Supplementary Materials

C Gödel Trick

In this appendix, we provide a recap of the *Gödel trick with categorical variables*, adapting its original presentation in (Daniele and van Krieken 2026) to our setting with fuzzy truth values in $[0, 1]$ discretized to $\{0, 1\}$ via a 0.5 threshold (instead of real values discretized to $\{-1, 1\}$ via the sign function). We also adapt the proof in (Daniele and van Krieken 2026, Section 8), showing that the Gödel Trick is equivalent to the classical *Gumbel-max trick* (Gumbel 1954), of which the widely used *Gumbel-softmax trick* (Jang, Gu, and Poole 2017) is a smooth approximation.

Gödel Trick Recap. Direct gradient descent on Gödel logic is prone to stalling in local minima. The Gödel Trick counteracts this by injecting noise. In our formulation, it takes a set of logits z_i (one per candidate branch of a choice operator), and applies the following three steps, depicted in Figure 1:

1. **Noise addition (only during training):** $z'_i := z_i + n_i$, with $n_i \overset{\text{i.i.d.}}{\sim} \text{Gumbel}(0, \beta)$.
2. **Re-centering:** $z''_i := z'_i - \bar{z}'$, where $\bar{z}'$ is the mean of the two largest perturbed logits.
3. **Sigmoid application:** $w_i := \sigma\left(\frac{z''_i}{T}\right)$, where $T > 0$ is a temperature hyperparameter.

At test time, we can binarize the weights w_i at 0.5, ensuring that exactly one w_i equals 1 and the rest are 0. Thanks to Theorem 1, this discretization does not alter the final Boolean predictions of the model.

Equivalence to the Gumbel-max Trick. The *Gumbel-max trick* is a reparameterization method for sampling from a categorical distribution. Let the categorical distribution be defined by the probabilities $\pi := softmax(\theta)$. Then, the probability that $\arg\max_i (\theta_i + g_i) = j$, where $g_i \sim Gumbel(0, 1)$, is precisely equal to π_j. Hence,

$$\text{onehot}(\arg\max_i (\theta_i + g_i))$$

produces an exact sample from the categorical distribution with probabilities π. This is how the Gumbel-max trick works, and the Gumbel-softmax trick is a differentiable approximation which substitutes $\text{onehot} \circ \arg\max$ with $softmax$.

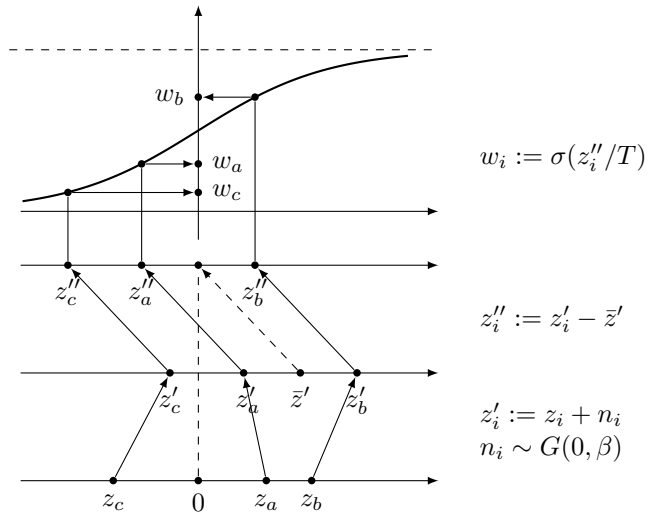


Figure 1: Given a choice operator $[a, b, c]$, the figure illustrates the differentiable three-stage procedure that turns the raw, real-valued logits z_i attached to each candidate sub-formula into logical gates $w_i \in [0, 1]$. From the bottom up: (1) during training, i.i.d. Gumbel noise $n_i \sim \text{Gumbel}(0, \beta)$ is added to each logit; (2) the mean $\bar{z}'$ of the two largest perturbed logits is subtracted from all values; (3) temperature-scaled sigmoid function is applied.

Setting $\theta_i := z_i/\beta$ and $g_i := n_i/\beta$, the perturbation step of the Gödel Trick becomes:

$$z'_i = \beta\,(\theta_i + g_i)$$

The centering step subtracts the mean of the two largest perturbed logits, which does not affect the $\arg\max$, since it simply shifts all values by the same constant. Similarly, also the multiplication by a positive constant and the application of a monotonically increasing function such as the sigmoid do not affect the $\arg\max$. Hence:

$$\arg\max_i w_i = \arg\max_i z''_i = \arg\max_i z'_i = \arg\max_i(\theta_i + g_i)$$

Since the maximum weight is by construction the only one surpassing the threshold value 0.5, discretizing the weights w_j yields precisely the same effect as $\text{onehot}(\arg\max_i(w_i))$. Thus, we can conclude that

$$\rho(w_j) = \text{onehot}(\arg\max_i (\theta_i + g_i))_j$$

for every j. Moreover, because we are using Gödel semantics and Theorem 1 applies, the thresholding of the weights is implicit when considering binarized output predictions of the entire model. Hence, in our context, the Gödel trick is equivalent to the Gumbel-max trick, which is more precise than the Gumbel-softmax one.

D Counterexample for Product Fuzzy Logic

Let us consider the disjunctive compilation of $[a, b]$:

$$(w_a \wedge a) \vee (w_b \wedge b)$$

and interpret it with the following fuzzy values: $\mathcal{I}(w_a) = 0.6$, $\mathcal{I}(a) = 0.7$, $\mathcal{I}(w_b) = 0.4$ and $\mathcal{I}(b) = 0$. Using product fuzzy logic,

$$\mathcal{I}((w_a \wedge a) \vee (w_b \wedge b)) = 0.6 * 0.7 = 0.42 < 0.5$$

whose discretized Boolean value $\rho_{0.5}(0.42) = 0$ (False). On the other hand, if $\mathcal{B}$ is the Boolean discretization of $\mathcal{I}$, so that $\mathcal{B}(w_a) = 1$, $\mathcal{B}(a) = 1$, $\mathcal{B}(w_b) = 0$ and $\mathcal{B}(b) = 0$, then

$$\mathcal{B}((w_a \wedge a) \vee (w_b \wedge b)) = 1$$

This means that the discretized explanation disagrees with the behavior of the network. Analogous counterexamples can be built for Łukasiewicz logic. Instead, with Gödel logic, the Boolean interpretation is always in accordance with the fuzzy truth values (Theorem 1):

$$\mathcal{I}((w_a \wedge a) \vee (w_b \wedge b)) = \max(\min(0.6, 0.7), \min(0.4, 0)) = 0.6$$

E Disjunctive vs Conjunctive Compilations

Recall that a choice node $[\Phi_1, \ldots, \Phi_n]$ can be compiled in two dual ways:

- Disjunctive compilation: $\bigvee_{i=1}^{n}(w_i \wedge \Phi_i)$
- Conjunctive compilation: $\bigwedge_{i=1}^{n}(\neg w_i \vee \Phi_i)$

Duality comes from De Morgan's laws: the negation of the disjunctive compilation is equivalent to the conjunctive compilation on the negated subformulas, and vice versa. Indeed,

$$\neg \bigvee_{i=1}^{n} w_i \wedge \Phi_i \equiv \bigwedge_{i=1}^{n} \neg w_i \vee \neg \Phi_i$$

and

$$\neg \bigwedge_{i=1}^{n} \neg w_i \vee \Phi_i \equiv \bigvee_{i=1}^{n} w_i \wedge \neg \Phi_i$$

Under Gödel semantics, conjunction and disjunction are interpreted as $\min$ and $\max$ respectively, so the two translations become

$$\text{Disj.}: \quad \max_i \min(w_i, \Phi_i)$$
$$\text{Conj.}: \quad \min_i \max(1 - w_i, \Phi_i)$$

Case Study: Selecting Reliable Rules. Equation (3) proposes the following LoH model, which is also used inside conjunctive neurons of the type in (6):

$$\bigwedge_{i=1}^{n} [r_i, \top] \tag{3}$$

Notice that the two weights in the compilation of a choice operator choosing only among two subformulas must sum to one.[1] Hence, we will write w_i and $1 - w_i$ for the (respective) weights of r_i and $\top$ in $[r_i, \top]$.

With conjunctive compilation, (3) yields:

$$\min_i(\min(\max(1 - w_i, r_i), \max(w_i, \top))) = \\ = \min_i(\max(1 - w_i, r_i))$$

On the other hand, with disjunctive compilation, it becomes:

$$\min_i(\max(\min(w_i, r_i), \min(1 - w_i, \top))) = \\ = \min_i(\max(\min(w_i, r_i), 1 - w_i))$$

Thus, (3) simplifies more under conjunctive compilation. This simplification has an important effect on the training dynamics. Indeed, when a rule r_i is already (almost) satisfied for a specific example (i.e., $r_i \approx 1$), also the inner $\max(1 - w_i, r_i)$ becomes ≈ 1. Hence, the outer $\min_i$ ignores that index and concentrates on a rule whose truth value is *smaller* (if any). Then, it is the weight associated to that less-satisfied rule that will receive gradient signal for updating. Intuitively, when particular data gives no evidence for preferring r_i over $\top$ (because their truth values are the same), the network also receives no signal to change the associated gate w_i. By contrast, $\max(\min(w_i, r_i), 1 - w_i)$ evaluates to $\max(w_i, 1 - w_i)$ when $r_i \approx 1$. Consequently, the loss may push w_i upwards or downwards even when the data provide no information for choosing between r_i and $\top$. These unwanted updates can slow convergence and may bias the learned subset of rules.

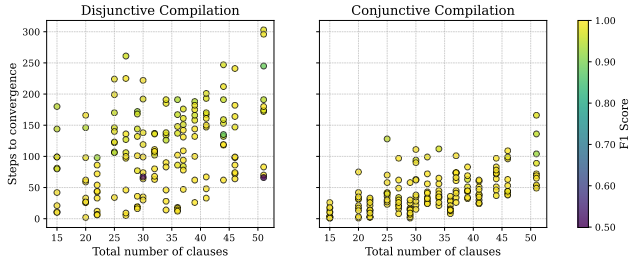
Empirical Evaluation. We conducted some experiments validating the previous findings and extending to different LoH models. The experiments are conducted in the following way. We consider 10 propositional variables and randomly generate some ground-truth clauses and some additional clauses, of width ranging from 2 to 5. For any of the 2^{10} possible Boolean interpretations, a label is produced using the ground-truth clauses as rules. Then the dataset is divided into 75% for training and 25% for evaluation. An LoH model is trained to select some rules among the ground-truth + additional rules. For any considered number of ground-truth clauses and additional clauses, the same experiment is repeated 10 times. For each execution, the test-set F1 score is

[1] Indeed, let z_1 and z_2 be the stored parameters. By design, the weights w_1 and w_2 are obtained applying the sigmoid function to $z_1 - \frac{z_1+z_2}{2} = \frac{z_1-z_2}{2}$ and $z_2 - \frac{z_1+z_2}{2} = \frac{z_2-z_1}{2}$ (respectively). The two logit values are opposite to each other, so the weights $w_1 = \sigma(\frac{z_1-z_2}{2T})$ and $w_2 = \sigma(\frac{z_2-z_1}{2T})$ must sum to 1.

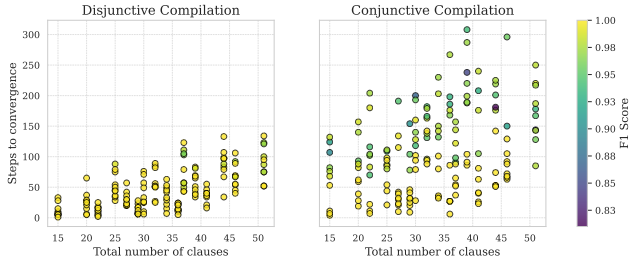
recorded, together with the number of optimization steps to convergence. The criterion we use for deciding convergence is the following: either 100% accuracy is achieved, or there is no change in accuracy for 64 consecutive steps, or a limit of 64 epochs (384 steps) is reached. The values of the hyper-parameters were fixed: 128 as batch size, 0.15 as learning rate, 1 as temperature, and $Gumbel(0, 1)$ noise.

Figure 2 reports the plots with the experiments' results for different models. Subfigure 2a refers to simple rule selection, with LoH models as in (3). Subfigure 2b does the same, but in the dual set-up: both ground-truth and additional clauses c_i are *conjunctive* clauses, and the LoH model $\bigvee_{i=1}^{n}[c_i,\ \bot]$ learns a disjunction of them. Finally, Subfigure 2c considers again rules made of (disjunctive) clauses. However, each of the m ground-truth clauses is placed on a different set $\{r_{i,1},\ r_{i,2}, \ldots, r_{i,k+1}\}$, together with k additional clauses. Then, we use the model in (4), selecting one rule per set.

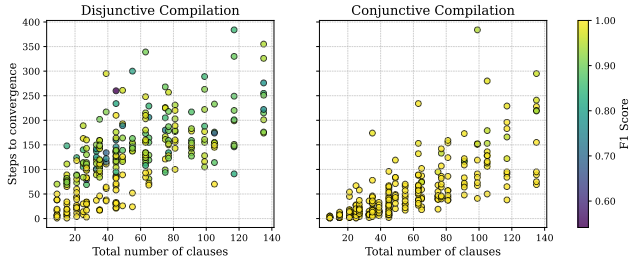
These plots corroborate our suggestion to use conjunctive compilation inside conjunctions (Subfigures 2a and 2c), and disjunctive compilation inside disjunctions (Subfigure 2b). Indeed, such compilation choices achieved better results both in terms of final accuracy and of convergence speed.



(a) With LoH model $\bigwedge_{i=1}^{n}[r_i,\ \top]$



(b) With LoH model $\bigvee_{i=1}^{n}[c_i,\ \bot]$

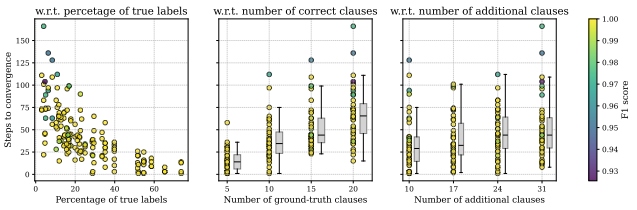


(c) With LoH model $\bigwedge_{i=1}^{m}[r_{i,1},\ r_{i,2}, \ldots, r_{i,k+1}]$

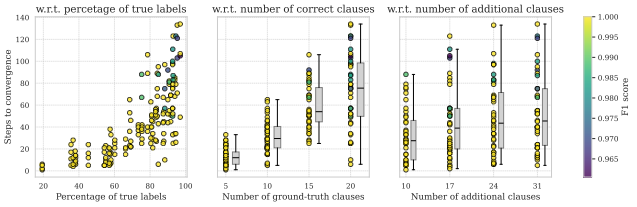
Figure 2: Comparison of disjunctive and conjunctive compilations.

F Further Experiments with Artificial Data

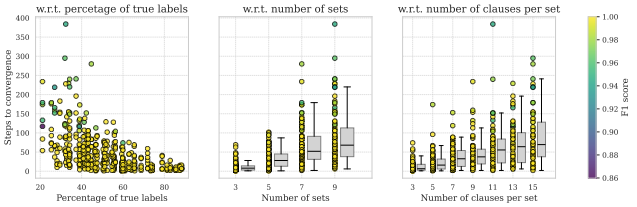
Figure 3 reprises the experiments of Appendix E. However, it considers only the best compilations (disjunctive for 3b, and conjunctive for 3a and 3c), while highlighting the influence of different experimental factors. In particular, the middle column shows how both F1 score and convergence speed have a strong negative correlation with the number of ground-truth clauses. Clearly, the more clauses in the ground truth, the more need to be selected for a perfect score, and the more difficult the learning. However, since the ground-truth clauses are independently generated, their number is also strongly correlated with data imbalance. As shown in the first column, learning a conjunction of clauses suffers most when there are too few positive samples (i.e., samples in which the ground-truth formula is satisfied). Conversely, a shortage of negative samples drives poorer performance when learning a disjunction. In contrast, the number of additional, "misleading" clauses is not as impactful as the ground-truth, as evidenced by the third column. This is true at least when the number of additional clauses is comparable to that of the ground truth.[2]



(a) With LoH model $\bigwedge_{i=1}^{n}[r_i,\ \top]$



(b) With LoH model $\bigvee_{i=1}^{n}[c_i,\ \bot]$



(c) With LoH model $\bigwedge_{i=1}^{m}[r_{i,1},\ r_{i,2}, \ldots, r_{i,k+1}]$

Figure 3: Clauses selection performance w.r.t. percentage of true labels, number of ground-truth clauses and number of additional clauses.

[2]Regarding Subfigure 3c, notice that the *total* number of added clauses is a *multiple* of the number of additional clauses *per set*.

G Wildfire Risk Assessment Task Details

Dataset generation. The dataset contains 2048 synthetic RGB images of size 256×256. For each image, we first sample Boolean assignments to the visual concepts (Forest and DryVegetation), deciding whether the scene should contain dense forest and whether the vegetation should be dry. A simple drawing routine then renders forest-like patches, and the color scheme distinguish greener from dry vegetation. The image generation process may also produce additional elements, such as houses or rivers, that are random artifact not correlated with the wildfire risk.

Independently, we sample Boolean values for the seven non-visual features (LowHum, PowerLines, etc.), as iid $Bernoulli(0.5)$ random variables. The ground-truth wildfire-risk label for each sample is obtained by evaluating Equation (2) on the full 9-dimensional Boolean vector. The images, the non-visual features and the risk labels are stored in a PyTorch dataset and split into training and test sets (75%/25%).

Model Architectures and training. The neurosymbolic models integrate a perception and a reasoning module, which are trained end-to-end:

- **Perception Module (CNN):** A three-layer Convolutional Neural Network processes $3 \times 256 \times 256$ RGB images to extract the visual predicates Forest and DryVegetation. Each block consists of a convolution (channels: $3 \to 16 \to 16 \to 16$), ReLU activation, MaxPool (2×2), and Dropout ($p = 0.15$). A final classification head outputs fuzzy values for the two visual concepts, through a $sigmoid$ activation.
- **Reasoning Module (LoH):** This layer accepts the visual predictions alongside the non-visual boolean features. It comes in the four different knowledge regimes discussed in Section 6.

Training uses standard binary cross–entropy on the wildfire label, with Adam optimizers for both the logical and perceptual parts. The CNN learning rate is $8 \cdot 10^{-4}$ and the LoH learning rate is $8 \cdot 10^{-2}$. For each knowledge regime, the training is performed 20 times with different random seeds. Analogously for ∂ILP.

H Comparison of Different Templates

Let us consider the following ground-truth CNF formula ϕ made of 5 definite clauses of width 3:

$$(\neg v_3 \vee \neg v_8 \vee v_7) \wedge (\neg v_{10} \vee \neg v_3 \vee v_4) \wedge (\neg v_1 \vee \neg v_9 \vee v_{10}) \wedge (\neg v_2 \vee \neg v_6 \vee v_8) \wedge (\neg v_4 \vee \neg v_3 \vee v_5)$$

For any of the 2^{10} possible Boolean interpretations of the propositional variables $v_1, \ldots, v_{10}$, a ground-truth label is produced using ϕ. This dataset is divided into 75% for training and 25% for evaluation, and each LoH model is trained to construct 5 clauses. The values of the hyperparameters are fixed: 128 as batch size, 0.15 as learning rate, 1 as temperature, and $Gumbel(0, 1)$ noise. Figure 4 compares the average learning curves of LoH models adhering to different templates:

- "5 clauses" conjoins five copies of $\bigvee_{i=1}^{10}[\neg v_i, v_i, \bot]$
- "5 definite clauses" uses the conjunction of five copies of formula (10) with n=10, i.e., $\bigvee_{i=1}^{10}[\neg v_i, \bot] \vee [v_1, \ldots, v_{10}]$
- "5 definite clauses of width 3" uses the conjunction of five copies of $[\neg v_1, \ldots, \neg v_{10}] \vee [\neg v_1, \ldots, \neg v_{10}] \vee [v_1, \ldots, v_{10}]$
- "5 definite clauses with heads given" uses $(\bigvee_{i=1}^{10}[\neg v_i, \bot] \vee v_7) \wedge (\bigvee_{i=1}^{10}[\neg v_i, \bot] \vee v_4) \wedge (\bigvee_{i=1}^{10}[\neg v_i, \bot] \vee v_{10}) \wedge (\bigvee_{i=1}^{10}[\neg v_i, \bot] \vee v_8) \wedge (\bigvee_{i=1}^{10}[\neg v_i, \bot] \vee v_5)$
- "5 definite clauses with first given" conjoins $\neg v_3 \vee \neg v_8 \vee v_7$ with four copies of $\bigvee_{i=1}^{10}[\neg v_i, \bot] \vee [v_1, \ldots, v_{10}]$

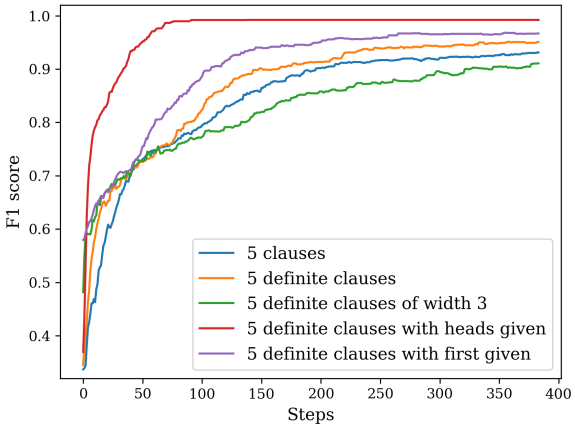


Figure 4: Average training curves, over 20 runs, of LoH formulas following different templates, for learning the same ground-truth.

We can notice that adding explicit knowledge (fixing either a clause or the clause heads) yield better learning curves than the purely syntactic alternatives. Regarding such three purely syntactic templates, the model learning definite clauses slightly outperformed the most general one, thanks to a reduced search space. However, despite having an even smaller hypothesis space, the model learning definite clauses of width 3 is the one performing worse. This may be explained by the fact that the other models can update the weights of the negative literals independently, whereas updates in $[\neg v_1, \ldots, \neg v_{10}]$ always involve more variables at the same time, or because the final formula is invariant to permuting the three chosen literals, but the parameterization is not. Anyway, one may still employ this model because it guarantees formulas of a prescribed template, irrespective of raw predictive performance. Moreover, among the 20 runs, each of the LoH models found the 100%-correct formula multiple times. This suggest that trying different runs and picking the best can be a useful strategy.

I Datasets Properties & Runtimes

For each tabular dataset, Table 1 summarizes the number of features before and after binarization, together with references and size.

Table 1: Datasets properties.

Dataset	# instances	# classes	# original features	# binary features
adult (Becker and Kohavi 1996)	32561	2	14	155
bank-marketing (Moro, Rita, and Cortez 2014)	45211	2	16	88
banknote (Lohweg 2012)	1372	2	4	17
blogger (blo 2012)	100	2	5	15
chess (Bain and Hoff 1994)	28056	18	6	40
connect-4 (Tromp 1995)	67557	3	42	126
letRecog (Slate 1991)	20000	26	16	155
magic04 (Bock 2004)	19020	2	10	79
mushroom (Schlimmer 1981)	8124	2	22	117
nursery (Rajkovic 1989)	12960	5	8	27
tic-tac-toe (Aha 1991)	958	2	9	27
wine (Cortez et al. 2009)	178	3	13	37

Table 2 reports the approximated runtime of a single training plus evaluation run, for each benchmark discussed in Section 7.1. The values are relative to the selected hyperparameters, and the table also reports the corresponding models' parameter count. All experiments were conducted on a cluster node equipped with an Nvidia RTX A5000 with 60GB RAM.

J Hyperparameters

The following outlines the hyperparameter search spaces for the models presented in Section 7. The hyperparameter choices for each dataset are available in the code repository.

- Decision Tree: min_samples_split (2–50), max_depth (2–50), min_samples_leaf (1–50).
- Random Forest: n_estimators (50–500), min_samples_split (2–50), max_depth (2–50), min_samples_leaf (1–50).
- XGBoost: max_depth (5–20), n_estimators (10–500), learning_rate (10^{-3}–$2 \cdot 10^{-1}$).
- Neural Network: number of hidden layers (1–3), number of units per layer (4–128), learning rate (10^{-4}–10^{-1}). Batch size fixed at 256, and ReLU activations.
- DLN: number of hidden layers (1–10), number of units per layer (16–512), grad_factor (1.–2.), learning rate (10^{-3}–10^{-1}), τ (1–100). Batch size fixed at 128.
- MLLP/CRS: number of hidden layers (1,3), number of units per layer (16–256), weight decay (10^{-8}–10^{-2}), learning rate (10^{-4}–10^{-1}), random binarization rate (0–0.99). Batch size (128) and learning rate scheduler were set to the default value.
- LoH: learning rate (0.01–0.2), Gumbel noise scale β (0.4–1.2), temperature (0.4–1.2), and temperature rescaling factor applied every 10 epochs (0.9925–1). Like NN and MLLP/CRS, we also optimized the architecture, allowing the TPE algorithm to select the layer type (any-clause vs fixed-size-clauses), the number of hidden layers (1–2), layer sizes (16–256), and whether the output layer is conjunctive or disjunctive (with the other layers alternating). In case of fixed-size-clauses layer type, also the hyperparameter k (2–8) was tuned for each layer.

K Decision Rules of Visual Tic-Tac-Toe

Both CRS and LoH allow for the automatic extraction of logical formulas. In order to interpret such decision rules, we need to assign proper names to the input propositions. In particular, for visual tic-tac-toe, there are three input propositions for every image in the tic-tac-toe grid, each corresponding to an output unit of the feature-extractor CNN. The assignment of labels (such as X, O or B) to such units is done by thresholding their average activations with respect to each image class (0, 1, and 2), in the following way:

- if the average activation is never > 0.5, the label for the unit is $\bot$, which can later be simplified from the formulas;
- if the average activation is > 0.5 for one class and < 0.5 for the other two, the label for the unit is the one corresponding to the activating class;
- if the average activation is > 0.5 for two classes and < 0.5 for the other, the label for the unit is the logical negation of the non-activating class;
- if the average activation is always > 0.5, the label for the unit is $\top$, which can later be simplified from the formulas.

As an example, if an output unit of the CNN exhibits an average activation below 0.5 for images of class 1 but above 0.5 for images of the other two classes, we assign it the label $\neg O$ (recalling that digit 1 was associated with O). In this way, we can assign names to the input propositions of the logical models, by combining the labels of the CNN output units with the positions in the 3×3 tic-tac-toe grid.

Table 3 reports the *best* decision rules learned by CRS and LoH on the Visual Tic-Tac-Toe task. The formulas were simplified removing the appearances of $\top$ and $\bot$, and also removing redundant clauses. We do not provide the formulas learned by DLN because the implementation of DLN we used did not have a function to write them in a human-readable way. Moreover, to boost performance, DLN actually learns an ensemble with majority voting, not a single formula. The DLN run with highest Symbolic eval achieved .885 F1-score.

Table 2: Average runtimes relative to a single training + evaluation run.

Dataset	DLN		MLLP/CRS		LoH	
	Time (s)	# Params	Time (s)	# Params	Time (s)	# Params
adult	520	57680	499	74260	287	17584
bank-marketing	615	38576	687	70556	452	81612
banknote	17	23728	14	4807	12	7790
blogger	4	36592	4	3485	6	6195
chess	956	53568	970	117774	546	29464
connect-4	1214	51344	650	25542	873	55212
letRecog	414	41280	511	126302	528	86518
magic04	467	50960	271	19440	364	31104
mushroom	110	31024	78	16422	67	82900
nursery	284	51840	302	48331	311	68186
tic-tac-toe	25	18288	38	4002	32	11310
wine	2	6960	5	6400	4	2640

References

Aha, D. 1991. Tic-Tac-Toe Endgame. UCI Machine Learning Repository.

Bain, M., and Hoff, A. 1994. Chess (King-Rook vs. King). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C57W2S.

Becker, B., and Kohavi, R. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

2012. BLOGGER. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5HK6P.

Bock, R. 2004. MAGIC Gamma Telescope. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C52C8B.

Cortez, P.; Cerdeira, A.; Almeida, F.; Matos, T.; and Reis, J. 2009. Wine Quality. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56S3T.

Daniele, A., and van Krieken, E. 2026. Gradient-based optimization on gödel logic as discrete local search. *arXiv preprint arXiv:2503.01817*.

Gumbel, E. J. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Lohweg, V. 2012. Banknote Authentication. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C55P57.

Moro, S.; Rita, P.; and Cortez, P. 2014. Bank Marketing. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5K306.

Rajkovic, V. 1989. Nursery. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5P88W.

Schlimmer, J. 1981. Mushroom. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5959T.

Slate, D. 1991. Letter Recognition. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5ZP40.

Tromp, J. 1995. Connect-4. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C59P43.

Table 3: Best decision rules learned on the Visual Tic-Tac-Toe task. The labels X_i and $\neg O_j$ were assigned post-hoc to each output of the trained CNN in the way explained in appendix K.

Model		Symb. eval	Formula
CRS	DNF	.991	$(\neg O_1 \wedge \neg O_5 \wedge \neg O_9) \vee (\neg O_3 \wedge \neg O_5 \wedge \neg O_7)$ $\vee(\neg O_2 \wedge \neg O_4 \wedge \neg O_7 \wedge \neg O_9) \vee (\neg O_1 \wedge \neg O_2 \wedge \neg O_3 \wedge \neg O_4 \wedge \neg O_7)$ $\vee(\neg O_1 \wedge \neg O_2 \wedge \neg O_3 \wedge \neg O_4 \wedge \neg O_8) \vee (\neg O_1 \wedge \neg O_2 \wedge \neg O_3 \wedge \neg O_4 \wedge \neg O_9)$ $\vee(\neg O_1 \wedge \neg O_2 \wedge \neg O_3 \wedge \neg O_5 \wedge \neg O_8) \vee (\neg O_1 \wedge \neg O_2 \wedge \neg O_3 \wedge \neg O_6 \wedge \neg O_7)$ $\vee(\neg O_1 \wedge \neg O_2 \wedge \neg O_3 \wedge \neg O_6 \wedge \neg O_8) \vee (\neg O_1 \wedge \neg O_2 \wedge \neg O_4 \wedge \neg O_6 \wedge \neg O_7)$ $\vee(\neg O_1 \wedge \neg O_3 \wedge \neg O_4 \wedge \neg O_7 \wedge \neg O_8) \vee (\neg O_1 \wedge \neg O_3 \wedge \neg O_6 \wedge \neg O_8 \wedge \neg O_9)$ $\vee(\neg O_1 \wedge \neg O_4 \wedge \neg O_5 \wedge \neg O_6 \wedge \neg O_7) \vee (\neg O_1 \wedge \neg O_4 \wedge \neg O_5 \wedge \neg O_6 \wedge \neg O_8)$ $\vee(\neg O_1 \wedge \neg O_4 \wedge \neg O_7 \wedge \neg O_8 \wedge \neg O_9) \vee (\neg O_1 \wedge \neg O_6 \wedge \neg O_7 \wedge \neg O_8 \wedge \neg O_9)$ $\vee(\neg O_2 \wedge \neg O_3 \wedge \neg O_4 \wedge \neg O_5 \wedge \neg O_8) \vee (\neg O_2 \wedge \neg O_3 \wedge \neg O_4 \wedge \neg O_6 \wedge \neg O_9)$ $\vee(\neg O_2 \wedge \neg O_3 \wedge \neg O_6 \wedge \neg O_7 \wedge \neg O_9) \vee (\neg O_2 \wedge \neg O_4 \wedge \neg O_5 \wedge \neg O_6 \wedge \neg O_8)$ $\vee(\neg O_2 \wedge \neg O_4 \wedge \neg O_5 \wedge \neg O_6 \wedge \neg O_9) \vee (\neg O_2 \wedge \neg O_4 \wedge \neg O_5 \wedge \neg O_8 \wedge \neg O_9)$ $\vee(\neg O_2 \wedge \neg O_5 \wedge \neg O_6 \wedge \neg O_7 \wedge \neg O_8) \vee (\neg O_2 \wedge \neg O_5 \wedge \neg O_7 \wedge \neg O_8 \wedge \neg O_9)$ $\vee(\neg O_3 \wedge \neg O_4 \wedge \neg O_5 \wedge \neg O_6 \wedge \neg O_8) \vee (\neg O_3 \wedge \neg O_4 \wedge \neg O_5 \wedge \neg O_6 \wedge \neg O_9)$ $\vee(\neg O_3 \wedge \neg O_4 \wedge \neg O_6 \wedge \neg O_8 \wedge \neg O_9) \vee (\neg O_3 \wedge \neg O_4 \wedge \neg O_7 \wedge \neg O_8 \wedge \neg O_9)$ $\vee(\neg O_3 \wedge \neg O_6 \wedge \neg O_7 \wedge \neg O_8 \wedge \neg O_9)$
	CNF	.984	$(\neg O_1 \vee \neg O_2 \vee \neg O_3) \wedge (\neg O_1 \vee \neg O_4 \vee \neg O_7)$ $\wedge(\neg O_1 \vee \neg O_5 \vee \neg O_9) \wedge (\neg O_2 \vee \neg O_5 \vee \neg O_8)$ $\wedge(\neg O_3 \vee \neg O_5 \vee \neg O_7) \wedge (\neg O_3 \vee \neg O_6 \vee \neg O_9)$ $\wedge(\neg O_4 \vee \neg O_5 \vee \neg O_6) \wedge (\neg O_7 \vee \neg O_8 \vee \neg O_9)$ $\wedge(\neg O_2 \vee \neg O_3 \vee \neg O_4 \vee \neg O_9) \wedge (\neg O_2 \vee \neg O_5 \vee \neg O_7 \vee \neg O_9)$
LoH	DNF	1.00	$(X_1 \wedge X_2 \wedge X_3) \vee (X_4 \wedge X_5 \wedge X_6)$ $\vee(X_7 \wedge X_8 \wedge X_9) \vee (X_1 \wedge X_4 \wedge X_7)$ $\vee(X_2 \wedge X_5 \wedge X_8) \vee (X_3 \wedge X_6 \wedge X_9)$ $\vee(X_1 \wedge X_5 \wedge X_9) \vee (X_3 \wedge X_5 \wedge X_7)$
	CNF	.999	$(X_1 \vee X_5 \vee X_9) \wedge (X_3 \vee X_5 \vee X_7)$ $\wedge(X_1 \vee X_2 \vee X_6 \vee X_7) \wedge (X_1 \vee X_3 \vee X_4 \vee X_8)$ $\wedge(X_1 \vee X_3 \vee X_5 \vee X_8) \wedge (X_1 \vee X_3 \vee X_6 \vee X_8)$ $\wedge(X_1 \vee X_5 \vee X_6 \vee X_7) \wedge (X_1 \vee X_6 \vee X_7 \vee X_8)$ $\wedge(X_2 \vee X_3 \vee X_4 \vee X_9) \wedge (X_2 \vee X_4 \vee X_5 \vee X_9)$ $\wedge(X_2 \vee X_4 \vee X_7 \vee X_9) \wedge (X_2 \vee X_5 \vee X_6 \vee X_7)$ $\wedge(X_2 \vee X_5 \vee X_7 \vee X_9) \wedge (X_2 \vee X_6 \vee X_7 \vee X_9)$ $\wedge(X_3 \vee X_4 \vee X_5 \vee X_8) \wedge (X_3 \vee X_4 \vee X_5 \vee X_9)$ $\wedge(X_3 \vee X_4 \vee X_8 \vee X_9) \wedge (X_1 \vee X_2 \vee X_3 \vee X_4 \vee X_7)$ $\wedge(X_1 \vee X_2 \vee X_3 \vee X_6 \vee X_9) \wedge (X_1 \vee X_2 \vee X_5 \vee X_6 \vee X_8)$ $\wedge(X_1 \vee X_4 \vee X_5 \vee X_6 \vee X_8) \wedge (X_1 \vee X_4 \vee X_7 \vee X_8 \vee X_9)$ $\wedge(X_2 \vee X_4 \vee X_5 \vee X_6 \vee X_8) \wedge (X_3 \vee X_6 \vee X_7 \vee X_8 \vee X_9)$