins for each feature have the same number of points. Each bin will be associated with a new variable that it is going to be set to 'true' if the value of the original value belongs to the respective bin. Missing values assign the value 'unknown' to all related new binarised variables. In this way, we can generate a set of interpretations $D \coloneqq \{\mathcal{I}_1, \ldots, \mathcal{I}_d\}$. From $D$, we can derive the dataset $\mathsf{D} \coloneqq \{\boldsymbol{x}^i \mid \mathcal{I}_i \in D\}$ as explained in Section 3. $\mathsf{D}$ expresses the possibility values of variables and their negation for each interpretation in $D$. The first column in Table 1, shows the datasets that we considered for the benchmark. These are freely available at UCI machine learning repository [8]. Briefly, with 'breast cancer', 'hepatitis', 'horse', 'hypothyroid', 'lymphography', and 'primary tumor', the model should predict the type of disease or the patient's survival. With 'auto', 'credit', 'chess', 'glass', 'mushroom', and 'wine', the model should classify the specific type of object under scrutiny; for example, whether a mushroom is edible. More details about these can be found at the UCI website[2]. Most datasets have a substantial amount of missing values.

**Evaluation**. RIDDLE minimises the MSE (mean squared error) during training (see column 'MSE' in Table 1). Compared with arbitrary linear/ReLu feedforward deep network architectures, RIDDLE performs slightly better (in addition to being explainable). Therefore, we will focus our comparison on the accuracy of the FURIA algorithm [20] that represents the state-of-the-art in propositional Rule Induction with confidence values. We use the FURIA implementation available on Weka [17], and compare it with RIDDLE on classification tasks using the standard definition of accuracy. To use the trained RIDDLE model for classification, we look at RIDDLE output $(\Pi(\neg\mathsf{t}_1), \ldots, \Pi(\neg\mathsf{t}_m))$ and if $\Pi(\neg\mathsf{t}_i) \leq 0.4$, then the variable $\mathsf{t}_i$ is preferred over its negation and we assume that the variable $\mathsf{t}_i$ is predicted to be true ($N(\mathsf{t}) = 1 - \Pi(\neg\mathsf{t})$). We carried our tests on the same benchmark datasets used by the aforementioned rule induction system.

**Model selection**. We split each dataset in 70%, 10%, and 20% for training, validation and test, respectively. Finding the best combinations of hyperparameters (number of layers, nodes per layers, learning rate etc.) can be a time-consuming task. But, we noticed that, in general, RIDDLE performs well even with small networks and extremely well with deeper configurations. We adjust the hyperparameters on a grid search fashion using Tune [23]. The number of hidden layers varies from 1 to 10 and the number of nodes per layer from 2 to 500, the batch size from 8 to 64, and the learning rate from 0.1 to 0.001. The final models' sizes correlate positively with the number of variables given as input and with the number of rules that hold in the dataset. On average, the resulting network has 6 layers with 50 nodes. Each model is trained with a batch of size 16 over 100 epochs with a learning rate of 0.01, and with early-stopping. That is, we stop the training routine if the validation loss has not has not decreased by more than 0.01 for 10 subsequent epochs. Moreover, we fixed a weight-decay factor of 0.001. This means that the gradients used for updating the parameters are summed to the constant value 0.001.

**Results**. Table 1 shows the results of our experiments. The column 'Inst.' shows the number of instances, #V shows the number of variables in the original dataset and #DV is the num-

[2] `http://archive.ics.uci.edu/ml`

| | | | | | Accuracy | | Model complexity | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | RIDDLE | FURIA | RIDDLE | | FURIA | |
| Dataset | Inst. | #V | #DV | MSE | | | Size | Count | Size | Count |
| Anneal | 798 | 39 | 286 | 1e-4 | 97 | 97 | **2.9** | **17.1** | 3.2 | 21.7 |
| Audiology | 226 | 71 | 708 | 4e-3 | **95** | 91 | **1.7** | **19.0** | 2.1 | 19.3 |
| Auto | 205 | 26 | 354 | 3e-4 | **98** | 85 | 2 | 3.4 | 2 | **3.3** |
| Credit-A | 690 | 15 | 158 | 2e-1 | 87 | **89** | **3.1** | **6.8** | 4.1 | 7.3 |
| Credit-G | 1000 | 21 | 166 | 2e-2 | **74** | 72 | **2.2** | 10.3 | 3.1 | **10.2** |
| Breast cancer | 699 | 21 | 160 | 3e-2 | **91** | 90 | **3.3** | **5.1** | 3.6 | 9.9 |
| Chess | 3196 | 37 | 146 | 1e-2 | 91 | **99** | **4.6** | **15.3** | 4.7 | 28.3 |
| Glass | 214 | 10 | 72 | 4e-3 | **94** | 68 | **1.8** | **5.9** | 2.0 | 16.7 |
| Hepatitis | 155 | 20 | 122 | 4e-4 | **86** | 75 | 3.1 | 3.4 | **2.2** | 4.1 |
| Horse | 368 | 28 | 252 | 5e-3 | 83 | **85** | **1.8** | 5.9 | 2.7 | **5.0** |
| Hypothyroid | 3163 | 30 | 140 | 8e-3 | 95 | 95 | **4.8** | **11.3** | 6.3 | 18.1 |
| Lymphography | 148 | 19 | 118 | 9e-4 | 86 | 86 | **2.1** | 5.4 | 2.2 | **5.3** |
| Mushroom | 8124 | 19 | 234 | 1e-4 | 97 | **98** | 2.8 | **5.3** | **2.4** | 6.0 |
| Primary tumor | 339 | 18 | 60 | 6e-5 | **94** | 75 | 1.9 | **3.6** | 1.9 | 4.0 |
| Wine | 178 | 14 | 112 | 1e-4 | **96** | 90 | 3.4 | 8.2 | 3.4 | **8.1** |

Table 1: Accuracy and model complexity of RIDDLE and FURIA compared with different datasets

ber variables after discretization and binarization. 'MSE' is the loss of RIDDLE on the test data. Often, RIDDLE converges to a minimum after only 30 epochs and soon after the early-stopping routine stops the training. The average training time with the largest datasets (mushrooms, chess, and hypothyroid) is 2 seconds for RIDDLE and 1 for FURIA. The 'Accuracy' columns in Table 1 show that in most cases RIDDLE generalises from training data better than FURIA. However, FURIA performs better in complete information scenarios as in the 'chess' dataset. But, even in such case RIDDLE outputs a simpler model. In contrast, RIDDLE yields better accuracy in datasets with more missing data, such as 'glass', or 'hepatitis'.

The rightmost part of Table 1 reports the size and the number of induced clauses found by each model per dataset. For instance, rules found by RIDDLE in the 'hepatitis' are

$$((1.9 \leq \mathsf{bilirubin}) \text{ and } \neg\mathsf{has\_ascites} \rightarrow \mathsf{dies}, 0.8)$$
$$((3.7 \leq \mathsf{albumin}) \text{ and } \mathsf{firm\_liver} \rightarrow \mathsf{lives}, 1).$$

Moreover, RIDDLE has better model complexity with fewer and shorter rules, even when RIDDLE performed worse in terms of accuracy. This suggests a bias towards simpler models. When RIDDLE encodes known rules with additional layers (end of Section 3), the performance improves.

| Dataset | Min | Max | Median | S.D. |
|---|---|---|---|---|
| Anneal | 93 | 98 | 96 | 1.18 |
| Audiology | 94 | 96 | 95 | 0.63 |
| Auto | 95 | 99 | 98 | 2.48 |
| Credit-A | 66 | 92 | 86 | 4.12 |
| Credit-G | 68 | 82 | 76 | 5.65 |
| Breast cancer | 85 | 96 | 91 | 4.52 |
| Chess | 83 | 98 | 88 | 6.04 |
| Glass | 81 | 99 | 88 | 4.54 |
| Hepatitis | 71 | 89 | 86 | 6.83 |
| Horse | 78 | 86 | 84 | 5.71 |
| Hypothyroid | 91 | 98 | 94 | 4.37 |
| Lymphography | 83 | 88 | 87 | 2.91 |
| Mushroom | 94 | 98 | 97 | 1.17 |
| Primary tumor | 85 | 99 | 93 | 5.13 |
| Wine | 84 | 98 | 95 | 3.59 |

Table 2: Additional statistics from the empirical evaluation of RIDDLE obtained by running 40 training instances.

Table 2 shows the statistical results concerning the experiments. For each dataset, we stored the minimal and maximal accuracy. Then, we computed the median and standard deviation (S.D.). From Table 2 we can conclude that RIDDLE has consistent performance in most of the datasets considered.

We remark that another advantage of RIDDLE is that the values provided with the rules have a clear meaning, in terms of necessity. Meanwhile, fuzzy approaches such as FURIA provide a weaker foundation for the interpretation of the values associated with the rules. Additionally, the necessity values also distinguish RIDDLE from approaches such as decision trees which, usually, do not provide a measure of a rule's reliability.

# 5 Conclusion

In this work, we introduced RIDDLE: a novel deep learning architecture specialised in performing Rule Induction in the presence of incomplete or uncertain data. RIDDLE is a white-box model as its trained weights have a clear meaning concerning the decisions that the model takes while performing inference on the input. These weights can be translated into propositionally complete rules that are simpler than the rules found by state-of-the-art algorithms. In addition, each rule is associated with certainty degree expressing the confidence of the model about the induced rule. Not only that, RIDDLE can also seamlessly incorporate background knowledge via rule injection. Thus, RIDDLE provides an efficient, flexible, and interpretable solution for Rule Induction.

**Future work**. The next step is to optimise the matrix computation in RIDDLE's implementation and speed-up both training and inference time. Also, we will evaluate the effect of different methods of drawing possibilities distributions from imprecise data [10, 13] on accuracy.

# Acknowledgements

Part of this work has been done in the context of CEDAS (Center for Data Science, University of Bergen, Norway). The first author is supported by the "L. Meltzers Høyskolefond" utdeling 2022. The second author is supported by the ERC project "Lossy Preprocessing" (LOPRE), grant number 819416, led by Prof Saket Saurabh.

# References

[1] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, Apr 1988. doi: 10.1023/A:1022821128753. URL `https://doi.org/10.1023/A:1022821128753`.

[2] J. L. J. L. Bell and M. Machover. *A course in mathematical logic / by J. L. Bell and M. Machover.* North-Holland, Amsterdam, 1977. ISBN 0720428440. doi: https://doi.org/10.1007/978-1-4757-4385-2.

[3] L. Bottou. Stochastic learning. In O. Bousquet and U. von Luxburg, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004. doi: 10.1007/978-3-540-28650-9_7.

[4] I. Bratko. Applications of machine learning: Towards knowledge synthesis. *New Generation Computing*, 11(3):343–360, Sep 1993. ISSN 1882-7055. doi: 10.1007/BF03037182.

[5] W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pages 115–123. Morgan Kaufmann, 1995. doi: 10.1016/b978-1-55860-377-6.50023-2.

[6] S. Dash, O. Günlük, and D. Wei. Boolean decision rules via column generation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4660–4670, 2018. doi: https://doi.org/10.48550/arXiv.1805.09901.

[7] J. A. Dhanraj, M. Prabhakar, C. P. Ramaian, M. Subramaniam, J. M. Solomon, and N. Vinayagam. Increasing the Wind Energy Production by Identifying the State of Wind Turbine Blade. In *Lecture Notes*

*in Mechanical Engineering*, pages 139–148. Springer Nature Singapore, 2022. doi: 10.1007/978-981-16-7909-4_13.

[8] D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[9] D. Dubois and H. Prade. Resolution principles in possibilistic logic. *Int. J. Approx. Reasoning*, 4(1):1–21, 1990. doi: https://doi.org/10.1016/0888-613X(90)90006-N.

[10] D. Dubois and H. Prade. When upper probabilities are possibility measures. *Fuzzy Sets and Systems*, 49:65–74, 1992. doi: https://doi.org/10.1016/0165-0114(92)90110-P.

[11] D. Dubois and H. Prade. *Possibility Theory*, pages 6927–6939. Springer New York, New York, NY, 2009. ISBN 978-0-387-30440-3. doi: 10.1007/978-0-387-30440-3_413.

[12] D. Dubois and H. Prade. Possibility theory and its applications: Where do we stand? In *Springer Handbook of Computational Intelligence*, pages 31–60. 2015. doi: 10.1007/978-3-662-43505-2_3.

[13] D. Dubois and H. Prade. Practical methods for constructing possibility distributions. *International Journal of Intelligent Systems*, 31(3): 215–239, 2016. doi: https://doi.org/10.1002/int.21782.

[14] D. Dubois, J. Lang, and H. Prade. *Possibilistic Logic*, page 439–513. Oxford University Press, Inc., 1994.

[15] M. Elkano, J. A. Sanz, E. Barrenechea, H. Bustince, and M. Galar. CFM-BD: A distributed rule induction algorithm for building compact fuzzy models in big data classification problems. *IEEE Transactions on Fuzzy Systems*, 28(1):163–177, jan 2020. doi: 10.1109/tfuzz.2019.2900856.

[16] N. Fatima, A. S. Imran, Z. Kastrati, S. M. Daudpota, and A. Soomro. A systematic literature review on text generation using deep neural network models. *IEEE Access*, 10: 53490–53503, 2022. doi: 10.1109/access.2022.3174108.

[17] E. Frank, M. A. Hall, G. Holmes, R. Kirkby, B. Pfahringer, and I. H. Witten. *Weka: A machine learning workbench for data mining.*, pages 1305–1314. Springer, Berlin, 2005. URL `http://researchcommons.waikato.ac.nz/handle/10289/1497`.

[18] C. Glanois, Z. Jiang, X. Feng, P. Weng, M. Zimmer, D. Li, W. Liu, and J. Hao. Neuro-symbolic hierarchical rule induction. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 7583–7615. PMLR, 2022. doi: arxiv-2112.13418.

[19] I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. doi: https://doi.org/10.1007/s10710-017-9314-z.

[20] J. Hühn and E. Hüllermeier. Furia: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19(3): 293–319, Dec 2009. ISSN 1573-756X. doi: 10.1007/s10618-009-0131-8.

[21] C. Joslyn. Towards an empirical semantics of possibility through maximum uncertainty. 1991.

[22] R. Kusters, Y. Kim, M. Collery, C. de Sainte Marie, and S. Gupta. Differentiable rule induction with learned relational features. Jan. 2022. doi: arXiv:2201.06515.

[23] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica. Tune: A research platform for distributed model selection and training. 2018. doi: https://arxiv.org/abs/1807.05118.

[24] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. doi: 10.1109/tpami.2021.3059968.

[25] S. Muggleton, L. De Raedt, D. Poole, I. Bratko, P. Flach, K. Inoue, and A. Srinivasan. Ilp turns 20. *Machine Learning*, 86

(1):3–23, Jan 2012. ISSN 1573-0565. doi: 10.1007/s10994-011-5259-2.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[27] V. Podgorelec, P. Kokol, B. Stiglic, and I. Rozman. Decision trees: An overview and their use in medicine. *Journal of Medical Systems*, 26(5):445–463, Oct 2002. ISSN 1573-689X. doi: 10.1023/A:1016409317640.

[28] L. Qiao, W. Wang, and B. Lin. Learning accurate and interpretable decision rule sets from neural networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 4303–4311. AAAI Press, 2021. doi: https://arxiv.org/abs/2103.02826.

[29] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, 1(5):206–215, 2019. doi: 10.1038/s42256-019-0048-x.

[30] G. Scala, A. Federico, V. Fortino, D. Greco, and B. Majello. Knowledge generation with rule induction in cancer omics. *International Journal of Molecular Sciences*, 21(1):18, dec 2019. doi: 10.3390/ijms21010018.

[31] J. Vreeken, M. van Leeuwen, and A. Siebes. Krimp: mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1):169–214, Jul 2011. ISSN 1573-756X. doi: 10.1007/s10618-010-0202-x.

[32] L. WJ. A rule-based process control method with feedback. *Advances in Instrumentation 41, 169–175*, 1987.

[33] P. Xu, Z. Ding, and M. Pan. A hybrid interpretable credit card users default prediction model based on RIPPER. *Concurrency and Computation: Practice and Experience*, 30 (23):e4445, feb 2018. doi: 10.1002/cpe.4445.