CONFIDENCE NEURAL LOGIC NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks achieve strong predictive performance, but their decisions are often opaque, which is problematic in domains that require explicit, human-understandable reasoning. Neural rule learners address this by learning logical rules from data, yet they typically aggregate rules through a linear layer that has no formal logical meaning, leaving unclear what reasoning the trained network performs. We introduce Confidence Neural Logic Networks (CNLNet) and prove that conjunctive clauses followed by a linear layer exactly realize a weighted logic program: the forward pass performs weighted-satisfiability inference, and the predicted class provably coincides with the one favored by the program. Clause structure is learned end-to-end with differentiable L_0 gates. We extend this correspondence to depth with the Confidence Neural Logic Block, whose hidden units are themselves weighted-satisfiability decisions, so that deep networks remain layered logic programs, and we compose blocks into class-conditional chains that avoid gradient interference across classes. On 18 tabular and image benchmarks with up to 11M samples, CNLNet achieves a better average rank than interpretable baselines, including RRL (outperformed on 16 datasets), C4.5, SATNet, and NeuroLogic, and is comparable to strong black-box models such as PLNN, LightGBM, and XGBoost.

1 INTRODUCTION

Neural networks achieve remarkable accuracy, but their decisions remain hard to understand and verify, which is problematic wherever predictions must be justified by explicit reasoning. Logic offers such reasoning through explicit rules and principled inference, yet symbolic systems struggle to learn from noisy, high-dimensional data, while neural networks learn efficiently but opaquely. This raises a central question: how can we build neural networks whose internal structure directly represents logical rules, so that inference is performed by, not merely approximated with, the network?

Early approaches encode prior knowledge into network weights (Towell & Shavlik, 1993; d'Avila Garcez & Zaverucha, 1999) or learn rules within predefined relational templates (Evans & Grefenstette, 2018; Dong et al., 2019), which limits them to settings where such knowledge is available. More recent rule learners learn interpretable rules directly from data (Wang et al., 2020; 2021; Qiao et al., 2021), but two challenges remain. First, to improve accuracy they typically aggregate shallow rule sets with a linear layer (Wang et al., 2021) that has no formal logical reading, so it is unclear what reasoning the trained network actually performs. Second, logical rules are discrete and hard to optimize with gradients, and this difficulty compounds when rules are composed across layers.

We introduce *Confidence Neural Logic Networks* (CNLNet), which give every layer an exact logical meaning. Our contributions are:

- **Exact semantics.** We prove that conjunctive clauses followed by a linear layer exactly realize a weighted logic program: the forward pass performs weighted-satisfiability inference, and the predicted class provably coincides with the one favored by the program for any weights. Clause structure is learned end-to-end with differentiable L_0 gates, and the evaluation-time network computes exactly the extracted program.
- **Deep CNLNet.** We propose the Confidence Neural Logic Block, whose hidden units are themselves weighted-satisfiability decisions, so that deep networks remain layered logic programs. Batch normalization and a residual default rule keep the stack trainable, and class-conditional chains remove gradient interference across classes.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- **Evaluation.** On 18 tabular and image benchmarks with up to 11M samples, CNLNet achieves a better average rank than interpretable baselines (RRL, C4.5, SATNet, NeuroLogic) and is comparable to black-box models such as PLNN, LightGBM, and XGBoost. Pruning reduces the learned programs without loss of accuracy, and Shallow inference is faster than RRL on every dataset.

2 Related Work

Differentiable rule learning. Closest to CNLNet are networks whose forward pass encodes Boolean rules and is trained end-to-end. MLLP and RRL (Wang et al., 2020; 2021; 2024) stack conjunction and disjunction layers trained with random binarization and gradient grafting; DR-Net (Qiao et al., 2021) learns DNF rule sets with L_0-gated literals (Louizos et al., 2018); Net-DNF (Katzir et al., 2021) uses soft DNF blocks for tabular data; RL-Net (Dierckx et al., 2023) and Neural Rule Lists (Xu et al., 2025) learn ordered rule lists; and follow-ups improve feature binarization (Zhang et al., 2023) and rule diversity (Yang et al., 2024). The Tsetlin machine (Granmo, 2018) learns conjunctive clauses that cast signed votes for classes, while differentiable logic gate networks (Petersen et al., 2022; 2024; Yue & Jha, 2024) learn a distribution over gates at each neuron and discretize after training. Relational variants learn first-order rules (Evans & Grefenstette, 2018; Yang et al., 2017a; Sadeghian et al., 2019; Dong et al., 2019; Qu et al., 2021; Glanois et al., 2022; Gao et al., 2025). These methods either discretize after training, so that the trained network and the extracted rules can disagree, or aggregate rules through a final linear layer without a formal logical reading. CNLNet instead proves that its output layer is the exact realization of a weighted logic program, extends this reading to every hidden unit of its deep blocks, and composes depth through class-conditional chains.

Weighted logic and neural reasoning. Selecting the assignment that maximizes the total weight of satisfied formulas is MAP inference in Markov logic networks (Richardson & Domingos, 2006), i.e., weighted MaxSAT; probabilistic soft logic (Bach et al., 2017) relaxes it to continuous truth values. Logical Neural Networks (Riegel et al., 2020; Sen et al., 2022) map formulas to neurons with weighted real-valued connectives, DeepProbLog and Scallop (Manhaeve et al., 2018; Li et al., 2023b) attach neural predicates to a given logic program, and SATNet (Wang et al., 2019) embeds a differentiable MaxSAT relaxation from which rules can be extracted (Li et al., 2023a; Geng et al., 2025). CNLNet learns both the formulas and their weights from data without a given program, and computes MAP inference exactly, rather than through a relaxation, with a single forward pass.

Logic as a training signal. Semantic loss (Xu et al., 2018), semantic-based regularization (Diligenti et al., 2017; 2012), Logic Tensor Networks (Serafini & d'Avila Garcez, 2016; Donadello et al., 2017), DL2 (Fischer et al., 2019), constraint-based learning (Ciravegna et al., 2020), and posterior regularization (Hu et al., 2016) add logical constraints as auxiliary losses. The resulting network is encouraged, but not required, to satisfy the rules; the rules guide training but do not perform inference.

Discrete rule learning and interpretability by design. Decision trees (Quinlan, 1993; Breiman et al., 1984; Lin et al., 2020), rule induction (Cohen, 1995), rule lists (Yang et al., 2017b; Angelino et al., 2018), rule sets (Wang et al., 2017; Lakkaraju et al., 2016; Dash et al., 2018), and inductive logic programming (Muggleton, 1995; Muggleton & de Raedt, 1994; Srinivasan et al., 1994) are interpretable by construction, but rely on combinatorial search that scales poorly with data size and rule complexity. Post-hoc explainers such as LIME, SHAP, and Anchors (Ribeiro et al., 2016; Lundberg & Lee, 2017; Ribeiro et al., 2018) explain opaque models without guaranteeing fidelity to them, motivating inherently interpretable models (Rudin, 2019). Concept-based models (Koh et al., 2020; Zarlenga et al., 2022; Ciravegna et al., 2023; Barbiero et al., 2023) route predictions through human-defined concepts, and some derive logic rules over them, but require concept supervision.

Multi-task optimization methods mitigate conflicting gradients across heads that share parameters, by multi-objective descent (Sener & Koltun, 2018), loss reweighting (Chen et al., 2018), or gradient manipulation (Yu et al., 2020; Liu et al., 2021; Navon et al., 2022). These methods adjust the optimizer while all parameters remain shared; CNLNet's class-conditional chains remove the sharing, and hence the interference, architecturally.

3 METHODOLOGY

3.1 LOGIC PROGRAMS AS CONFIDENCE NEURAL NETWORKS

We study C-way classification over inputs reduced to N Boolean variables via discretization (quantile thresholds for continuous features, one-hot for categorical ones), yielding $\mathbf{x} \in \{0,1\}^N$; a class $c \in \{1,\dots,C\}$ is represented by the one-hot vector $\mathbf{y}(c) \in \{0,1\}^C$, whose c-th coordinate $y^{(c)}$ is 1 and whose remaining coordinates $y^{(c')}$ ($c' \neq c$) are 0. A *literal* is x_i or $\neg\mathrm{x}_i$; a *conjunctive clause* $s^\wedge$ is a conjunction of literals, satisfied by $\mathbf{x}$ ($\mathbf{x} \models s^\wedge$) iff every literal in it holds. A *propositional logic program* $\mathcal{P}$ is a finite set of formulas relating clauses to class indicators, each carrying a non-negative *confidence* γ; classification is the search for the class assignment with the strongest total confidence-weighted support from these formulas — its *weighted-satisfiability score* (see Section 3.2).

Symbolic representation The formulas of $\mathcal{P}$ are indexed by a conjunctive clause $j = 1,\dots,R$ and a class $c = 1,\dots,C$. Each formula is a weighted implication

$$\gamma_j^{(c)} :\ h_{jc} \leftarrow s_j^\wedge, \qquad s_j^\wedge = \bigwedge_{t\in\mathcal{T}_j} x_t \wedge \bigwedge_{k\in\mathcal{K}_j} \neg x_k, \tag{1}$$

where the head $h_{jc} \in \{y^{(c)}, \neg y^{(c)}\}$ is either the class indicator or its negation, and $\gamma_j^{(c)} \geq 0$ is its confidence. The conjunctive clause $s_j^\wedge$ is built over the N input variables, with disjoint index sets $\mathcal{T}_j, \mathcal{K}_j \subseteq \{1,\dots,N\}$ for the positive and negated literals, respectively. Crucially, $s_j^\wedge$ does not depend on c: the same R conjunctive clauses are shared across all classes, and only the head polarity and the confidence are class-specific.

Connectionist representation We realize $\mathcal{P}$ as a two-layer network. The lower layer consists of R AND-nodes parameterized by a ternary matrix $\mathbf{W}^\wedge \in \{-1,0,1\}^{N\times R}$, whose column j encodes $s_j^\wedge$: $w_{ij}^\wedge = 1$ if $i \in \mathcal{T}_j$, -1 if $i \in \mathcal{K}_j$, and 0 otherwise. Node j outputs $a_j = \mathbb{I}_{\mathbf{x}\models s_j^\wedge}$. The upper layer is a real-valued matrix $\mathbf{W} \in \mathbb{R}^{R\times C}$ with entries

$$w_{jc} = \begin{cases} \gamma_j^{(c)} & \text{if } h_{jc} = y^{(c)}, \\ -\gamma_j^{(c)} & \text{if } h_{jc} = \neg y^{(c)}, \end{cases} \tag{2}$$

so that $\mathrm{sign}(w_{jc})$ encodes the head polarity and $|w_{jc}|$ its confidence. These two layers form a *shallow* CNLNet, illustrated in Figure 1. Each class additionally has a bias b_c, which we treat as the confidence of an always-satisfied clause with an empty body, $s_0^\wedge = \top$, $a_0 \equiv 1$; we leave it implicit in the sums below.

3.2 REASONING VIA SATISFIABILITY SCORING

Unlike traditional rule-based or probabilistic inference procedures, CNLNet performs reasoning by solving a weighted Boolean satisfiability problem through its network representation. Given an input assignment $\mathbf{x}$, classification is formulated as the search for the Boolean one-hot assignment of the target variables $(y^{(1)},\dots,y^{(C)})$ that maximizes the total weight of the satisfied logical formulas. Specifically, a Boolean one-hot assignment $\mathbf{y}(c)$ sets $y^{(c)} = 1$ (*True*) and $y^{(c')} = 0$ (*False*) for every $c' \neq c$.

Let $\mathcal{P}$ consist of the weighted formulas in (1), for an input $\mathbf{x}$ and a class c, the weighted-satisfiability score is

$$f(\mathbf{x},\mathbf{y}(c)) = \sum_{j=1}^{R}\sum_{c'=1}^{C} \gamma_j^{(c')}\, \mathbb{I}_{(\mathbf{x},\mathbf{y}(c))\models\phi_{jc'}},$$

where $\mathbf{y}(c)$ is the one-hot assignment with $y^{(c)} = 1$ and $y^{(c')} = 0$ for all $c' \neq c$. The shallow CNLNet computes

$$g(\mathbf{x})_c = \sum_{j=1}^{R} w_{jc}\, I_j, \qquad I_j = \mathbb{I}_{\mathbf{x}\models s_j^\wedge},$$

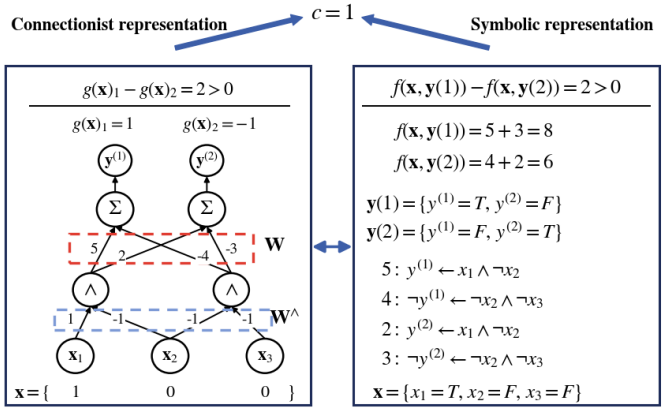


Figure 1: Shallow CNLNet as the exact network realization of a weighted logic program. *Left:* positive (1) and negative (-1) edges are the positive and negated literals of each conjunctive clause ($\mathbf{W}^{\wedge}$); real-valued edges carry the weights from class confidences ($\mathbf{W}$); a worked example—each AND-node fires ($a_j = 1$) if and only if its conjunctive clause is satisfied, and $\arg\max_c g(\mathbf{x})_c$ is the class with the highest output. *Right:* equivalent reasoning steps—each weighted formula relates a conjunctive clause to a class literal; for each candidate class assignment $\mathbf{y}(c)$, the weighted-satisfiability score $f(\mathbf{x}, \mathbf{y}(c))$ sums the confidences of the formulas satisfied by $(\mathbf{x}, \mathbf{y}(c))$, and the class with the highest score is selected. Both sides yield the same score difference ($2 > 0$) and thus the same prediction, $c = 1$.

Theorem 3.1 (Equivalence)**.** *For every input* $\mathbf{x}$ *there exists a constant* $B(\mathbf{x})$*, independent of* c*, such that*

$$f(\mathbf{x}, \mathbf{y}(c)) = B(\mathbf{x}) + g(\mathbf{x})_c \qquad \textit{for all } c \in \{1, \dots, C\}.$$

Consequently, for all c, c'*,*

$$g(\mathbf{x})_c - g(\mathbf{x})_{c'} = f(\mathbf{x}, \mathbf{y}(c)) - f(\mathbf{x}, \mathbf{y}(c')), \qquad \textit{and} \qquad \arg\max_c g(\mathbf{x})_c = \arg\max_c f(\mathbf{x}, \mathbf{y}(c)).$$

Proof. The proof is given in Appendix A. □

Remark (multiple formulas per pair). If $\mathcal{P}$ contains both $y^{(c)} \leftarrow s_j^{\wedge}$ with confidence γ^+ and $\neg y^{(c)} \leftarrow s_j^{\wedge}$ with confidence γ^-, the same argument applies with $w_{jc} = \gamma^+ - \gamma^-$. Likewise, the bias b_c is the weight w_{0c} of the clause $s_0^{\wedge} = \top$ (with $I_0 = 1$), so the theorem holds unchanged with the bias included; the extracted program lists it as a formula with an empty body (Appendix I).

3.3 Learning Conjunctive Clause Structure with Differentiable Gates

Choosing $\mathbf{W}^{\wedge}$ is a discrete structure-search problem: which literals to include in each clause, and with what polarity. Instead of fixing this structure a priori or searching it combinatorially, we learn it end-to-end with L_0 stochastic gates (Louizos et al., 2018). Each literal i in conjunctive clause j has a free real weight v_{ij} and a gate parameter θ_{ij}; during training,

$$\xi \sim \mathrm{Unif}(0,1), \quad \bar{z}_{ij} = \sigma\Big(\tfrac{\log \xi - \log(1-\xi) + \theta_{ij}}{\beta}\Big)(\zeta - \eta) + \eta, \quad z_{ij} = \min(1, \max(0, \bar{z}_{ij})), \qquad (3)$$

a differentiable "hard-concrete" relaxation of a Bernoulli gate, where σ is the logistic sigmoid and (β, η, ζ) are fixed hyperparameters — a temperature $\beta > 0$ and stretch constants $\eta < 0 < 1 < \zeta$ that let the gate reach exact 0 and 1. Its expected-L_0 penalty $\Pr[z_{ij} \neq 0] = \sigma\big(\theta_{ij} - \beta \log(-\eta/\zeta)\big)$ serves as the sparsity regularizer; at evaluation the noise and the temperature are dropped, giving the deterministic gate $z_{ij} = \min\big(1, \max\big(0, \sigma(\theta_{ij})(\zeta - \eta) + \eta\big)\big)$. The extracted logic program keeps literal i in clause j iff $z_{ij} > 0$, which sets the gated weight $\tilde{v}_{ij} = z_{ij} v_{ij}$ and recovers the ternary clause matrix $\mathbf{W}^{\wedge}$ through its sign, $w_{ij}^{\wedge} := \mathbb{I}_{z_{ij}>0} \operatorname{sign}(v_{ij})$ (distinct from the real-valued confidence weights w_{jc} of Eq. (2)).

The AND-node score for clause j is

$$r_j(\mathbf{x}) = 1 + \sum_{i=1}^{N} \tilde{v}_{ij}\, x_i - \sum_{i=1}^{N} \max(\tilde{v}_{ij}, 0) = 1 - \sum_{i:\tilde{v}_{ij}>0} \tilde{v}_{ij}(1 - x_i) \; - \sum_{i:\tilde{v}_{ij}<0} |\tilde{v}_{ij}|\, x_i, \tag{4}$$

which attains its maximum $r_j(\mathbf{x}) = 1$ exactly when every gated literal is satisfied, and decreases by $|\tilde{v}_{ij}|$ for each violated literal. The AND-node output is obtained by binarizing this score,

$$a_j(\mathbf{x}) = \mathrm{Binarize}\big(r_j(\mathbf{x})\big) = \mathbb{I}_{r_j(\mathbf{x})>1-\epsilon}, \tag{5}$$

trained with a straight-through estimator (Bengio et al., 2013) that passes the incoming gradient where $r_j \geq 0$ and, once $r_j \geq 1$, only gradients that would lower it (a one-sided version of the gradient of $\mathrm{clip}(r_j, 0, 1)$). At evaluation, r_j is computed over the extracted literals only, i.e., those with $|\tilde{v}_{ij}| \geq \epsilon$. Since each violated extracted literal lowers r_j by at least ϵ, we have $a_j(\mathbf{x}) = 1$ if and only if all extracted literals are satisfied, so $a_j(\mathbf{x}) = \mathbb{I}_{\mathbf{x}\models s_j^{\wedge}}$ exactly, as required in Section 3.2.

3.4 DEEP CNLNET

The shallow CNLNet makes a single weighted-satisfiability decision over conjunctions of input literals. Given enough clauses it can represent any decision rule, but only as a flat set of conjunctions over input literals. Depth allows later clauses to reuse intermediate concepts. We require that (i) every hidden unit retains the weighted-satisfiability semantics in Section 3.2, so that the network remains a logic program at every depth, and (ii) the stack remains trainable through its binarizations. Stacking AND layers directly satisfies neither: hidden units lose weighted aggregation, and gradients through successive straight-through binarizations vanish in practice (App. B.1).

The Confidence Neural Logic Block. Let $\mathbf{h}^{(0)} = \mathbf{x}$ and $H_0 = N$. Block $l+1$ maps $\mathbf{h}^{(l)} \in \{0,1\}^{H_l}$ to $\mathbf{h}^{(l+1)} \in \{0,1\}^{H_{l+1}}$ by

$$\begin{aligned} &\mathbf{a}^{(l+1)} = \mathrm{Binarize}\big(\mathbf{r}^{(l+1)}(\mathbf{h}^{(l)})\big), \quad \mathbf{g}^{(l+1)} = \mathrm{BN}\big(\mathbf{U}^{(l+1)\top}\mathbf{a}^{(l+1)} + \mathbf{b}^{(l+1)}\big), \quad \mathbf{c}^{(l+1)} = \mathbb{I}_{\mathbf{g}^{(l+1)}>0}, \\ &\mathbf{h}^{(l+1)} = \mathbb{I}_{\mathbf{c}^{(l+1)}+\widetilde{\mathbf{h}}^{(l)}>0}, \end{aligned} \tag{6}$$

where $\mathbf{r}^{(l+1)}$ are the scores (4) of R_{l+1} AND-nodes with their own gated weights, Binarize is defined in (5), $\mathbf{U}^{(l+1)} \in \mathbb{R}^{R_{l+1}\times H_{l+1}}$ and $\mathbf{b}^{(l+1)}$ are signed confidence weights and biases, BN is batch normalization (Ioffe & Szegedy, 2015), and $\widetilde{\mathbf{h}}^{(l)}$ is the residual aligned to width H_{l+1}:

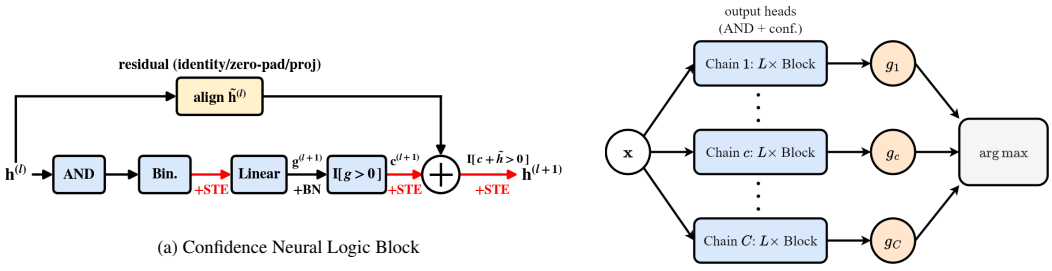
$$\widetilde{\mathbf{h}}^{(l)} = \begin{cases} \mathbf{h}^{(l)} & H_l = H_{l+1} \quad \text{(identity)}, \\ \big(\mathbf{h}^{(l)}, \mathbf{0}\big) & H_l < H_{l+1} \quad \text{(zero-padding)}, \\ \mathbf{P}^{(l)}\mathbf{h}^{(l)} & H_l > H_{l+1} \quad \text{(learned projection)}. \end{cases} \tag{7}$$

The three thresholds in (6) are exact indicators in the forward pass and are trained with straight-through estimators (Bengio et al., 2013): STE_1 for Binarize and STE_2 for $\mathbb{I}_{\mathbf{g}>0}$ back-propagate the gradient of $\mathrm{clip}(\cdot, -\kappa, \kappa)$-type windows, while the combining threshold, whose input $\mathbf{c} + \widetilde{\mathbf{h}}$ takes only the discrete levels $\{0, 1, 2\}$ for identity and zero-padding residuals, passes the gradient through unchanged (STE_3). Unless stated otherwise, all hidden blocks share a common width $H \geq N$, so the first residual is a zero-padding and all later residuals are identities; we resort to the learned projection only for very high-dimensional inputs.

Each hidden unit is a weighted-satisfiability decision. At inference, BN is a fixed affine map, which we fold into $\mathbf{U}$ and $\mathbf{b}$, so that $g_k = \sum_j u_{jk} a_j + b_k$.

Proposition 3.1 (Hidden units as weighted-SAT decisions)**.** *Fix a block and a hidden unit k.*

(i) Clause path. *Let c'_k be a Boolean variable and consider the weighted formulas $|u_{jk}| : c'_k \leftarrow s_j^{\wedge}$ if $u_{jk} > 0$, and $|u_{jk}| : \neg c'_k \leftarrow s_j^{\wedge}$ if $u_{jk} < 0$, for every clause j of the block, together with $|b_k| : c'_k \leftarrow \top$ if $b_k > 0$ and $|b_k| : \neg c'_k \leftarrow \top$ if $b_k < 0$. Then $c_k^{(l+1)} = 1$ if and only if the assignment $c'_k = \mathrm{True}$ has a strictly higher weighted-satisfiability score than $c'_k = \mathrm{False}$.*



(a) Confidence Neural Logic Block

(b) Class-Conditional Deep CNLNet

Figure 2: Deep CNLNet. **(a)** A Confidence Neural Logic Block: $\mathbf{c} = \mathbb{I}\big[\mathrm{BN}(\mathbf{U}^\top \mathrm{Binarize}(\mathbf{r}(\mathbf{h}^{(l)})) + \mathbf{b}) > 0\big]$ and $\mathbf{h}^{(l+1)} = \mathbb{I}\big[\mathbf{c} + \tilde{\mathbf{h}}^{(l)} > 0\big]$; the thresholds are trained with straight-through estimators. Each clause-path unit c_k is a weighted-satisfiability decision over the block's conjunctive clauses, and the residual acts as a default rule that keeps the input feature on (Proposition 3.1). **(b)** Class-conditional Deep CNLNet: each class owns a private chain of blocks, so no parameter receives gradient from more than one class.

(ii) Residual merge. *Let h'_k be a Boolean variable and consider the formulas $1 : h'_k \leftarrow c_k^{(l+1)}$ and, for the residual, $1 : h'_k \leftarrow h_k^{(l)}$ for an identity or zero-padding residual (no residual formula if coordinate k is padded), or $|P_{ki}| : h'_k \leftarrow h_i^{(l)}$ (respectively $\neg h'_k \leftarrow h_i^{(l)}$) for each i with $P_{ki} > 0$ (respectively < 0) for a learned projection. Then $h_k^{(l+1)} = 1$ if and only if $h'_k = \mathrm{True}$ has a strictly higher weighted-satisfiability score than $h'_k = \mathrm{False}$.*

The proof applies Theorem 3.1 with the two classes c'_k (respectively h'_k) and their negations, treating the bias as a clause with an empty body. For an identity or zero-padding residual, (ii) reads $h_k^{(l+1)} = c_k^{(l+1)} \vee h_k^{(l)}$: the residual acts as a default rule that keeps a feature on whenever it is on in the input, and the clauses can switch a feature on but never off. A depth-L network is therefore a layered weighted logic program: the atoms of each block are defined by two weighted-satisfiability decisions over the atoms of the layer below, and the output layer is the class-level program in Section 3.1. Substituting each layer into the next yields a flat program over $\mathbf{x}$; we therefore interpret deep models layer by layer instead, which keeps each layer's program small.

Trainability. The confidence weights are unbounded, so their pre-threshold values can drift outside the gradient window of the straight-through estimators, and the gradient reaching early blocks vanishes as depth grows. BN and the residual counteract this in complementary ways. BN keeps the pre-threshold values of each unit centered near the threshold, inside the active window. The residual adds an identity term to the block Jacobian: because the combining threshold passes gradients through unchanged, $\partial h_k^{(l+1)}/\partial \mathbf{h}^{(l)} = \partial c_k/\partial \mathbf{h}^{(l)} + \mathbf{e}_k$ for identity and zero-padding residuals, whatever the value of g_k. Gradients can thus reach earlier blocks even when the AND path is saturated. The learned projection forfeits this guarantee and discards input dimensions that later clauses can no longer access, which can make depth underperform the shallow model (Section 4.2); hence our default $H \geq N$.

Class-conditional chains. A multi-class model could route all classes through one shared stack of blocks, followed by C output heads. Each shared parameter then receives the sum of C class-specific gradient contributions.

Lemma 3.1 (Gradient interference in a shared stack)**.** *Let δ_c be the gradient contribution of class c to a shared parameter vector, with $\mathbb{E}\langle \delta_c, \delta_{c'} \rangle = 0$ for $c \neq c'$ and $\mathbb{E}\|\delta_c\|^2 = \sigma^2$. Then the total update $\delta = \sum_c \delta_c$ satisfies $\mathbb{E}\|\delta\|^2 = C\sigma^2$, so each class accounts for a fraction $1/\sqrt{C}$ of the root-mean-square update norm.*

Empirically, shared stacks lose substantial macro-F1 relative to the shallow model on benchmarks with many classes (Section 4.2). We therefore give each class c its own chain of blocks $\mathcal{B}_c^{(1)}, \ldots, \mathcal{B}_c^{(L)}$,

each defined by (6) (Figure 2b):

$$\mathbf{h}_c^{(0)} = \mathbf{x}, \qquad \mathbf{h}_c^{(l)} = \mathcal{B}_c^{(l)}\big(\mathbf{h}_c^{(l-1)}\big) \;\; (l = 1, \ldots, L), \qquad g(\mathbf{x})_c = \sum_j w_{jc}\, a_{jc}\big(\mathbf{h}_c^{(L)}\big), \tag{8}$$

where a_{jc} are the AND-node outputs of an output layer of the form in Section 3.1, specific to class c. Prediction remains $c^* = \arg\max_c g(\mathbf{x})_c$, and each parameter receives gradient from exactly one class. The equivalence in Section 3.2 still holds at the output layer: treating the final hidden atoms of all chains as input variables, a clause of chain c_0 keeps its formula for class c_0 and receives confidence 0 for every other class, so the Theorem 3.1 applies unchanged. The price is that the number of parameters grows linearly with C.

4 Experiment

We evaluate CNLNet on 18 tabular and image classification benchmarks — 13 datasets from the RRL benchmark suite (Wang et al., 2021), the UCI *car* dataset, and four large-scale binary tasks with up to $\sim$11M samples — spanning binary to 26-way problems, extreme class imbalance, and nearly five orders of magnitude in sample size. Every model, RRL included, is evaluated under an identical leak-free 5-fold protocol: train, validation, and test never overlap, and the held-out test fold is touched exactly once. Continuous features are discretized against 9 per-feature quantile thresholds, fixed across all datasets — a sweep on *magic* found no consistent gain from denser binarization. Dataset statistics, the full protocol, and per-dataset hyperparameters appear in App. C.

All variants are trained end-to-end by minimizing a task loss (cross-entropy, or focal loss under severe class imbalance, which we find more reliable than reweighting by class frequency) plus the expected-L_0 norm — averaged over every AND layer in the network (every block, every chain, and the output heads) — and an L1 penalty on confidence weights, optimized in alternating phases so clause and confidence sparsity do not fight each other early in training. Two choices are specific to depth: gate parameters are initialized at a negative value for every hidden block, so clauses start *empty* and vacuously satisfied by every input, avoiding the cold start of Section 3.4 (at the default initialization roughly 84% of gates start open, making early clauses too restrictive to fire); and, on the most extreme-imbalance tasks, where clauses can stop firing altogether and the OR-residual of (6) degenerates into a pass-through (the AND/BN path contributes nothing, so the block's added depth is wasted), we drop each block's residual skip $\widetilde{\mathbf{h}}^{(l)}$ independently per example during training (residual dropout), forcing the loss gradient back through the AND/BN path and reviving clauses that would otherwise stay dead.

4.1 Main Results

Comparison with interpretable and hybrid methods. Table 1 reports the performance of CNLNet per dataset (breakdown of shallow and deep versions in Appendix. C.6), all trained under a flat λ_{L_0}=0.1. Against the interpretable and hybrid methods, CNLNet has the best average rank (4.50, against 6.14 for NeuroLogic's MLP, 6.22 for SATNet, 7.31 for LR, 8.11 for RRL, and 8.33 for C4.5). It wins on 17 of 18 datasets against C4.5, 16 against RRL (by 6.0 points on average, losing only on *tic-tac-toe* and *credit-fraud*), 13 against SATNet, and 12 against LR. It is ahead of the best of these six baselines on 8 datasets, most clearly on *chess* (+6.3) and *car* (+4.2); where it trails, the gap is at most 3.6 points except on *forest-cover*, where C4.5 leads by 14.7. Its largest margins over RRL are on many-class problems (*chess* +40.6, *connect-4* +15.5). NeuroLogic's distilled rules lose to CNLNet on all 18 datasets, with MLP→Rules gaps above 50 points on *chess*, *letRecog*, *facebook*, *fashion*, and *forest-cover*, i.e., post-hoc extraction destroys fidelity to the MLP. Deep is the better variant on 12 datasets and Shallow on the other 6 (*adult*, *banknote*, *letRecog*, *dota2*, *facebook*, *credit-fraud*), by at most 2.6 points. In mean macro-F_1 over the 18 datasets, CNLNet is ahead of C4.5 by 4.6, RRL by 6.0, SATNet by 8.3, and LR by 9.1 points.

Comparison with black-box methods. Although it is fully interpretable, CNLNet ranks fourth overall (4.50), close to XGBoost (3.94) and comparible to LightGBM (4.17) and PLNN (4.42), and ahead of random forests (5.64) and SVMs (7.72). It achieves the best result among all 12 methods on six datasets (*adult*, *bank-marketing*, *car*, *chess*, *connect-4*, and *porto-seguro*), mostly datasets with categorical or combinatorial structure that conjunctive rules capture naturally. Black-box methods

remain ahead on datasets dominated by continuous features, most notably *forest-cover*, where random forests lead by 16.6 points, and *higgs*, *facebook*, and *credit-fraud* (2.8–3.2 points).

Dataset	**CNLNet**	Interpretable				Hybrid (NeuroLogic)		Black-box				
		RRL	C4.5	LR	SATNet	MLP	Rules	SVM	PLNN	RF	LGBM	XGB
adult	**79.17***	78.69	71.65	77.22	78.44	78.38	77.21	77.08	78.38	75.92	77.51	77.16
bank-marketing	**77.10***	74.36	69.61	72.25	75.43	72.45	52.15	72.34	75.71	67.98	75.59	76.09
banknote	99.56	98.82	99.34	**99.63**	98.82	99.12	98.75	99.56	98.82	99.48	99.26	99.56
car	**98.71***	90.16	94.48	82.75	90.92	75.71	67.77	83.56	93.03	90.62	98.49	98.03
chess	**86.54***	45.93	80.24	36.76	64.76	68.52	14.58	30.86	83.35	74.43	85.62	85.18
connect-4	**72.77***	57.23	61.78	57.52	70.71	68.62	48.79	56.25	72.53	62.97	68.89	70.99
letRecog	94.86*	87.78	87.81	85.26	91.87	91.76	21.45	83.12	95.64	96.39	**96.43**	95.79
magic	84.80	81.20	79.14	81.40	85.03	85.30	79.25	81.55	85.22	84.69	**85.59**	85.05
tic-tac-toe	97.02	98.03	91.56	97.58	96.64	73.76	66.14	98.13	97.44	98.37	**99.65**	98.73
wine	96.68	94.96	94.94	97.79	97.21	91.86	88.15	97.25	93.28	**98.89**	97.84	97.26
activity	96.74	95.31	93.56	98.78	49.13	98.78	87.93	98.84	98.68	97.96	99.10	**99.11**
dota2	59.43	53.72	51.78	**59.62**	59.24	59.17	48.65	59.59	59.47	57.10	59.06	58.36
facebook	86.92	85.51	80.77	89.15	13.03	89.81	30.89	89.04	**90.07**	87.52	85.69	88.24
fashion	84.75	76.75	72.48	79.53	84.28	83.76	27.67	79.13	85.31	83.76	86.16	**86.46**
Large-scale (>250K samples)												
porto-seguro	**52.94***	49.07	50.38	42.69	49.12	49.07	8.59	42.74	49.10	49.07	44.70	47.40
forest-cover	71.07	66.23	85.73	52.54	81.08	81.10	21.12	58.16	85.71	**87.63**	76.06	78.77
credit-fraud	90.40	91.45	87.43	58.71	90.59	91.84	69.24	58.28	91.64	92.55	92.45	**93.20**
higgs	71.94	68.55	66.02	69.37	**75.58**	75.01	59.03	69.35	75.18	74.75	72.96	74.18
Avg. Rank	4.50	8.11	8.33	7.31	6.22	6.14	11.50	7.72	4.42	5.64	4.17	**3.94**

Table 1: Test macro-F_1 (%, 5-fold mean) of CNLNet (best of Shallow/Deep) against interpretable, hybrid, and black-box baselines. For NeuroLogic, *MLP* is the underlying network before rule extraction and *Rules* is the distilled DNF rules as a standalone classifier. **Bold**: best per row; underline: second best. *Avg. Rank*: mean per-dataset rank over all 12 columns (ties split evenly). * indicates significant outperformance of CNLNet over interpretable models (t-test with $p < 0.05$).

4.2 Analysis

Program size (Appendix. D). Confidence pruning reduces CNLNet's programs without reducing test macro-F_1 below that of the unpruned program (the pruning depth is picked on the test fold, so sizes are an oracle estimate). The shallow model shrinks by 44–68% on *adult*, *bank-marketing*, *banknote*, *magic*, *tic-tac-toe*, and *wine* (e.g., *adult*: $198.6 \rightarrow 110.4$ clauses), but by only 4–11% on the high-C datasets with 1500–2000 clauses (*fashion*, *chess*, *connect-4*, *letRecog*); the deep model shrinks by 22–94% (e.g., *dota2*: $714.6 \rightarrow 46.4$). After pruning, shallow CNLNet is smaller than C4.5 on 15 of 18 datasets, often by orders of magnitude (*higgs*: 172.0 vs. 1.5×10^6). RRL yields the most compact rule sets on most datasets; shallow CNLNet undercuts it on *facebook*, *porto-seguro*, and *credit-fraud*, but is larger on high-C and high-dimensional datasets such as *chess*, *letRecog*, and *fashion*. The deep model is typically larger than the shallow one, since each class owns a private chain of blocks, but it is smaller after pruning on five datasets, e.g., *connect-4* (748.8 vs. 1347.6), *activity* (102.8 vs. 534.6), and *dota2* (46.4 vs. 138.2).

Inference time (Appendix. E). Shallow CNLNet is faster than RRL on every dataset, on both CPU and GPU (2–390× and 2–590×; e.g., *facebook*: 30.0 vs. 11,699.7 μs on CPU), since its clause matching reduces to two dense matrix multiplications; on GPU it is 35 to 22,500× faster than SATNet. The deep model is usually slower than the shallow one but still faster than RRL on CPU on 13 of 18 datasets. C4.5 is the fastest CPU method on 16 of 18 datasets, but shallow CNLNet on GPU beats it on 14 of 18 (all but *banknote*, *car*, *tic-tac-toe*, and *wine*).

BN and the residual matter when blocks are stacked (Appendix. F). With two hidden blocks, the full block (BN + residual) is best on three of four datasets. Without either component, training can fail outright: *tic-tac-toe* collapses to 39.52% with a std of only 0.08, a reproducible failure. On *wine*, neither component helps alone (BN 89.62, residual 94.63, vanilla 94.67), whereas their combination reaches 98.29. With a single hidden block, the four variants lie within about one point of each other on *car*, *tic-tac-toe*, and *adult* (on *wine*, BN alone is 2.3 points below Vanilla), consistent with the view that BN and the residual address a trainability problem that only arises when blocks are stacked.

Class-conditional chains avoid interference at high C (Appendix. F). On *chess* (C=18), a shared chain falls far below the shallow model (67.09 vs. 83.47), whereas class-conditional chains exceed it (86.54), as predicted by Lemma 3.1. On *car* (C=4), where interference is mild, the two are within noise (98.55 vs. 98.71) and both above Shallow (94.81). On *letRecog* (C=26), class-conditional chains improve on the shared chain (92.87 vs. 92.10) but neither deep variant reaches the shallow model (94.86), so removing interference is necessary but not always sufficient for depth to pay off.

Sharing granularity at depth 2 (Appendix. F). With two blocks, fully class-conditional chains are best on *car* and *chess* and on par with the share→class hybrid on *letRecog* (91.25 vs. 91.39); fully shared chains are near-worst everywhere and both hybrids are markedly less stable (std up to 14 points), so partial sharing brings no reliable benefit over fully class-conditional chains.

Depth helps on high-cardinality and large-scale data (Appendix. F). One hidden block improves over the shallow model on *chess* (+3.1), *activity* (+1.5), and *forest-cover* (+3.6), while on the saturated control *banknote* it does not help (99.56 → 97.57). A second block further improves *forest-cover* (73.15) but not *chess* (85.51), and it fails on *activity* (63.69), where the best epochs (767–6698 of 8100) rule out under-training. We attribute this to depth-1 hyperparameters copied unchanged at this dataset's high input dimension ($N \approx 5000$); depth 2 generally needed a doubled epoch budget.

Hidden width should exceed the input width (Appendix. G). On both datasets, accuracy increases monotonically from $H < N$ to $H = N$ to $H > N$, and only $H > N$ improves on the shallow model (*chess*: 86.54 vs. 83.47; *connect-4*: 72.77 vs. 70.82). A lossy projection residual ($H < N$) performs worst, falling 4–17 points below the shallow model. A lossless identity residual ($H = N$) is not sufficient either: it still trails the shallow model by 4.3 points on *chess* and 2.1 on *connect-4*. Depth thus needs a lossless residual and capacity beyond the N input literals, so we use $H > N$ except on *activity* and *facebook* ($N \approx 5000$).

Hyperparameter sensitivity (Appendix. H). Both models are robust to the L_0 coefficient: halving or doubling λ_{L_0} changes validation macro-F_1 by at most about 2 points on most datasets, and for the deep model on *chess* and *fashion* a stronger penalty even helps (+4.8 and +6.2 points from 0.5× to 2×). Capacity is the most influential knob, especially on the high-C datasets. For the shallow model, tripling the number of clauses gains 5.6 points on *chess* and 3.4 on *connect-4*. For the deep model, doubling the hidden width raises *chess* from 62.92 (at 0.5×) to 88.20 and *connect-4* from 65.12 to 74.57, in line with the width ablation above. Halving the learning rate changes little except for the shallow model on *activity* (94.77 → 74.86) and the deep model on the high-C datasets (*chess* −12.5, *letRecog* −3.9), suggesting under-training; the 4–6 point drops on *wine* and *tic-tac-toe* amount to a few samples of a small validation split, and variations of about one point are within its noise.

5 Conclusion

We introduced Confidence Neural Logic Networks (CNLNet), rule learners whose forward pass at inference is exactly weighted-satisfiability inference over an explicit set of confidence-weighted clauses. This correspondence gives the linear aggregation of conjunctive clauses a precise logical meaning, and it extends to depth: in the Confidence Neural Logic Block, every hidden unit is itself a weighted-satisfiability decision, so deep networks remain layered logic programs. Batch normalization and a residual default rule keep deep stacks trainable, and class-conditional chains remove the gradient interference that otherwise grows with the number of classes. Across 18 benchmarks with up to 11M samples, CNLNet achieves a better average rank than interpretable baselines (RRL, C4.5, SATNet, NeuroLogic) and is comparable to strong black-box models such as PLNN, LightGBM, and XGBoost, with Shallow inference faster than RRL and programs that pruning reduces without loss of accuracy.

Limitations and future work. RRL still yields more compact rule sets on most datasets, and LightGBM and XGBoost keep a slightly better average rank. Class-conditional chains increase the number of parameters linearly with the number of classes. Promising directions include sharing intermediate concepts across classes without reintroducing interference, and extending CNLNet to relational and first-order settings.

AI USE STATEMENT

Generative AI tools were used only for code debugging and language editing, including grammar correction, vocabulary refinement, and improving the clarity and readability of the manuscript. They were not used to generate research claims, design experiments, analyze experimental results, or make scientific conclusions. All AI-assisted edits were reviewed and revised by the authors to ensure technical accuracy and consistency with the actual experiments. The authors take full responsibility for the final content of the paper, including all text, claims, results, and artifacts.

ETHICS STATEMENT

This work uses only publicly available tabular and image classification benchmarks (UCI, Kaggle, and Fashion-MNIST); none involve human-subjects research or personally identifying data collected by the authors. We are not aware of fairness, privacy, dual-use, or other ethical concerns specific to this work beyond those already documented by the original dataset releases.

REPRODUCIBILITY STATEMENT

The complete experimental protocol — fold construction, feature discretization, and per-dataset hyperparameters for every result in this paper — is given in App. C, and every theoretical claim stated in Sections 3.2–3.4 is proved in full in Appendices A–B. Baseline training and evaluation protocols, including any necessary deviations from a baseline's original specification, are documented in Section C.4. Together with the anonymized code released as supplementary material, these allow the reported results to be reproduced; timing measurements depend on the hardware.

REFERENCES

Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, 18(234): 1–78, 2018.

Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss Markov random fields and probabilistic soft logic. *J. Mach. Learn. Res.*, 18(1):3846–3912, January 2017. ISSN 1532-4435.

Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Mateo Espinosa Zarlenga, Lucie Charlotte Magister, Alberto Tonda, Pietro Lio, Frederic Precioso, Mateja Jamnik, and Giuseppe Marra. Interpretable neural-symbolic concept reasoning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 1801–1825. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/barbiero23a.html.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013. URL http://arxiv.org/abs/1308.3432.

Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 793–802, 2018.

Gabriele Ciravegna, Francesco Giannini, Stefano Melacci, Marco Maggini, and Marco Gori. A constraint-based approach to learning and explanation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3658–3665, 2020.

Gabriele Ciravegna, Pietro Barbiero, Francesco Giannini, Marco Gori, Pietro Liò, Marco Maggini, and Stefano Melacci. Logic explained networks. *Artif. Intell.*, 314(C), January 2023. ISSN

0004-3702. doi: 10.1016/j.artint.2022.103822. URL https://doi.org/10.1016/j.artint.2022.103822.

William W. Cohen. Fast effective rule induction. In Armand Prieditis and Stuart Russell (eds.), *Machine Learning Proceedings 1995*, pp. 115–123. Morgan Kaufmann, San Francisco (CA), 1995. ISBN 978-1-55860-377-6. doi: https://doi.org/10.1016/B978-1-55860-377-6.50023-2. URL https://www.sciencedirect.com/science/article/pii/B9781558603776500232.

Sanjeeb Dash, Oktay Günlük, and Dennis Wei. Boolean decision rules via column generation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 4660–4670, Red Hook, NY, USA, 2018. Curran Associates Inc.

A. d'Avila Garcez and G. Zaverucha. The connectionist inductive learning and logic programming system. *Appl. Intelligence*, 11(1):59–77, 1999. ISSN 0924-669X. doi: 10.1023/A:1008328630915. URL http://dx.doi.org/10.1023/A:1008328630915.

Lucile Dierckx, Rosana Veroneze, and Siegfried Nijssen. RL-Net: Interpretable rule learning with neural networks. In *Advances in Knowledge Discovery and Data Mining: 27th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2023, Osaka, Japan, May 25–28, 2023, Proceedings, Part I*, pp. 95–107, Berlin, Heidelberg, 2023. Springer-Verlag. ISBN 978-3-031-33373-6. doi: 10.1007/978-3-031-33374-3_8. URL https://doi.org/10.1007/978-3-031-33374-3_8.

Michelangelo Diligenti, Marco Gori, Marco Maggini, and Leonardo Rigutini. Bridging logic and kernel machines. *Mach. Learn.*, 86(1):57–88, January 2012. ISSN 0885-6125. doi: 10.1007/s10994-011-5243-x. URL https://doi.org/10.1007/s10994-011-5243-x.

Michelangelo Diligenti, Marco Gori, and Claudio Saccà. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165, 2017. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2015.08.011. URL https://www.sciencedirect.com/science/article/pii/S0004370215001344. Combining Constraint Solving with Mining and Learning.

I. Donadello, L. Serafini, and A. S. d'Avila Garcez. Logic tensor networks for semantic image interpretation. In *IJCAI-17*, pp. 1596–1602, 2017.

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. In *ICLR*, 2019. URL https://openreview.net/pdf?id=B1xY-hRctX.

Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *J. Artif. Int. Res.*, 61(1):1–64, January 2018. ISSN 1076-9757.

Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. DL2: Training and querying neural networks with logic. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1931–1941. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/fischer19a.html.

Kun Gao, Katsumi Inoue, Yongzhi Cao, Hanpin Wang, and Feng Yang. Differentiable rule induction from raw sequence inputs. In *International Conference on Learning Representations*, 2025.

Chuqin Geng, Xiaojie Xu, Zhaoyue Wang, Ziyu Zhao, and Xujie Si. Decoding interpretable logic rules from neural networks. *CoRR*, abs/2501.08281, 2025. doi: 10.48550/ARXIV.2501.08281. URL https://doi.org/10.48550/arXiv.2501.08281.

Chuqin Geng, Anqi Xing, Li Zhang, Ziyu Zhao, Jingkai Xu, Yuhe Jiang, and Xujie Si. NEURO-LOGIC: From neural representations to interpretable logic rules. In *ICLR 2026 Workshop on Unifying Concept Representation Learning*, 2026.

Claire Glanois, Zhaohui Jiang, Xuening Feng, Paul Weng, Matthieu Zimmer, Dong Li, Wulong Liu, and Jianye Hao. Neuro-symbolic hierarchical rule induction. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7583–7615. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/glanois22a.html.

Ole-Christoffer Granmo. The tsetlin machine - A game theoretic bandit driven approach to optimal pattern recognition with propositional logic. *CoRR*, abs/1804.01508, 2018. URL http://arxiv.org/abs/1804.01508.

Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing. Harnessing deep neural networks with logic rules. In *ACL*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, 2015.

Liran Katzir, Gal Elidan, and Ran El-Yaniv. Net-DNF: Effective deep modeling of tabular data. In *International Conference on Learning Representations*, 2021. URL https://api.semanticscholar.org/CorpusID:233440557.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pp. 5338–5348, 2020.

Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 1675–1684, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939874. URL https://doi.org/10.1145/2939672.2939874.

Zhaoyu Li, Jinpei Guo, Yuhe Jiang, and Xujie Si. Learning reliable logical rules with SATNet. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023a. URL http://papers.nips.cc/paper_files/paper/2023/hash/2ff46d83d1dcc063e075058b29d55efe-Abstract-Conference.html.

Ziyang Li, Jiani Huang, and Mayur Naik. Scallop: A language for neurosymbolic programming. *Proc. ACM Program. Lang.*, 7(PLDI), June 2023b. doi: 10.1145/3591280. URL https://doi.org/10.1145/3591280.

Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 18878–18890. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/9d27fdf2477ffbff837d73ef7ae23db9-Paper.pdf.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through L_0 regularization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=H1Y8hhg0b.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4765–4774, 2017.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 3749–3759. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7632-deepproblog-neural-probabilistic-logic-programming.pdf.

Stephen Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13(3):245–286, Dec 1995.

Stephen Muggleton and Luc de Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19-20:629–679, 1994. ISSN 0743-1066. doi: https://doi.org/10.1016/0743-1066(94)90035-3. URL https://www.sciencedirect.com/science/article/pii/0743106694900353. Special Issue: Ten Years of Logic Programming.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16428–16446. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/navon22a.html.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Deep differentiable logic gate networks. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Felix Petersen, Hilde Kuehne, Christian Borgelt, Julian Welzel, and Stefano Ermon. Convolutional differentiable logic gate networks. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Litao Qiao, Weijia Wang, and Bill Lin. Learning accurate and interpretable decision rule sets from neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 4303–4311, 2021.

Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. RNNLogic: Learning logic rules for reasoning on knowledge graphs. In *International Conference on Learning Representations*, 2021.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*, 2018. URL https://api.semanticscholar.org/CorpusID:3366554.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Mach. Learn.*, 62(1–2):107–136, February 2006. ISSN 0885-6125. doi: 10.1007/s10994-006-5833-1. URL https://doi.org/10.1007/s10994-006-5833-1.

Ryan Riegel, Alexander G. Gray, Francois P. S. Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh K. Srivastava. Logical neural networks. *CoRR*, abs/2006.13155, 2020. URL https://arxiv.org/abs/2006.13155.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215, 2019. doi: 10.1038/s42256-019-0048-x.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. DRUM: End-to-end differentiable rule mining on knowledge graphs. In *Advances in Neural Information Processing Systems 32*, 2019.

Prithviraj Sen, Breno Carvalho, Ryan Riegel, and Alexander Gray. Neuro-symbolic inductive logic programming with logical neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8212–8220, 2022.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 525–536, Red Hook, NY, USA, 2018. Curran Associates Inc.

Luciano Serafini and Artur S. d'Avila Garcez. Learning and reasoning with logic tensor networks. In *AI*IA*, pp. 334–348, 2016. ISBN 978-3-319-49129-5. doi: 10.1007/978-3-319-49130-1_25. URL https://doi.org/10.1007/978-3-319-49130-1_25.

A. Srinivasan, S. H. Muggleton, R.D. King, and M.J.E. Sternberg. Mutagenesis: ILP experiments in a non-determinate biological domain. In *Proceedings of the 4th International Workshop on Inductive Logic Programming, volume 237 of GMD-Studien*, pp. 217–232, 1994.

G. Towell and J. Shavlik. The extraction of refined rules from knowledge-based neural networks. *Machine Learning*, 13:71–101, 1993.

Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6545–6554. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/wang19e.html.

Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. A Bayesian framework for learning rule sets for interpretable classification. *J. Mach. Learn. Res.*, 18 (1):2357–2393, January 2017. ISSN 1532-4435.

Zhuo Wang, Wei Zhang, Ning Liu, and Jianyong Wang. Transparent classification with multilayer logical perceptrons and random binarization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 6331–6339, 2020.

Zhuo Wang, Wei Zhang, Ning Liu, and Jianyong Wang. Scalable rule-based representation learning for interpretable classification. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Zhuo Wang, Wei Zhang, Ning Liu, and Jianyong Wang. Learning interpretable rules for scalable data representation and classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(2):1121–1133, 2024.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, July 2018. URL http://starai.cs.ucla.edu/papers/XuICML18.pdf.

Sascha Xu, Nils Philipp Walter, and Jilles Vreeken. Neural rule lists: Learning discretizations, rules, and order in one go. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 148617–148645. Curran Associates, Inc., 2025. doi: 10.52202/085713-4970. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/da813258fb4db548b0a6c34d79da8594-Paper-Conference.pdf.

Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2319–2328. Curran Associates, Inc., 2017a. URL http://papers.nips.cc/paper/6826-differentiable-learning-of-logical-rules-for-knowledge-base-reasoning.pdf.

Hongyu Yang, Cynthia Rudin, and Margo Seltzer. Scalable Bayesian rule lists. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3921–3930, 2017b.

Yang Yang, Wendi Ren, and Shuang Li. HyperLogic: Enhancing diversity and accuracy in rule learning with hypernets. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Chang Yue and Niraj K. Jha. Learning interpretable differentiable logic networks. *IEEE Transactions on Circuits and Systems for Artificial Intelligence*, 2024.

Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, Pietro Lio, and Mateja Jamnik. Concept embedding models: beyond the accuracy-explainability trade-off. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Wei Zhang, Yongxiang Liu, Zhuo Wang, and Jianyong Wang. Learning to binarize continuous features for neuro-rule networks. In Edith Elkind (ed.), *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pp. 4584–4592. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/510. URL https://doi.org/10.24963/ijcai.2023/510. Main Track.

A Appendix: Weighted-Satisfiability Equivalence

Proof of Theorem 3.1. Fix $\mathbf{x}$ and c. An implication $h_{jc'} \leftarrow s_j^{\wedge}$ is violated if and only if its body holds ($I_j = 1$) and its head is false under $\mathbf{y}(c)$. By the one-hot structure of $\mathbf{y}(c)$, the head $h_{jc'}$ is false exactly in the following cases:

- $h_{jc'} = y^{(c')}$ and $c' \neq c$;
- $h_{jc'} = \neg y^{(c')}$ and $c' = c$.

Let $\Gamma = \sum_{j,c'} \gamma_j^{(c')}$ denote the total confidence, and let $P_j = \sum_{c':h_{jc'}=y^{(c')}} \gamma_j^{(c')}$ denote the total confidence of the positive-head formulas associated with clause j. The confidence of the formulas of clause j violated under $\mathbf{y}(c)$ is

$$V_j(c) = I_j\Big(\sum_{\substack{c' \neq c \\ h_{jc'}=y^{(c')}}} \gamma_j^{(c')} \;+\; \mathbb{I}_{h_{jc}=\neg y^{(c)}}\, \gamma_j^{(c)} \Big).$$

If $h_{jc} = y^{(c)}$, the bracket equals $P_j - \gamma_j^{(c)} = P_j - w_{jc}$; if $h_{jc} = \neg y^{(c)}$, it equals $P_j + \gamma_j^{(c)} = P_j - w_{jc}$. In both cases $V_j(c) = I_j(P_j - w_{jc})$. Therefore

$$f(\mathbf{x}, \mathbf{y}(c)) = \Gamma - \sum_{j=1}^{R} V_j(c) = \underbrace{\Gamma - \sum_{j=1}^{R} I_j P_j}_{B(\mathbf{x})} \;+\; \sum_{j=1}^{R} w_{jc}\, I_j = B(\mathbf{x}) + g(\mathbf{x})_c.$$

Since $B(\mathbf{x})$ does not depend on c, it cancels in any difference between two classes, which yields the second statement. The ordering of the classes, and hence the $\arg\max$, is therefore identical under f and g. □

Bidirectional formulas. The result extends to formulas $\gamma_j^{(c)} : h_{jc} \leftrightarrow s_j^{\wedge}$ if the AND-nodes output bipolar values.

Proposition A.1. *Replace every implication in $\mathcal{P}$ by the corresponding equivalence $h_{jc} \leftrightarrow s_j^{\wedge}$, and let $\tilde{g}(\mathbf{x})_c = \sum_j w_{jc}\, a_j$ with $a_j = 2I_j - 1 \in \{-1, +1\}$. Then $f(\mathbf{x}, \mathbf{y}(c)) = \tilde{B}(\mathbf{x}) + \tilde{g}(\mathbf{x})_c$ for some $\tilde{B}(\mathbf{x})$ independent of c, so the conclusions of Theorem 3.1 hold with $\tilde{g}$ in place of g.*

Proof. Let $t_{jc'}(c) \in \{0, 1\}$ be the truth value of $h_{jc'}$ under $\mathbf{y}(c)$. The equivalence $\phi_{jc'}$ is satisfied if and only if $t_{jc'}(c) = I_j$, i.e.,

$$\mathbb{I}_{(\mathbf{x},\mathbf{y}(c))\models\phi_{jc'}} = t_{jc'}(c)\, I_j + \big(1 - t_{jc'}(c)\big)(1 - I_j) = (1 - I_j) + t_{jc'}(c)\, a_j.$$

Let $\Gamma_j = \sum_{c'} \gamma_j^{(c')}$ and $Q_j = \sum_{c':h_{jc'}=\neg y^{(c')}} \gamma_j^{(c')}$. A positive head is true only for $c' = c$, and a negated head is true for all $c' \neq c$, so $\sum_{c'} \gamma_j^{(c')} t_{jc'}(c) = Q_j + w_{jc}$ in both cases for h_{jc}. Hence

$$f(\mathbf{x}, \mathbf{y}(c)) = \underbrace{\sum_{j=1}^{R} \big[\Gamma_j(1 - I_j) + Q_j\, a_j\big]}_{\tilde{B}(\mathbf{x})} + \sum_{j=1}^{R} w_{jc}\, a_j,$$

which proves the claim. □

B Appendix: Proof of the Gradient-Interference Lemma

Proof of Lemma 3.1 (gradient interference). By pairwise uncorrelatedness and zero mean,

$$\mathbb{E}\|\delta\|^2 = \mathbb{E}\Big\| \sum_c \delta_c \Big\|^2 = \sum_c \mathbb{E}\|\delta_c\|^2 + \sum_{c \neq c'} \mathbb{E}\langle \delta_c, \delta_{c'} \rangle = C\sigma^2 + 0.$$

The signal carried by a single class has magnitude $\sqrt{\mathbb{E}\|\delta_c\|^2} = \sigma$, so the SNR is $\sigma/\sqrt{C\sigma^2} = 1/\sqrt{C}$. In the class-conditional model a parameter on chain c influences the output only through head c, so its gradient equals δ_c and the SNR is $\sigma/\sigma = 1$, independent of C. □

B.1 Diagnostic Measurements for the Optimization Obstacle

The gradient-vanishing obstacle of Section 3.4 can be measured directly on plain stacked blocks (no BN, no residual; `experiments/diagnostic.py`). On *car*, the fraction of hidden pre-activations inside Binarize's active gradient window falls from 62% at initialization to 4.5% after 400 epochs at depth one, and during the first epochs the gradient reaching the first AND layer is about $40\times$ (depth one) and $900\times$ (depth two) smaller than in the shallow model.

C Appendix: Experimental Details

C.1 Protocol

For every dataset we partition the rows into 5 stratified folds of roughly equal size and equal class proportions; each fold in turn serves once as the held-out test set, with the remaining four folds forming that split's training pool. Inside each training pool we further carve out a stratified 10% validation subset (again class-proportional), used *only* for checkpoint selection: training monitors validation macro-F_1 every epoch and keeps the best checkpoint, and the test fold is evaluated exactly once with that checkpoint. No test data touches training or model selection. RRL (Wang et al., 2021) is run through the same fold assignment and the same $90/10$ train/validation protocol so the comparison in Table 1 is split-for-split identical. We report macro-F_1 (mean$\pm$std over the 5 folds).

C.2 Feature discretization

Continuous features are binarized by comparison against per-feature empirical quantile thresholds (a one-hot "$\leq t$" encoding at each of 9 quantiles per feature, fixed across datasets; Section 4); categorical features are one-hot encoded. The resulting Boolean vector is the literal input $\mathbf{x} \in \{0,1\}^N$ of Section 3.1; N per dataset is reported in Table 3. The identical discretized tensors are handed to CNLNet and to the RRL clone.

C.3 Hyperparameters

All CNLNet models use the Adam optimizer and the two-phase L_0/L_1 objective of Section 4 with θ initialized negative for every hidden block (Section 4). Table 2 lists the per-dataset settings (number of clauses / clauses-per-hidden-block, hidden width $H \geq N$ except on *activity* and *facebook*, learning rate, L_0 and L_1 coefficients, batch size, epochs) for the Shallow and Deep configurations.

	Shallow						Deep (class-conditional, one hidden block)							
Dataset	clauses	lr	λ_{L_0}	λ_{L_1}	bs	epochs	out cl.	H	hid. cl.	lr	λ_{L_0}	λ_{L_1}	bs	epochs
adult	256	0.005	0.1	5e-4	512	1700	384	256	512	0.003	0.1	5e-4	4096	7200
bank-marketing	512	0.005	0.1	5e-4	512	2200	256	192	384	0.006	0.1	5e-4	4096	6100
banknote	256	0.005	0.1	1e-3	128	2500	128	64	128	0.006	0.1	1e-3	512	3100
car	128	0.01	0.1	1e-3	128	1500	100	64	200	0.01	0.1	1e-4	128	2400
chess	1500	0.005	0.1	1e-4	512	3500	384	128	256	0.006	0.1	1e-4	4096	8100
connect-4	1500	0.0025	0.1	1e-4	1024	4100	512	384	768	0.004	0.1	1e-4	4096	5900
letRecog	1500	0.005	0.1	1e-4	512	3800	640	256	512	0.006	0.1	1e-4	4096	9400
magic	512	0.005	0.1	5e-4	512	2100	256	192	384	0.006	0.1	5e-4	4096	6100
tic-tac-toe	128	0.008	0.1	1e-3	64	2500	96	64	128	0.0035	0.1	1e-3	128	3700
wine	64	0.005	0.1	1e-3	32	1000	128	128	256	0.004	0.1	1e-3	64	2400
activity	1500	0.003	0.1	1e-4	256	2700	1000	512	1500	0.003	0.1	1e-4	2048	7200
dota2	256	0.0015	0.1	1e-4	1024	1600	384	400	512	0.002	0.1	1e-4	1024	5400
facebook	512	0.0015	0.1	5e-4	256	2800	1000	512	1500	0.003	0.1	1e-4	2048	7200
fashion	2000	0.005	0.1	1e-5	400	3200	1000	800	1000	0.01	0.1	1e-5	4096	7900
porto-seguro	256	0.003	0.1	1e-4	2048	600	256	400	400	0.003	0.1	1e-4	2048	300
forest-cover	512	0.005	0.1	1e-4	4096	400	512	150	200	0.005	0.1	1e-4	4096	300
credit-fraud	256	0.003	0.1	1e-4	1024	600	256	300	300	0.003	0.1	1e-4	1024	300
higgs	256	0.003	0.1	1e-4	8192	80	256	256	256	0.003	0.1	1e-4	8192	80

Table 2: Per-dataset hyperparameters that produce Table 1 (Adam; Deep uses θ-init -3.0 and one hidden block per chain, "out cl." = output clauses, H = hidden width, "hid. cl." = clauses per hidden block). Dataset-specific extras: the *banknote* Shallow run uses 20 quantile thresholds; *tic-tac-toe* Shallow adds λ_{bin}=0.01; the *porto-seguro* and *credit-fraud* Shallow runs use focal loss (γ=2), and the *porto-seguro* Deep run uses focal loss (γ=2) and residual dropout 0.3 (Section 4).

C.4 BASELINES

C4.5, LR, SVM, RF, LightGBM, and XGBoost are trained and evaluated under the same leak-free 5-fold protocol and feature representation as CNLNet, so results are directly comparable fold-for-fold; RRL, NeuroLogic, SATNet, and PLNN follow the same protocol. PLNN and NeuroLogic cover all 18 datasets (*higgs*'s 8.8M-row training set required capping the grounding-tree depth and subsampling its fit set for NeuroLogic to complete in tractable time); SATNet covers all 18; on *higgs*, whose 8.8M-row training set makes its per-step solver cost dominant, it is trained with a reduced budget (batch size 8192, half the default clause count m, and a 9-epoch cap versus the default 50; ≈ 11.6 hours per fold), and its best epoch is the last on every fold, so the reported figure likely understates what the default budget would reach. On two datasets with $N{\sim}5000$ literals (*activity*, *facebook*), SATNet's fixed-size differentiable-solver kernel forces a far smaller clause budget than elsewhere in the roster, and the resulting training is highly unstable rather than merely less accurate: *activity*'s 5 folds span 13.51–98.85 test-F_1 (49.13 ± 34.13, only one fold converging to a healthy optimum) and *facebook*'s 5 folds collapse consistently low (13.03 ± 1.90). Both are reported as measured in Table 1, since this instability appears to be a genuine property of SATNet's optimizer at this literal count rather than a one-off failure. On *porto-seguro* SATNet likewise uses a reduced budget (m=184 clauses for n=368 variables, versus $m \approx 2n$ on the other datasets not listed above).

NeuroLogic (Geng et al., 2026) is, unlike every other baseline here, a *post-hoc decompositional* rule extractor rather than an end-to-end rule learner: it distills a global DNF rule set out of an already-trained black-box network instead of training a rule-structured model directly. We train a generic MLP (64→32, ReLU) on the same binarized features as CNLNet, then apply NeuroLogic's own algorithm — per-class Neural Activation Pattern predicates at the last hidden layer, selected by a linear threshold search over a balanced purity score, combined into per-class DNF clauses via positive-predicate distillation with minimal-clause (subset) pruning, and grounded back to the input space with a decision tree standing in for the original paper's C5.0 (every input feature here is already binarized, the same substitution used for our C4.5 baseline) — to obtain a standalone classifier. NeuroLogic's published evaluation covers only a Transformer/sentiment-analysis case study, not the small-scale tabular benchmark (its own Appendix B: XOR/MAGIC/METABRIC vs. C5.0/ECLAIRE/CGXPLAIN) that is the relevant comparison point for this roster, so we reimplement its core algorithmic primitives (the purity search and positive/subset-pruned distillation) and make independent choices where the original specification is underdetermined, including the multiclass decision rule used to turn per-class DNFs into one label per test input.

C.5 DATASET STATISTICS

The benchmark mixes small UCI-scale problems (e.g. *banknote*, *tic-tac-toe*, *wine*), medium tabular sets (*adult*, *magic*, *bank-marketing*, *connect-4*), high-cardinality multi-class tasks that stress the class-conditional design (*chess*, C=18; *letRecog*, C=26), high-dimensional inputs that exercise the residual-alignment cases of Section 3.4 (*activity*, *facebook*, *dota2*), an image task (*fashion*), and four large-scale binary tasks that test scalability and extreme imbalance: *forest-cover* (581K rows) and *higgs* (11M rows) probe throughput, while *credit-fraud* (minority rate 0.17%, i.e. 578:1) and *porto-seguro* (3.6%, 26:1) probe the focal-loss / residual-dropout remedies of Section 4. Per-dataset statistics are listed in Table 3.

Dataset	#samples	N (literals)	C	imbalance	source
adult	30,162	128	2	3.0:1	UCI
bank-marketing	45,211	86	2	7.5:1	UCI
banknote	1,372	36	2	1.2:1	UCI
car	1,728	21	4	18.6:1	UCI
chess	28,056	40	18	168.6:1	UCI
connect-4	67,557	126	3	6.9:1	UCI
letRecog	20,000	99	26	1.1:1	UCI
magic	19,020	90	2	1.8:1	UCI
tic-tac-toe	958	27	2	1.9:1	UCI
wine	178	117	3	1.5:1	UCI
activity	10,299	4,994	6	1.4:1	UCI
dota2	102,944	394	2	1.1:1	UCI
facebook	22,470	4,714	4	2.1:1	UCI
fashion	70,000	784	10	1.0:1	Zalando
porto-seguro	595,212	366	2	26.4:1	Kaggle
forest-cover	581,012	134	7	103.1:1	UCI
credit-fraud	284,807	270	2	577.9:1	Kaggle
higgs	11,000,000	224	2	1.1:1	UCI

Table 3: Dataset statistics: number of samples, number of Boolean literals N after discretization (9 quantile thresholds per continuous feature, matching the deep model's input; the *banknote* Shallow run uses 20 thresholds, giving 80 literals), number of classes C, class-imbalance ratio (majority:minority), and source. The 13 standard datasets follow the RRL suite (Wang et al., 2021); the four large-scale binary tasks are Kaggle/UCI additions (*porto-seguro*, *forest-cover*, *credit-fraud*, *higgs*); *car* is a further UCI addition.

C.6 FULL PER-BASELINE RESULTS

This section is for full results in Table 4.

D APPENDIX: PROGRAM SIZE

Table 5 compares program size head-to-head with RRL, SATNet, and C4.5. For Shallow, Deep, and RRL we report R_{agg} under the same criterion: the number of distinct rules after merging rows with an *identical literal pattern* — clauses (CNLNet) or binarized conjunction/disjunction connectivity rows (RRL) that select exactly the same literals fire identically on *any* input, not just coincidentally on the held-out test set, so this is a property of the learned program itself rather than of which finite sample happens to be evaluated (unlike merging by test-set activation agreement, which conflates structurally distinct rules that merely agree on this sample and understates true program size). For RRL, the all-zero (no-literal) connectivity pattern — which a large fraction of RRL's logical-layer nodes collapse to in practice — is excluded from the count, since a rule that selects no literals is a constant, not an interpretable condition. (A CNLNet clause with no literals is likewise an always-true clause, i.e. an extra bias; it is merged with its duplicates like any other pattern, so it changes the count

Dataset	CNLNet		Interpretable				Hybrid (NeuroLogic)		Black-box				
	Shallow	Deep	RRL	C4.5	LR	SATNet	MLP	Rules	SVM	PLNN	RF	LGBM	XGB
adult	**79.17**	78.43	78.69	71.65	77.22	78.44	78.38	77.21	77.08	78.38	75.92	77.51	77.16
bank-marketing	76.85	**77.10**	74.36	69.61	72.25	75.43	72.45	52.15	72.34	75.71	67.98	75.59	76.09
banknote	99.56	97.57	98.82	99.34	**99.63**	98.82	99.12	98.75	99.56	98.82	99.48	99.26	99.56
car	94.81	**98.71**	90.16	94.48	82.75	90.92	75.71	67.77	83.56	93.03	90.62	98.49	98.03
chess	83.47	**86.54**	45.93	80.24	36.76	64.76	68.52	14.58	30.86	83.35	74.43	85.62	85.18
connect-4	70.82	**72.77**	57.23	61.78	57.52	70.71	68.62	48.79	56.25	72.53	62.97	68.89	70.99
letRecog	94.86	92.87	87.78	87.81	85.26	91.87	91.76	21.45	83.12	95.64	96.39	**96.43**	95.79
magic	84.41	84.80	81.20	79.14	81.40	85.03	85.30	79.25	81.55	85.22	84.69	**85.59**	85.05
tic-tac-toe	96.87	97.02	98.03	91.56	97.58	96.64	73.76	66.14	98.13	97.44	98.37	**99.65**	98.73
wine	96.14	96.68	94.96	94.94	97.79	97.21	91.86	88.15	97.25	93.28	**98.89**	97.84	97.26
activity	95.22	96.74	95.31	93.56	98.78	49.13	98.78	87.93	98.84	98.68	97.96	99.10	**99.11**
dota2	59.43	59.36	53.72	51.78	**59.62**	59.24	59.17	48.65	59.59	59.47	57.10	59.06	58.36
facebook	86.92	86.80	85.51	80.77	89.15	13.03	89.81	30.89	89.04	**90.07**	87.52	85.69	88.24
fashion	84.08	84.75	76.75	72.48	79.53	84.28	83.76	27.67	79.13	85.31	83.76	86.16	**86.46**
Large-scale (>250K samples)													
porto-seguro	52.12	**52.94**	49.07	50.38	42.69	49.12	49.07	8.59	42.74	49.10	49.07	44.70	47.40
forest-cover	67.46	71.07	66.23	85.73	52.54	81.08	81.10	21.12	58.16	85.71	**87.63**	76.06	78.77
credit-fraud	90.40	87.83	91.45	87.43	58.71	90.59	91.84	69.24	58.28	91.64	92.55	92.45	**93.20**
higgs	71.30	71.94	68.55	66.02	69.37	**75.58**	75.01	59.03	69.35	75.18	74.75	72.96	74.18
Avg. Rank	5.67	5.44	8.83	9.22	7.97	6.78	6.75	12.44	8.39	4.75	6.08	4.50	**4.17**

Table 4: Full per-variant breakdown behind Table 1: CNLNet **Shallow** and **Deep** separately (5-fold mean macro-F_1 %), followed by interpretable, hybrid (NeuroLogic) and black-box baselines. Bold marks the best of the row and underline the second best. *Avg. Rank* averages each method's per-dataset rank (1 = best that row, ties split evenly; Shallow and Deep ranked as separate methods here) over the datasets where it has a value.

by at most one for Shallow and one per class chain for Deep.) All figures are 5-fold mean$\pm$std on the held-out test set.

Dataset	Shallow		Deep		RRL	SATNet	C4.5
	Before	After	Before	After			
adult	198.6 ± 15.6	110.4 ± 24.3	636.6 ± 137.0	390.6 ± 187.3	12.2 ± 1.0	260.0 ± 0.0	5500.0 ± 70.1
bank-marketing	418.4 ± 32.5	218.0 ± 55.2	503.6 ± 3.4	390.6 ± 65.8	10.2 ± 0.7	176.0 ± 0.0	4361.6 ± 65.1
banknote	254.6 ± 2.3	80.6 ± 34.3	229.2 ± 26.3	59.4 ± 60.0	19.4 ± 2.5	76.0 ± 0.0	27.6 ± 1.0
car	91.6 ± 14.5	77.2 ± 20.8	351.4 ± 19.7	104.8 ± 26.6	21.8 ± 1.0	48.0 ± 0.0	111.2 ± 3.2
chess	1482.4 ± 4.8	1396.4 ± 63.2	6609.6 ± 37.9	5117.6 ± 678.1	77.4 ± 1.0	116.0 ± 0.0	7199.2 ± 41.2
connect-4	1491.6 ± 1.9	1347.6 ± 205.0	1379.6 ± 20.4	748.8 ± 315.6	31.2 ± 1.7	256.0 ± 0.0	13959.8 ± 269.1
letRecog	1496.8 ± 3.7	1330.6 ± 59.0	13390.4 ± 189.2	3858.8 ± 988.8	181.2 ± 1.5	248.0 ± 0.0	2065.4 ± 18.7
magic	441.0 ± 8.7	183.6 ± 69.8	508.8 ± 4.4	212.8 ± 61.6	13.2 ± 0.4	184.0 ± 0.0	2741.2 ± 19.7
tic-tac-toe	72.8 ± 4.7	36.8 ± 12.2	192.0 ± 0.0	93.0 ± 22.6	19.4 ± 0.5	56.0 ± 0.0	80.4 ± 7.7
wine	64.0 ± 0.0	24.4 ± 9.7	325.0 ± 20.2	119.4 ± 76.3	10.2 ± 2.2	240.0 ± 0.0	10.2 ± 0.4
activity	646.8 ± 37.5	534.6 ± 56.3	189.4 ± 68.9	102.8 ± 56.0	47.8 ± 3.5	1752.0 ± 0.0	282.4 ± 9.6
dota2	199.6 ± 4.5	138.2 ± 31.6	714.6 ± 36.8	46.4 ± 26.3	2.2 ± 0.7	792.0 ± 0.0	21279.0 ± 236.0
facebook	85.4 ± 9.8	50.4 ± 12.8	831.2 ± 875.4	248.6 ± 161.0	52.4 ± 7.6	1652.0 ± 0.0	2834.2 ± 18.4
fashion	2000.0 ± 0.0	1918.8 ± 91.5	9853.8 ± 30.5	4032.4 ± 472.3	54.2 ± 1.2	1588.0 ± 0.0	8105.4 ± 21.6
porto-seguro	127.2 ± 3.5	94.0 ± 23.1	245.2 ± 76.5	74.4 ± 42.3	127.8 ± 0.4	184.0 ± 0.0	33527.2 ± 1487.8
forest-cover	496.8 ± 4.0	394.4 ± 20.9	3424.8 ± 38.0	1883.6 ± 423.5	128.0 ± 0.0	280.0 ± 0.0	53949.4 ± 331.1
credit-fraud	14.2 ± 1.2	11.8 ± 2.6	184.4 ± 14.0	55.4 ± 21.4	63.6 ± 0.8	544.0 ± 0.0	292.8 ± 15.9
higgs	218.8 ± 3.5	172.0 ± 22.1	491.8 ± 9.4	337.0 ± 107.3	127.0 ± 0.9	224.0 ± 0.0	1535650.0 ± 15585.1

Table 5: Program size (number of distinct clauses/rules, 5-fold mean$\pm$std) for Shallow CNLNet, Deep CNLNet, RRL, SATNet, and C4.5. *Before* is R_{agg} (Section D); *After* is the smallest program on the confidence-pruning sweep (formulas removed weakest first) whose test macro-F_1 is at least that of the un-pruned program (the level is chosen on the test fold, i.e., an oracle size); the output bias is part of every program. SATNet's column is its number of MaxSAT clauses m, fixed by the architecture (hence ± 0.0) and not reduced by pruning; each clause is a dense real-valued constraint over all $n{+}1{+}\text{aux}$ variables, so this is an upper bound on the size of its learned program and is not directly comparable to sparse conjunctive rules. C4.5's rule count is its number of decision-tree leaves.

E Appendix: Inference Time

Table 6 reports wall-clock inference time (microseconds per test-set sample, 5-fold mean$\pm$std, minimum of 5 repeated timing runs per fold to reduce measurement noise), using each system's *rule-based* inference path – the same hard clause/tree evaluation Table 5 measures the size of, not a continuous relaxation. CNLNet Shallow's clause matching (Section D) is reformulated as two matrix

multiplications – an XOR-mismatch count against the ternary clause matrix, $X \cdot \mathrm{mask} + \mathrm{colsum}(B) - 2(X \cdot B)$, firing where the count is zero – rather than an elementwise broadcast-compare, so it is BLAS-backed like Deep's own forward pass and the comparison between them is not an artifact of one implementation being vectorized and the other not. CNLNet Shallow/Deep and RRL are timed on both CPU and GPU; C4.5 (a decision tree) has no GPU implementation, its own limitation rather than one equalized away; SATNet is GPU-only by construction (its differentiable-solver kernel) and is reconstructed at each fold's already-trained architecture shape (n, m, aux) with fresh random weights, since its forward-pass cost depends only on this shape, not on the learned weight values, so no retraining is needed to time it. Because all of this is dense, shape-dependent arithmetic whose cost does not depend on the input values, CNLNet Shallow/Deep and SATNet are timed on a correctly-shaped random input rather than the real (discretized) test fold, capped at $50{,}000$ samples – for the largest datasets this avoids re-discretizing the full pool (*higgs*'s 11M rows) solely to obtain a timing number, at no cost to the measurement itself.

Dataset	Shallow		Deep		RRL		SATNet (GPU)	C4.5 (CPU)
	CPU	GPU	CPU	GPU	CPU	GPU		
adult	1.98 ± 0.06	0.05 ± 0.00	7.88 ± 0.75	5.35 ± 0.66	37.40 ± 0.66	1.17 ± 0.00	229.38 ± 7.85	0.30 ± 0.01
bank-marketing	5.81 ± 0.90	0.05 ± 0.00	6.14 ± 0.45	4.70 ± 1.09	30.91 ± 0.35	0.97 ± 0.00	163.34 ± 3.46	0.19 ± 0.01
banknote	2.63 ± 0.13	1.22 ± 0.02	12.91 ± 0.25	12.02 ± 0.12	4.87 ± 0.03	3.42 ± 0.02	104.14 ± 2.82	0.62 ± 0.03
car	0.94 ± 0.02	0.97 ± 0.07	18.28 ± 0.86	18.18 ± 0.14	1.92 ± 0.03	2.00 ± 0.05	95.72 ± 3.08	0.48 ± 0.02
chess	14.49 ± 0.26	0.20 ± 0.00	61.84 ± 7.45	38.04 ± 2.09	49.80 ± 0.33	1.48 ± 0.00	154.34 ± 4.38	0.21 ± 0.01
connect-4	40.37 ± 1.22	0.25 ± 0.00	17.95 ± 1.05	8.30 ± 2.05	117.23 ± 3.90	3.53 ± 0.00	243.10 ± 7.48	0.26 ± 0.00
letRecog	16.96 ± 0.67	0.21 ± 0.00	135.38 ± 18.29	84.95 ± 6.08	412.52 ± 23.21	13.37 ± 0.02	279.03 ± 7.85	0.31 ± 0.01
magic	3.20 ± 0.02	0.09 ± 0.01	7.42 ± 0.78	5.21 ± 0.73	39.50 ± 1.35	1.16 ± 0.00	169.24 ± 4.43	0.26 ± 0.00
tic-tac-toe	0.82 ± 0.01	1.47 ± 0.10	17.15 ± 1.12	15.90 ± 0.19	3.55 ± 0.03	3.41 ± 0.02	93.67 ± 3.82	0.84 ± 0.02
wine	2.56 ± 0.17	8.45 ± 0.53	172.11 ± 2.18	160.18 ± 2.40	42.59 ± 5.31	33.12 ± 0.52	291.99 ± 5.18	4.72 ± 0.21
activity	67.78 ± 1.44	2.57 ± 0.01	456.23 ± 40.54	45.22 ± 4.01	6946.62 ± 135.59	248.29 ± 2.64	16752.53 ± 351.95	24.52 ± 0.35
dota2	3.62 ± 0.02	0.06 ± 0.00	10.85 ± 0.91	5.60 ± 0.99	96.47 ± 1.26	2.88 ± 0.00	1354.92 ± 36.84	1.34 ± 0.16
facebook	30.01 ± 0.38	0.76 ± 0.00	265.72 ± 20.99	26.81 ± 1.62	11699.70 ± 135.57	447.91 ± 0.09	16193.00 ± 381.25	7.83 ± 0.68
fashion	100.52 ± 12.53	0.74 ± 0.00	730.70 ± 124.99	107.87 ± 24.83	13492.19 ± 457.51	65.27 ± 0.01	8085.07 ± 149.98	12.96 ± 2.85
porto-seguro	5.99 ± 0.09	0.06 ± 0.00	8.24 ± 0.70	4.05 ± 0.19	218.27 ± 130.50	3.70 ± 0.00	264.62 ± 3.29	1.13 ± 0.17
forest-cover	9.69 ± 0.28	0.09 ± 0.00	22.32 ± 1.80	14.95 ± 0.24	40.34 ± 1.50	1.30 ± 0.00	253.61 ± 4.51	0.46 ± 0.01
credit-fraud	4.89 ± 0.08	0.05 ± 0.00	6.32 ± 0.54	3.90 ± 0.36	52.56 ± 0.87	1.64 ± 0.00	622.82 ± 18.25	0.57 ± 0.10
higgs	6.83 ± 0.28	0.05 ± 0.00	7.00 ± 0.32	2.77 ± 0.31	39.67 ± 1.36	1.47 ± 0.01	375.85 ± 14.73	1.31 ± 0.03

Table 6: Inference time (microseconds per test sample, 5-fold mean±std, min of 5 repeats), using each system's rule-based (hard) inference path. CNLNet Shallow/Deep and RRL are timed on CPU and GPU; SATNet is GPU-only (its differentiable-solver kernel); C4.5 is CPU-only (no GPU decision-tree implementation).

F Appendix: Ablations

We ablate four design choices, each isolating one claim from Sections 3.4–3.4. All numbers are test macro-F_1 (%, 5-fold mean±std) under the protocol of Section 4.

Two aspects of the protocol matter for interpreting this ablation. First, when the residual is enabled, the $\mathbb{I}[\mathbf{c}^{(l+1)} + \widetilde{\mathbf{h}}^{(l)} > 0]$ combination in (6) thresholds the *sum* of the clause path and the skip connection, whose value only ever takes the discrete levels $\{0, 1, 2\}$ (both paths off, exactly one on, or both on); the gradient-active window for this combine step is sized to that discrete range rather than to the wider continuous-input window used for the clause path alone, which would needlessly zero the gradient whenever both paths already agree (measured dead-zone rate on *car*, i.e. the paths-agree case: 26.0% for Residual, 28.4% for Full). Second, checkpoint selection tracks validation macro-F_1 with ties broken by validation loss, so that once val F_1 reaches its ceiling — which happens early on the small, easy datasets in this table — the selected checkpoint keeps advancing to the most confident epoch on that plateau rather than stopping at the first one to reach it (concretely, on one *tic-tac-toe* fold this is the difference between an epoch-1568 checkpoint at test F_1=91.1% and an epoch-3196 checkpoint, within the same run, at 98.5–100%).

*Full is the best of the four arms on three of the four two-block rows (*car*,* tic-tac-toe*,* wine*).* On *car* it beats the next-best arm, BN, by half a point (98.55 vs. 98.02). On *tic-tac-toe* the effect is dramatic: Vanilla collapses to 39.52% with the tightest std in the table (± 0.08, a fully reproducible failure, not noise), while BN, Residual, and Full all recover to 95–99%, Full both highest and most stable. *wine* makes the "jointly, not alone" point most starkly: BN alone is actively *worse* than Vanilla (89.62 vs. 94.67, also the noisiest arm at ± 5.84), Residual alone is statistically indistinguishable from Vanilla (94.63), and only the combination reaches 98.29. This matches Section 3.4: stacking two blocks

Dataset	Vanilla	+BN	+Residual	Full (BN+Res.)
One hidden block (production depth, Table 1/10)				
car	$98.34_{\pm 1.80}$	$98.13_{\pm 1.45}$	$98.29_{\pm 2.01}$	$\mathbf{98.71}_{\pm 1.17}$
tic-tac-toe	$\mathbf{97.34}_{\pm 1.68}$	$97.11_{\pm 0.95}$	$96.88_{\pm 0.89}$	$97.02_{\pm 0.97}$
wine	$\mathbf{97.34}_{\pm 0.16}$	$95.05_{\pm 1.20}$	$97.05_{\pm 2.64}$	$96.68_{\pm 2.11}$
adult	$78.21_{\pm 0.76}$	$78.14_{\pm 0.21}$	$77.98_{\pm 0.62}$	$\mathbf{78.43}_{\pm 0.58}$
Two hidden blocks				
car	$95.98_{\pm 1.71}$	$98.02_{\pm 1.73}$	$97.27_{\pm 2.62}$	$\mathbf{98.55}_{\pm 1.71}$
tic-tac-toe	$39.52_{\pm 0.08}$	$95.76_{\pm 0.92}$	$96.66_{\pm 1.31}$	$\mathbf{98.95}_{\pm 0.68}$
wine	$94.67_{\pm 1.83}$	$89.62_{\pm 5.84}$	$94.63_{\pm 2.46}$	$\mathbf{98.29}_{\pm 1.40}$
adult	$77.40_{\pm 0.84}$	$78.73_{\pm 0.71}$	$\mathbf{78.77}_{\pm 0.43}$	$78.48_{\pm 0.66}$

Table 7: **Block-component ablation.** Removing BatchNorm and/or the OR-residual from (6) in isolation and combined, one and two stacked hidden blocks per chain — the latter is the regime in which the two components matter. Chosen datasets are low-C so the effect is not confounded with the interference of Table 8. Bold marks the best of the four arms per row (the one-hidden-block Full arms reuse the main-table 5-fold Deep run, Table 4).

compounds STE saturation into a trainability failure, which on *tic-tac-toe* BN or the residual alone largely repairs, while on *wine* only their combination helps.

The one-hidden-block rows are the control this predicts against: with no compounding to fix, BN and the residual cost little and help little. On *tic-tac-toe* and *wine*, Vanilla is fractionally *ahead* of Full (97.34 vs. 97.02; 97.34 vs. 96.68); on *car* and *adult*, Full is fractionally ahead instead (98.71 vs. 98.34; 78.43 vs. 78.21) — but the arms land within about a point of each other on every one-block row except *wine*, where BN alone is 2.3 points lower, consistent with ordinary run-to-run noise rather than a second real effect. *adult*'s two-block row is the one exception to "Full wins": Vanilla is the worst arm (77.40), but Residual alone, not Full, is narrowly best (78.77 vs. 78.48) — BN and the residual each individually recover most of Vanilla's loss here, so combining them adds nothing further, unlike *wine*, where neither helped alone.

Dataset	C	Shallow	Deep (shared chain)	Deep (class-cond.)
car	4	$94.81_{\pm 5.50}$	$98.55_{\pm 1.75}$	$\mathbf{98.71}_{\pm 1.17}$
chess	18	$83.47_{\pm 0.94}$	$67.09_{\pm 0.72}$	$\mathbf{86.54}_{\pm 0.87}$
letRecog	26	$\mathbf{94.86}_{\pm 0.32}$	$92.10_{\pm 0.40}$	$92.87_{\pm 0.40}$

Table 8: **Sharing ablation** (Lemma 3.1). A single shared chain should degrade relative to shallow as C grows, while the class-conditional composition (Section 3.4) should recover and exceed shallow.

Table 8 contrasts shallow, shared-chain, and class-conditional depth at three points of C. On *chess* (C=18), the shared chain collapses well below even the shallow baseline (67.09 vs. 83.47), while class-conditional depth avoids the failure and beats Shallow outright (86.54) — the clearest confirmation in this roster of Lemma 3.1: at high C, forcing every class through one chain actively hurts, and separating the chains, not depth itself, is what fixes it. *car* (C=4, the low-cardinality control) shows no separation between the two deep variants (98.55 vs. 98.71, well within one standard deviation) and both are above Shallow (94.81); this is consistent with the interference argument, since at C=4 the interference predicted by Lemma 3.1 is mild, so there is little for the split to fix. *letRecog* (C=26, the highest-cardinality point here) shows a milder version of the *chess* ordering: the shared chain trails class-conditional by under a point (92.10 vs. 92.87) rather than collapsing outright, and neither yet beats Shallow (94.86) — so separating the chains helps a little at this cardinality but is not alone sufficient.

Table 9 extends this to two stacked hidden blocks per chain, at the same three points of C, with two additional *hybrid* arms that mix sharing granularity within a single model: *share→class* ties block 1's weights across classes and lets block 2 diverge per class, while *class→share* diverges immediately at block 1 and then applies a single weight-tied block 2 independently to each of the K already-different chains (not degenerate, since the K inputs to block 2 differ once block 1 has diverged). Both pure

arms (*share-share*, *class-class*) are the two homogeneous special cases of the same general sharing-granularity architecture used for the hybrids, so all four columns are architecturally comparable, not confounded by a different underlying model per arm.

Dataset	C	share-share	class-class	share→class	class→share
car	4	$93.67_{\pm 6.90}$	$\mathbf{98.40}_{\pm 1.47}$	$88.13_{\pm 10.05}$	$96.87_{\pm 2.44}$
chess	18	$56.84_{\pm 2.62}$	$\mathbf{77.00}_{\pm 8.10}$	$64.18_{\pm 13.93}$	$54.98_{\pm 8.05}$
letRecog	26	$88.47_{\pm 0.84}$	$91.25_{\pm 2.36}$	$\mathbf{91.39}_{\pm 0.52}$	$84.52_{\pm 11.24}$

Table 9: **Sharing ablation, depth 2** (extends Table 8 with two hidden blocks per chain and two hybrid share/class-conditional arms).

Dataset	Depth 0 (shallow)	Depth 1	Depth 2
banknote	$\mathbf{99.56}_{\pm 0.28}$	$97.57_{\pm 1.05}$	$98.38_{\pm 0.44}$
chess	$83.47_{\pm 0.94}$	$\mathbf{86.54}_{\pm 0.87}$	$85.51_{\pm 1.19}$
activity	$95.22_{\pm 0.46}$	$\mathbf{96.74}_{\pm 0.48}$	$63.69_{\pm 6.87}$
forest-cover	$67.46_{\pm 0.61}$	$71.07_{\pm 2.61}$	$\mathbf{73.15}_{\pm 0.39}$

Table 10: **Depth ablation.** Effect of stacking 0, 1, 2 hidden blocks per chain on macro-F_1. *banknote* is a saturated control (depth should not help); *chess*/*activity*/*forest-cover* are the high-cardinality and large-scale regimes where depth is expected to pay off.

Table 10 tests whether depth helps: it should help where reusable intermediate concepts exist, and should not help an already-saturated task. *banknote* is the saturated control, and the depth-1 result matches the prediction: depth does not help ($99.56 \rightarrow 97.57$, a mild regression consistent with adding unneeded capacity rather than a failure). *chess*, *activity*, and *forest-cover* are the opposite regime, and all three confirm the benefit at depth 1 ($83.47 \rightarrow 86.54$, $95.22 \rightarrow 96.74$, $67.46 \rightarrow 71.07$).

Depth 2 required a larger epoch budget than depth 1 to train to convergence: with the depth-1 budget copied unchanged, *banknote*, *chess*, and *forest-cover* all selected their checkpoint near the end of training on every fold, and *chess*/*forest-cover* scored *below* their own depth-1 run despite twice the parameters — a sign of an under-trained rather than a genuinely worse model, so all three were re-run with a doubled budget. Under the doubled budget, checkpoints are selected well before the ceiling on every fold, and results diverge by dataset: *banknote*'s depth-2 score ($98.38_{\pm 0.44}$) still does not beat the shallow baseline, confirming the saturated control holds at both depths; *chess* recovers to $85.51_{\pm 1.19}$, above its depth-0 baseline (83.47) but still short of depth-1 (86.54), unlike the two-block *car*/*tic-tac-toe*/*wine* rows of Table 7 where the full block wins outright — whether this reflects a genuine ceiling on stacking two class-conditional blocks, or further under-parameterizing of the depth-2 configuration (otherwise copied unchanged from depth-1), is not yet resolved; and *forest-cover* extends its depth-1 gain further, to $73.15_{\pm 0.39}$, above both shallower baselines, with a still-larger budget plausibly pushing it higher yet. *activity* instead collapses at depth 2 ($63.69_{\pm 6.87}$, well below both shallower baselines): per-fold best-epoch values (767–6698 of the 8100-epoch budget) rule out under-training, so this is a genuine convergence to a poor optimum under a depth-2 configuration — doubled hidden width copied unchanged from depth 1 with no re-parameterizing of `reg1`/`lr` at this dataset's much larger $N \approx 5000$ — rather than an epoch-budget artifact like the other three datasets' first attempt.

G Residual-alignment and Sensitivity Analysis

Beyond the four primary ablations above, this one further probes the secondary design choices of Sections 3.4 and 4; it reports test macro-F_1 (%, 5-fold mean±std).

Table 11 confirms the predicted monotonic ordering on both datasets: $H < N$ (lossy) $< H = N$ (exact, still constrains capacity) $< H > N$ (the production anchor's zero-pad regime). On *chess* (N=40), only $H > N$ beats the shallow baseline (86.54 vs. 83.47); both $H < N$ (66.58) and, more strikingly, the exactly-lossless $H = N$ (79.18) fall short of it despite $H = N$ having no information loss in the residual — capacity beyond the literal count evidently matters on its own, not

Dataset	N	Shallow	Deep, $H < N$ (lossy proj.)	Deep, $H = N$	Deep, $H > N$
chess	40	$83.47_{\pm 0.94}$	$66.58_{\pm 0.58}$	$79.18_{\pm 0.97}$	$\mathbf{86.54}_{\pm 0.87}$
connect-4	126	$70.82_{\pm 0.55}$	$66.96_{\pm 0.83}$	$68.69_{\pm 0.30}$	$\mathbf{72.77}_{\pm 0.17}$

Table 11: **Residual-alignment ablation** (Section 3.4). Hidden width H relative to the number of literals N. When $H < N$ the residual is a lossy learned projection and depth is expected to underperform even the shallow baseline; the lossless cases $H \geq N$ (identity / zero-pad) should let depth help.

just losslessness. *connect-4* (N=126) shows the same pattern: $H < N$ (66.96) and $H = N$ (68.69) both underperform shallow (70.82), and only $H > N$ (72.77) recovers the depth advantage. Together the two datasets rule out "any lossless residual is enough" in favor of the stronger requirement this paper's anchors already follow: H should exceed N with headroom, not merely match it.

H Appendix: Sensitivity Table Results

This appendix shows the results for different values of hyper-parameters.

	λ_{L_0}			clauses				lr	
Dataset	0.5×	1×	2×	0.5×	1×	2×	3×	1×	0.5×
banknote	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
car	100.00	100.00	100.00	98.45	100.00	100.00	100.00	100.00	100.00
tic-tac-toe	100.00	100.00	100.00	97.15	100.00	100.00	100.00	100.00	95.68
wine	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	93.73
magic	85.46	85.15	85.71	84.90	85.15	86.54	86.24	85.15	84.79
adult	79.55	79.46	79.26	79.05	79.46	79.24	79.19	79.46	78.76
bank-marketing	78.95	77.83	78.98	78.04	77.83	78.87	78.67	77.83	78.27
letRecog	94.09	94.36	94.35	92.89	94.36	95.16	95.59	94.36	94.60
chess	81.67	79.80	79.16	75.15	79.80	83.67	85.35	79.80	79.65
connect-4	70.79	69.58	69.84	67.10	69.58	72.22	72.94	69.58	67.92
dota2	59.82	59.90	59.83	60.01	59.90	59.90	59.63	59.90	60.43
fashion	85.26	85.30	85.09	84.92	85.30	85.30	–	85.30	85.84
activity	92.61	94.77	95.50	94.75	94.77	95.60	94.54	94.77	74.86
facebook	86.70	87.39	86.93	86.62	87.39	86.34	88.34	87.39	88.60

Table 12: **Hyperparameter sensitivity, Shallow.** Validation macro-F_1 (%) as each knob is scaled around its anchor value (1×) with the other two held fixed (one-at-a-time; 14-dataset roster, single split). A flat row indicates the model is insensitive to that knob; the L_0 coefficient λ_{L_0} trades program sparsity against accuracy, over-/under-provisioning the clause count tests capacity headroom, and lr tests optimization stability. *fashion*'s 3×-clauses cell is missing (that setting duplicates another cell in the grid). Notably *activity* is sharply lr-sensitive (94.77→74.86 at 0.5×), while *banknote* and *car* are saturated (at most one cell below 100) and *tic-tac-toe* and *wine* drop by 3–6 points only in isolated cells (0.5× clauses or lr), i.e., a few validation samples on these small datasets. Each cell is a single train/validation split (not the 5-fold protocol used elsewhere), one knob scaled at a time around its Table 2 anchor value with the other two held fixed.

I Appendix: Extracted Rule Examples

To make the compactness figures of Section D concrete, this section shows actual rules read off each system's fold-0 model on *car* (C=4, N=21 one-hot literals over {buying, maint, doors, persons, lug_boot, safety}), for the *vgood* class – the rarest of the four and the one where the three systems' induced logic differs most visibly. All clauses below are copied verbatim from the trained models (Shallow/Deep: sign of the masked AND-layer weight, Section 3.3; RRL: its own `rule_print` utility), not hand-simplified.

Dataset	λ_{L_0} 0.5×	λ_{L_0} 1×	λ_{L_0} 2×	hidden width H 0.5×	hidden width H 1×	hidden width H 1.5×	hidden width H 2×	lr 1×	lr 0.5×
banknote	98.17	98.17	99.08	100.00	98.17	99.08	99.08	98.17	97.25
car	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
tic-tac-toe	97.15	97.15	97.15	98.56	97.15	98.56	98.56	97.15	97.15
wine	100.00	100.00	100.00	100.00	100.00	93.73	100.00	100.00	93.94
magic	84.59	85.06	85.46	85.24	85.06	85.90	85.83	85.06	84.34
adult	79.56	79.55	79.51	78.69	79.55	79.36	79.42	79.55	79.27
bank-marketing	78.44	78.33	78.69	77.13	78.33	78.91	77.63	78.33	78.81
letRecog	93.31	93.29	93.48	88.61	93.29	93.99	94.16	93.29	89.37
chess	77.25	80.97	82.03	62.92	80.97	85.06	88.20	80.97	68.52
connect-4	71.93	72.30	72.33	65.12	72.30	72.15	74.57	72.30	69.27
dota2	59.90	60.24	59.99	59.82	60.24	59.84	59.85	60.24	59.69
fashion	79.30	83.41	85.52	83.37	83.41	–	–	83.41	79.65
activity	97.48	97.58	97.36	96.59	97.58	96.84	96.99	97.58	97.46
facebook	90.22	87.96	89.30	89.36	87.96	87.97	88.60	87.96	85.89

Table 13: **Hyperparameter sensitivity, Deep** (class-conditional, one hidden block). Same protocol as Table 12 with the clause-count knob replaced by hidden width H (Section 3.4). *car* is fully saturated (100.00 at every knob); *adult* and *bank-marketing* are both flat and insensitive to all three knobs, mirroring their Shallow rows. *connect-4* and especially *chess* ($62.92 \rightarrow 88.20$ from $0.5\times$ to $2\times$ hidden width) are markedly hidden-width-sensitive, consistent with high-C/lossy-residual regimes rewarding extra capacity (Table 11, Table 8). *dota2* is notably insensitive to λ_{L_0} (59.90–60.24 across 0.5–$2\times$) but shows a mild peak at $1\times$ hidden width, unlike *chess*'s monotonic capacity-hungry pattern.

CNLNet Shallow. After merging duplicate clauses (R_{agg}=91.6 on average, Table 5), the strongest evidence *for* and *against vgood* are both single conjunctive clauses over the raw literals, aggregated exactly as in Eq. (1):

- w=+5.234: `NOT maint_vhigh AND NOT persons_2 AND NOT lug_boot_small AND NOT safety_low`
- w=−12.295: `NOT buying_low AND NOT buying_med AND NOT maint_med` (i.e. buying price is *high* or *vhigh* and maintenance is not *med*)

Negative evidence dominates in magnitude: *vgood* is a narrow, exclusion-defined class, so a single disqualifying condition (an expensive-to-buy car is not *vgood*) carries more weight than any one positive combination can confirm on its own.

CNLNet Deep. The same class's chain has one hidden block (H=64) feeding an output block, exactly the composition of Section 3.4: block-1's 200 clauses are conjunctions over the same 21 literals as Shallow, and its linear scorer turns their truth values into 64 Boolean hidden features $h_0, \ldots, h_{63}$ – each itself a confidence-weighted aggregation of the form Eq. (1), one level deeper (H=64 $\geq$ N=21 makes the zero-pad residual lossless here, Section 3.4). The chain's output bias is +2.119 (an always-true formula in favor of *vgood*). One compact output-layer clause:

- w=−0.640: `IF` h_{36} `AND` h_{37} (evidence against *vgood*)

Substituting each hidden literal's own dominant block-1 clause makes this concrete – both h_{36} (w=7.045) and h_{37} (w=8.265) are driven mainly by the *same* underlying pattern, `NOT buying_low AND buying_med AND maint_high AND NOT maint_low AND NOT maint_med AND safety_high AND NOT safety_low` (a mid-priced, high-maintenance, high-safety car) – so here depth has learned two hidden units that partially duplicate one sub-concept rather than two independent ones, a concrete instance of the redundancy that merging and pruning (Section D) are designed to remove.

RRL. The same class's row of its rule dump (config from Section C, matching the checkpoint behind Table 1) includes:

- w=−1.0404: `safety_med`

- w=-0.7767: `buying_vhigh OR maint_vhigh OR safety_low`
- w=$+0.4374$: `persons_4 OR persons_more`

RRL's disjunction layer lets a single rule combine literals with `OR` directly; CNLNet's clauses are purely conjunctive, so the same disjunctive evidence is instead spread across several independently-weighted clauses in the aggregation of Eq. (1) – a difference in how each system spends its compactness budget (Table 5).

Full Logic Program (CNLNet Shallow, *car*) For completeness, the entire fold-0 Shallow program behind the examples above – all $R_{\text{agg}} = 81$ aggregated clauses (Section D) $\times$ 4 classes, i.e. every formula of Eq. (1) with its learned confidence $\gamma_j^{(c)}$, plus the four per-class output biases, which are formulas with an empty body ($\top$) – is listed below (328 formulas), sorted by $\gamma_j^{(c)}$ descending.

16.853466 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=low)
12.295085 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=med)
11.512652 : ¬(evaluation=vgood) ← (doors=2) ∧ ¬ (doors=4) ∧ ¬ (luggage boot=big)
10.777546 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=low)
9.065355 : (evaluation=unacc) ← (persons=2) ∧ ¬ (persons=4) ∧ ¬ (persons=more)
8.913157 : (evaluation=unacc) ← ¬ (safety=high) ∧ (safety=low) ∧ ¬ (safety=med)
8.670044 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=med)
8.335237 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (buying price=med)
8.223734 : ¬(evaluation=good) ← ¬ (luggage boot=big) ∧ (luggage boot=small) ∧ (safety=med)
6.816593 : ¬(evaluation=acc) ← (persons=2) ∧ ¬ (persons=4) ∧ ¬ (persons=more)
6.752473 : ¬(evaluation=acc) ← ¬ (safety=high) ∧ (safety=low) ∧ ¬ (safety=med)
6.510755 : ¬(evaluation=vgood) ← ¬ (persons=4) ∧ (persons=more) ∧ ¬ (luggage boot=med) ∧ (luggage boot=small)
6.461476 : ¬(evaluation=vgood) ← ¬ (safety=high) ∧ ¬ (safety=low)
6.192116 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (buying price=med)
5.983249 : ¬(evaluation=vgood) ← ¬ (buying price=med) ∧ (maintenance cost=vhigh)
5.476174 : ¬(evaluation=good) ← (buying price=low) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
5.234063 : (evaluation=vgood) ← ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=low)
5.230113 : ¬(evaluation=good) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
4.739934 : ¬(evaluation=vgood) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ (safety=high)
4.631388 : ¬(evaluation=vgood) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ (safety=high) ∧ ¬ (safety=low) ∧ ¬ (safety=med)
4.535906 : ¬(evaluation=vgood) ← ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
4.450823 : ¬(evaluation=vgood) ← ¬ (luggage boot=big) ∧ (luggage boot=small) ∧ (safety=med)
4.222991 : ¬(evaluation=vgood) ← ⊤
4.042840 : ¬(evaluation=good) ← (doors=2) ∧ ¬ (doors=4) ∧ ¬ (luggage boot=big)
4.033994 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
3.929332 : ¬(evaluation=vgood) ← ¬ (safety=high)
3.839239 : ¬(evaluation=good) ← (persons=2) ∧ ¬ (persons=4) ∧ ¬ (persons=more)
3.772784 : (evaluation=unacc) ← ¬ (buying price=med) ∧ (maintenance cost=vhigh)
3.768114 : ¬(evaluation=vgood) ← (persons=2) ∧ ¬ (persons=4) ∧ ¬ (persons=more)
3.762901 : ¬(evaluation=good) ← ¬ (buying price=med) ∧ (maintenance cost=vhigh)
3.722001 : ¬(evaluation=good) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (luggage boot=big) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=high)
3.674210 : (evaluation=unacc) ← (doors=2) ∧ ¬ (doors=4) ∧ ¬ (luggage boot=big)
3.672808 : ¬(evaluation=unacc) ← ¬ (buying price=high) ∧ ¬ (maintenance cost=high) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
3.512978 : (evaluation=unacc) ← ¬ (luggage boot=big) ∧ (luggage boot=small) ∧ (safety=med)
3.384717 : (evaluation=vgood) ← ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
3.341902 : ¬(evaluation=unacc) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (luggage boot=small) ∧ (safety=high) ∧ ¬ (safety=med)
3.254599 : ¬(evaluation=unacc) ← ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=low)
3.241191 : ¬(evaluation=vgood) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (luggage boot=big)
3.169707 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=vhigh) ∧ (safety=high)
3.116714 : (evaluation=unacc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (luggage boot=big) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=high)
3.098030 : ¬(evaluation=vgood) ← ¬ (buying price=vhigh) ∧ ¬ (safety=high)
3.045002 : ¬(evaluation=unacc) ← ¬ (maintenance

cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
2.995535 : ¬(evaluation=good) ← ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
2.835137 : (evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=low)
2.676141 : ¬(evaluation=good) ← ¬ (persons=4) ∧ (persons=more) ∧ ¬ (luggage boot=med) ∧ (luggage boot=small)
2.607783 : ¬(evaluation=vgood) ← (persons=4) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
2.559490 : (evaluation=acc) ← ⊤
2.553962 : ¬(evaluation=vgood) ← ¬ (luggage boot=big) ∧ (luggage boot=small) ∧ (safety=high) ∧ ¬ (safety=med)
2.516632 : ¬(evaluation=unacc) ← ¬ (buying price=high) ∧ ¬ (maintenance cost=high) ∧ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
2.480747 : ¬(evaluation=vgood) ← (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
2.471131 : (evaluation=unacc) ← ¬ (buying price=low) ∧ (buying price=vhigh) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
2.446298 : ¬(evaluation=good) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
2.389076 : (evaluation=unacc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
2.339326 : ¬(evaluation=vgood) ← ¬ (buying price=high) ∧ ¬ (safety=high)
2.175100 : ¬(evaluation=good) ← ¬ (safety=high) ∧ (safety=low) ∧ ¬ (safety=med)
2.166763 : ¬(evaluation=vgood) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
2.141630 : ¬(evaluation=unacc) ← ¬ (persons=2) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ (safety=high)
2.135510 : (evaluation=acc) ← ¬ (buying price=low) ∧ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ (maintenance cost=med) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
2.129198 : (evaluation=unacc) ← ¬ (persons=4) ∧ (persons=more) ∧ ¬ (luggage boot=med) ∧ (luggage boot=small)
2.128194 : ¬(evaluation=vgood) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (luggage boot=big) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=high)
2.127051 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ (buying price=vhigh) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
2.117242 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ (buying price=vhigh) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
2.085766 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med) ∧ (safety=high)
2.051554 : ¬(evaluation=unacc) ← ¬ (buying price=low) ∧ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ (maintenance cost=med) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
2.037550 : ¬(evaluation=vgood) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ (safety=high)
2.006906 : ¬(evaluation=unacc) ← ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=low)
1.954478 : ¬(evaluation=vgood) ← ¬ (luggage boot=big) ∧ ¬ (safety=high)
1.882244 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=vhigh) ∧ (safety=high)
1.873325 : ¬(evaluation=acc) ← ¬ (buying price=med) ∧ (maintenance cost=vhigh)
1.853562 : ¬(evaluation=acc) ← ¬ (buying price=low) ∧ (buying price=vhigh) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
1.835779 : (evaluation=vgood) ← ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=low)
1.811284 : ¬(evaluation=unacc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ (safety=high)
1.810710 : (evaluation=vgood) ← ¬ (buying price=high) ∧ ¬ (maintenance cost=high) ∧ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
1.784654 : (evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=vhigh) ∧ (safety=high)
1.784091 : (evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=med)
1.687768 : (evaluation=vgood) ← ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small) ∧ (safety=high) ∧ ¬ (safety=low)
1.674534 : ¬(evaluation=good) ← (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
1.672503 : (evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (buying price=med)
1.648044 : ¬(evaluation=unacc) ← ¬ (maintenance cost=vhigh) ∧ ¬ (doors=2) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
1.634603 : ¬(evaluation=acc) ← ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=low)
1.614127 : (evaluation=good) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ (safety=high) ∧ ¬ (safety=low) ∧ ¬ (safety=med)
1.609939 : ¬(evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med) ∧ (safety=high)
1.607684 : ¬(evaluation=acc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (luggage boot=big) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=high)
1.588089 : (evaluation=vgood) ← ¬ (buying price=high) ∧ ¬ (maintenance cost=high) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
1.583187 : ¬(evaluation=acc) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (luggage boot=small) ∧ (safety=high) ∧ ¬ (safety=med)
1.577674 : ¬(evaluation=good) ← ¬ (persons=2) ∧ ¬ (persons=4) ∧ ¬ (luggage boot=med)
1.565251 : ¬(evaluation=unacc) ← ¬ (buying

price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=vhigh) ∧ (safety=high)
1.553685 : ¬(evaluation=unacc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ (safety=high) ∧ ¬ (safety=low) ∧ ¬ (safety=med)
1.541572 : (evaluation=good) ← ¬ (buying price=high) ∧ ¬ (maintenance cost=high) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
1.511886 : ¬(evaluation=unacc) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small)
1.486204 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
1.472434 : (evaluation=good) ← ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
1.444686 : (evaluation=vgood) ← ¬ (maintenance cost=vhigh) ∧ ¬ (doors=2) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
1.438331 : ¬(evaluation=unacc) ← (buying price=low) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
1.432602 : ¬(evaluation=unacc) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=high) ∧ ¬ (safety=low) ∧ (safety=med)
1.429101 : ¬(evaluation=good) ← (persons=4) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
1.426772 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ (buying price=med) ∧ ¬ (maintenance cost=med)
1.407995 : (evaluation=vgood) ← ¬ (maintenance cost=high) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
1.401582 : ¬(evaluation=unacc) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
1.392518 : ¬(evaluation=unacc) ← ¬ (maintenance cost=high) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
1.382064 : ¬(evaluation=good) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (luggage boot=big)
1.381099 : ¬(evaluation=vgood) ← (buying price=high) ∧ ¬ (buying price=low) ∧ ¬ (maintenance cost=vhigh)
1.376157 : ¬(evaluation=vgood) ← (persons=4) ∧ ¬ (safety=high) ∧ ¬ (safety=low) ∧ ¬ (safety=med)
1.374200 : ¬(evaluation=vgood) ← ¬ (safety=high) ∧ (safety=low) ∧ (safety=med)
1.359393 : ¬(evaluation=vgood) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ (safety=high) ∧ ¬ (safety=low)
1.299306 : (evaluation=good) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ (safety=high)
1.283728 : ¬(evaluation=vgood) ← (buying price=low) ∧ ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ (safety=high)
1.278768 : ¬(evaluation=acc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ (safety=high)
1.264694 : ¬(evaluation=acc) ← ¬ (luggage boot=big) ∧ (luggage boot=small) ∧ (safety=med)
1.251575 : (evaluation=good) ← ¬ (buying price=high) ∧ ¬ (maintenance cost=high) ∧ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
1.230649 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
1.223466 : ¬(evaluation=vgood) ← ¬ (persons=2) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ (safety=high)
1.208832 : ¬(evaluation=unacc) ← ¬ (maintenance cost=vhigh) ∧ ¬ (doors=2) ∧ ¬ (safety=low)
1.207165 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ (persons=4) ∧ ¬ (persons=more)
1.199581 : ¬(evaluation=vgood) ← ¬ (doors=5more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
1.190492 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=vhigh)
1.188211 : (evaluation=good) ← ¬ (maintenance cost=high) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
1.174668 : ¬(evaluation=good) ← ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small) ∧ (safety=high) ∧ ¬ (safety=low)
1.161185 : (evaluation=vgood) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small)
1.144206 : ¬(evaluation=good) ← ¬ (safety=high) ∧ ¬ (safety=low)
1.141926 : ¬(evaluation=good) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (luggage boot=small) ∧ (safety=high) ∧ ¬ (safety=med)
1.118151 : ¬(evaluation=good) ← ¬ (doors=5more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
1.105665 : ¬(evaluation=good) ← ¬ (luggage boot=big) ∧ ¬ (safety=high)
1.100953 : (evaluation=acc) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
1.086216 : (evaluation=vgood) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (luggage boot=small) ∧ (safety=high) ∧ ¬ (safety=med)
1.079204 : ¬(evaluation=acc) ← ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
1.068981 : (evaluation=vgood) ← ¬ (maintenance cost=vhigh) ∧ ¬ (doors=2) ∧ ¬ (safety=low)
1.064789 : ¬(evaluation=unacc) ← ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small) ∧ (safety=high) ∧ ¬ (safety=low)
1.060806 : (evaluation=good) ← ¬ (persons=2) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ (safety=high)
1.033715 : (evaluation=acc) ← (buying price=high) ∧ ¬ (buying price=low) ∧ ¬ (maintenance cost=vhigh)
1.029971 : ¬(evaluation=vgood) ← ¬ (maintenance cost=med) ∧ ¬ (doors=4) ∧ ¬ (safety=low)
1.012151 : ¬(evaluation=acc) ← ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.979968 : (evaluation=vgood) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.978831 : (evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
0.970010 : ¬(evaluation=acc) ← ¬ (doors=4) ∧ ¬

(doors=5more) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.961424 : ¬(evaluation=unacc) ← (buying price=high) ∧ ¬ (buying price=low) ∧ ¬ (maintenance cost=vhigh)
0.957891 : ¬(evaluation=unacc) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=high) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.956969 : (evaluation=unacc) ← ¬ (persons=2) ∧ ¬ (persons=4) ∧ ¬ (luggage boot=med)
0.953760 : (evaluation=acc) ← (buying price=low) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
0.944334 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (doors=4) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.942424 : (evaluation=good) ← ¬ (maintenance cost=vhigh) ∧ ¬ (doors=2) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.935776 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (doors=4) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.930504 : ¬(evaluation=vgood) ← ¬ (safety=high) ∧ (safety=low)
0.927911 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (safety=high)
0.927634 : ¬(evaluation=vgood) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
0.922602 : (evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med) ∧ (safety=high)
0.895690 : (evaluation=good) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=high) ∧ ¬ (safety=low) ∧ (safety=med)
0.886805 : (evaluation=good) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=high) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.866032 : ¬(evaluation=unacc) ← ¬ (buying price=high) ∧ ¬ (maintenance cost=high)
0.865497 : ¬(evaluation=vgood) ← ¬ (persons=2) ∧ ¬ (persons=4) ∧ ¬ (luggage boot=med)
0.862629 : (evaluation=good) ← ¬ (maintenance cost=vhigh) ∧ ¬ (doors=2) ∧ ¬ (safety=low)
0.842602 : ¬(evaluation=unacc) ← ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med) ∧ ¬ (persons=4) ∧ (safety=high)
0.839181 : (evaluation=unacc) ← ⊤
0.834874 : ¬(evaluation=good) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ (safety=high) ∧ ¬ (safety=low)
0.832284 : ¬(evaluation=good) ← (luggage boot=small) ∧ ¬ (safety=high) ∧ ¬ (safety=low) ∧ ¬ (safety=med)
0.828528 : (evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=low)
0.825665 : ¬(evaluation=acc) ← ¬ (maintenance cost=high) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.824976 : ¬(evaluation=good) ← (safety=high) ∧ (safety=low) ∧ ¬ (safety=med)
0.823795 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ (persons=4) ∧ ¬ (persons=more)
0.818175 : ¬(evaluation=vgood) ← (persons=2) ∧ (persons=4) ∧ ¬ (persons=more)
0.812765 : (evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
0.812420 : ¬(evaluation=acc) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small)
0.807228 : ¬(evaluation=unacc) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.785733 : ¬(evaluation=acc) ← (buying price=low) ∧ ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ (safety=high)
0.773658 : (evaluation=unacc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (luggage boot=big)
0.767899 : ¬(evaluation=unacc) ← ¬ (luggage boot=big) ∧ (safety=high)
0.757074 : ¬(evaluation=good) ← ¬ (maintenance cost=low)
0.751209 : ¬(evaluation=unacc) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (doors=2) ∧ ¬ (safety=low)
0.750883 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med) ∧ (safety=high)
0.739302 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
0.680285 : ¬(evaluation=vgood) ← ¬ (luggage boot=big) ∧ (safety=high)
0.678689 : ¬(evaluation=acc) ← (persons=2) ∧ ¬ (persons=4) ∧ ¬ (persons=more) ∧ (safety=low)
0.660357 : (evaluation=unacc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ (safety=high)
0.651784 : ¬(evaluation=acc) ← ¬ (buying price=high) ∧ ¬ (maintenance cost=high) ∧ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.648960 : ¬(evaluation=acc) ← ¬ (buying price=high) ∧ ¬ (maintenance cost=high) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.644195 : ¬(evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
0.626226 : ¬(evaluation=vgood) ← (persons=2) ∧ ¬ (persons=more)
0.625769 : (evaluation=vgood) ← (buying price=low) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
0.621378 : ¬(evaluation=good) ← ¬ (luggage boot=big) ∧ (luggage boot=small) ∧ (safety=high) ∧ ¬ (safety=med)
0.613631 : (evaluation=acc) ← ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med) ∧ ¬ (persons=4) ∧ (safety=high)
0.596783 : (evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ (persons=4) ∧ ¬ (persons=more)

0.589280 : ¬(evaluation=vgood) ← (luggage boot=small) ∧ ¬ (safety=high) ∧ ¬ (safety=low) ∧ ¬ (safety=med)
0.583054 : ¬(evaluation=acc) ← ¬ (luggage boot=big) ∧ (safety=high) ∧ ¬ (safety=low)
0.580432 : ¬(evaluation=good) ← ¬ (maintenance cost=low) ∧ ¬ (doors=4)
0.574908 : (evaluation=good) ← ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=low)
0.571528 : (evaluation=good) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.567548 : (evaluation=unacc) ← (persons=2) ∧ ¬ (persons=4) ∧ ¬ (persons=more) ∧ (safety=low)
0.564927 : ¬(evaluation=vgood) ← ¬ (maintenance cost=low)
0.542365 : ¬(evaluation=vgood) ← ¬ (luggage boot=big) ∧ (safety=high) ∧ ¬ (safety=low)
0.540124 : ¬(evaluation=unacc) ← ¬ (luggage boot=big) ∧ (safety=high) ∧ ¬ (safety=low)
0.532475 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=med) ∧ ¬ (doors=4) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.509409 : ¬(evaluation=acc) ← ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.504565 : ¬(evaluation=good) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ (safety=high)
0.501683 : (evaluation=acc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (luggage boot=big)
0.499645 : ¬(evaluation=acc) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=high) ∧ ¬ (safety=low) ∧ (safety=med)
0.494660 : ¬(evaluation=good) ← ¬ (maintenance cost=med) ∧ ¬ (doors=4) ∧ ¬ (safety=low)
0.493858 : ¬(evaluation=unacc) ← ¬ (safety=high) ∧ ¬ (safety=low)
0.487933 : (evaluation=unacc) ← (persons=4) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.487763 : ¬(evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.481959 : ¬(evaluation=unacc) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
0.473790 : (evaluation=vgood) ← ¬ (buying price=high) ∧ ¬ (maintenance cost=high)
0.467248 : ¬(evaluation=good) ← ¬ (safety=high)
0.460469 : (evaluation=vgood) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=high) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.460025 : ¬(evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ (persons=4) ∧ ¬ (persons=more)
0.457093 : ¬(evaluation=good) ← ⊤
0.453117 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=med) ∧ ¬ (doors=4) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.439237 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.437664 : (evaluation=unacc) ← (buying price=low) ∧ ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ (safety=high)
0.431671 : (evaluation=good) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.415213 : ¬(evaluation=acc) ← ¬ (persons=4) ∧ (persons=more) ∧ ¬ (luggage boot=med) ∧ (luggage boot=small)
0.399627 : ¬(evaluation=good) ← (persons=2) ∧ ¬ (persons=more)
0.399483 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=vhigh)
0.397810 : ¬(evaluation=acc) ← ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small) ∧ (safety=high) ∧ ¬ (safety=low)
0.387599 : ¬(evaluation=vgood) ← (persons=2) ∧ ¬ (persons=4)
0.383107 : ¬(evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=med)
0.382283 : ¬(evaluation=acc) ← ¬ (luggage boot=big) ∧ (safety=high)
0.372742 : ¬(evaluation=acc) ← ¬ (maintenance cost=vhigh) ∧ ¬ (doors=2) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.372372 : ¬(evaluation=unacc) ← (persons=2) ∧ ¬ (persons=4)
0.371963 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (doors=4) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.365264 : ¬(evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
0.343788 : ¬(evaluation=unacc) ← ¬ (doors=5more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.337127 : ¬(evaluation=acc) ← ¬ (persons=2) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ (safety=high)
0.335185 : ¬(evaluation=vgood) ← ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med) ∧ ¬ (persons=4) ∧ (safety=high)
0.334771 : ¬(evaluation=vgood) ← ¬ (maintenance cost=low) ∧ ¬ (doors=4)
0.330622 : (evaluation=good) ← ¬ (buying price=high) ∧ ¬ (maintenance cost=high)
0.325422 : (evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.321942 : ¬(evaluation=acc) ← ¬ (persons=2) ∧ ¬ (persons=4) ∧ ¬ (luggage boot=med)
0.321317 : (evaluation=vgood) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.320045 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ (buying price=med) ∧ ¬ (maintenance cost=med)
0.320039 : ¬(evaluation=good) ← (persons=2) ∧ ¬ (persons=4)
0.319987 : ¬(evaluation=acc) ← ¬ (maintenance cost=vhigh) ∧ ¬ (doors=2) ∧ ¬ (safety=low)
0.317464 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (doors=4) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.315876 : (evaluation=unacc) ← (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)

0.315323 : ¬(evaluation=good) ← (persons=4) ∧ ¬ (safety=high) ∧ ¬ (safety=low) ∧ ¬ (safety=med)
0.311176 : ¬(evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=vhigh)
0.310806 : (evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=vhigh)
0.309251 : ¬(evaluation=unacc) ← ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.298241 : ¬(evaluation=unacc) ← ¬ (buying price=vhigh) ∧ ¬ (safety=high)
0.297058 : ¬(evaluation=good) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.292935 : (evaluation=acc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ (safety=high) ∧ ¬ (safety=low) ∧ ¬ (safety=med)
0.289287 : ¬(evaluation=unacc) ← ¬ (safety=high) ∧ (safety=low) ∧ (safety=med)
0.287366 : (evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (safety=high)
0.286766 : (evaluation=unacc) ← (persons=2) ∧ ¬ (persons=more)
0.284554 : ¬(evaluation=unacc) ← ¬ (safety=high)
0.283440 : ¬(evaluation=good) ← ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med) ∧ ¬ (persons=4) ∧ (safety=high)
0.279860 : ¬(evaluation=acc) ← ¬ (buying price=high) ∧ ¬ (maintenance cost=high)
0.263071 : ¬(evaluation=good) ← ¬ (buying price=high) ∧ ¬ (safety=high)
0.260110 : ¬(evaluation=acc) ← ¬ (maintenance cost=med) ∧ ¬ (doors=4) ∧ ¬ (safety=low)
0.257467 : (evaluation=acc) ← ¬ (safety=high)
0.246398 : ¬(evaluation=acc) ← (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.245260 : ¬(evaluation=unacc) ← ¬ (buying price=high) ∧ ¬ (safety=high)
0.244084 : ¬(evaluation=vgood) ← (safety=high) ∧ (safety=low) ∧ ¬ (safety=med)
0.242651 : (evaluation=good) ← ¬ (buying price=vhigh) ∧ ¬ (safety=high)
0.241060 : ¬(evaluation=acc) ← (persons=4) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.233903 : ¬(evaluation=vgood) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (safety=high)
0.230966 : (evaluation=unacc) ← (persons=4) ∧ ¬ (safety=high) ∧ ¬ (safety=low) ∧ ¬ (safety=med)
0.227985 : ¬(evaluation=vgood) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.221836 : (evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (doors=4) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.218796 : ¬(evaluation=good) ← (buying price=low) ∧ ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ (safety=high)
0.215808 : ¬(evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.213640 : (evaluation=acc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.213520 : (evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=med) ∧ ¬ (doors=4) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.205887 : ¬(evaluation=acc) ← ¬ (maintenance cost=low)
0.199223 : ¬(evaluation=good) ← (persons=2) ∧ (persons=4) ∧ ¬ (persons=more)
0.193805 : ¬(evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (maintenance cost=med) ∧ ¬ (doors=4) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.187279 : (evaluation=acc) ← (doors=2) ∧ ¬ (doors=4) ∧ ¬ (luggage boot=big)
0.186586 : ¬(evaluation=unacc) ← ¬ (luggage boot=big) ∧ (luggage boot=small) ∧ (safety=high) ∧ ¬ (safety=med)
0.186244 : ¬(evaluation=unacc) ← ¬ (safety=high) ∧ (safety=low)
0.186095 : ¬(evaluation=acc) ← (safety=high) ∧ (safety=low) ∧ ¬ (safety=med)
0.184952 : (evaluation=good) ← ¬ (buying price=low) ∧ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ (maintenance cost=med) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.181138 : ¬(evaluation=good) ← ¬ (luggage boot=big) ∧ (safety=high) ∧ ¬ (safety=low)
0.178548 : (evaluation=vgood) ← ¬ (buying price=low) ∧ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ (maintenance cost=med) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.177045 : ¬(evaluation=vgood) ← ¬ (safety=high) ∧ (safety=low) ∧ ¬ (safety=med)
0.170975 : (evaluation=unacc) ← (safety=high) ∧ (safety=low) ∧ ¬ (safety=med)
0.166703 : ¬(evaluation=acc) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.164098 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.163224 : (evaluation=unacc) ← ¬ (maintenance cost=low)
0.160424 : ¬(evaluation=acc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ (safety=high) ∧ ¬ (safety=low)
0.156070 : ¬(evaluation=good) ← ¬ (buying price=low)
0.155056 : (evaluation=unacc) ← ¬ (buying price=low) ∧ (buying price=med) ∧ ¬ (maintenance cost=med)
0.152061 : ¬(evaluation=acc) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.149680 : (evaluation=acc) ← ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=low)
0.148708 : (evaluation=good) ← (buying price=high) ∧ ¬ (buying price=low) ∧ ¬ (maintenance cost=vhigh)
0.148406 : ¬(evaluation=good) ← ¬ (safety=high) ∧ (safety=low) ∧ (safety=med)
0.147975 : ¬(evaluation=acc) ← (luggage boot=small) ∧ ¬ (safety=high) ∧ ¬ (safety=low) ∧ ¬ (safety=med)

0.147085 : (evaluation=unacc) ← ¬ (maintenance cost=low) ∧ ¬ (doors=4)
0.142101 : ¬(evaluation=unacc) ← ¬ (maintenance cost=med) ∧ ¬ (doors=4) ∧ ¬ (safety=low)
0.141092 : (evaluation=acc) ← ¬ (buying price=high) ∧ ¬ (safety=high)
0.137795 : ¬(evaluation=acc) ← (persons=2) ∧ ¬ (persons=more)
0.132293 : ¬(evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (doors=4) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.127432 : ¬(evaluation=acc) ← (persons=4) ∧ ¬ (safety=high) ∧ ¬ (safety=low) ∧ ¬ (safety=med)
0.121049 : ¬(evaluation=vgood) ← ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=vhigh) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=high) ∧ ¬ (safety=low) ∧ (safety=med)
0.118321 : ¬(evaluation=acc) ← ¬ (buying price=low)
0.117544 : ¬(evaluation=acc) ← ¬ (maintenance cost=low) ∧ ¬ (doors=4)
0.115761 : (evaluation=acc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ (safety=high)
0.101850 : (evaluation=good) ← ¬ (buying price=low) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.101400 : ¬(evaluation=acc) ← ¬ (luggage boot=big) ∧ (luggage boot=small) ∧ (safety=high) ∧ ¬ (safety=med)
0.100864 : (evaluation=good) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (doors=2) ∧ ¬ (safety=low)
0.095284 : (evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.090773 : (evaluation=acc) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (doors=2) ∧ ¬ (safety=low)
0.082618 : ¬(evaluation=acc) ← ¬ (buying price=low) ∧ (buying price=med) ∧ ¬ (maintenance cost=med)
0.081067 : (evaluation=good) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small)
0.079519 : ¬(evaluation=acc) ← ¬ (doors=5more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.071824 : (evaluation=unacc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.069134 : ¬(evaluation=acc) ← ¬ (buying price=vhigh) ∧ ¬ (safety=high)
0.067070 : ¬(evaluation=unacc) ← ¬ (doors=4) ∧ ¬ (doors=5more) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=big) ∧ (safety=high) ∧ ¬ (safety=low)
0.055160 : (evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (doors=4) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high) ∧ ¬ (safety=low)
0.053034 : (evaluation=unacc) ← ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.052814 : (evaluation=acc) ← ¬ (safety=high) ∧ ¬ (safety=low)
0.051392 : ¬(evaluation=unacc) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med) ∧ (persons=4) ∧ ¬ (persons=more) ∧ ¬ (safety=high)
0.047061 : ¬(evaluation=acc) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (maintenance cost=high) ∧ ¬ (persons=2) ∧ ¬ (safety=low)
0.046302 : (evaluation=vgood) ← ¬ (buying price=high) ∧ ¬ (buying price=vhigh) ∧ ¬ (doors=2) ∧ ¬ (safety=low)
0.046103 : ¬(evaluation=acc) ← ¬ (safety=high) ∧ (safety=low)
0.042085 : (evaluation=unacc) ← (luggage boot=small) ∧ ¬ (safety=high) ∧ ¬ (safety=low) ∧ ¬ (safety=med)
0.040869 : (evaluation=good) ← ¬ (safety=high) ∧ (safety=low)
0.024192 : (evaluation=acc) ← (persons=2) ∧ (persons=4) ∧ ¬ (persons=more)
0.023102 : ¬(evaluation=acc) ← (persons=2) ∧ ¬ (persons=4)
0.021803 : ¬(evaluation=unacc) ← ¬ (buying price=low)
0.020115 : ¬(evaluation=good) ← ¬ (maintenance cost=vhigh) ∧ ¬ (persons=2) ∧ ¬ (luggage boot=small) ∧ ¬ (safety=low)
0.011479 : (evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (doors=4) ∧ ¬ (luggage boot=big) ∧ ¬ (safety=high)
0.008310 : (evaluation=good) ← ¬ (luggage boot=big) ∧ (safety=high)
0.006883 : ¬(evaluation=vgood) ← ¬ (buying price=low)
0.001617 : ¬(evaluation=unacc) ← (persons=2) ∧ (persons=4) ∧ ¬ (persons=more)
0.000748 : ¬(evaluation=good) ← (persons=2) ∧ ¬ (persons=4) ∧ ¬ (persons=more) ∧ (safety=low)
0.000609 : ¬(evaluation=acc) ← ¬ (safety=high) ∧ (safety=low) ∧ (safety=med)
0.000169 : ¬(evaluation=acc) ← ¬ (buying price=low) ∧ ¬ (buying price=med)
0.000041 : ¬(evaluation=good) ← ¬ (buying price=low) ∧ ¬ (buying price=med) ∧ ¬ (maintenance cost=low) ∧ ¬ (maintenance cost=med)
0.000011 : (evaluation=vgood) ← (persons=2) ∧ ¬ (persons=4) ∧ ¬ (persons=more) ∧ (safety=low)