Neurosymbolic Theory Revision through Predicate Invention

Neurosymbolic Artificial Intelligence
XX(X):2–??

DOI: 10.1177/ToBeAssigned
www.sagepub.com/

SAGE

Sieben Bocklandt[1,2], Vincent Derkinderen[1,2], Wannes Meert[1,2] and Luc De Raedt[1,2,3]

Abstract

Neurosymbolic models combine prior background knowledge with neural networks, achieving the advantages of both. Unfortunately, these models typically assume the knowledge to be perfect and complete. This limits their their usability, as it is unrealistic in real-world environments, particularly in domains that require both structured reasoning and perception. For example, it is infeasible in autonomous driving to encode all traffic rules and safety conditions. To address this issue, we propose a novel methodology that iteratively revises an initial imperfect and incomplete logical background theory with neural concepts. Our approach, termed NeTheR, performs a limited number of high-impact modifications to improve the model's performance while maintaining the integrity of the original symbolic structure. Historically, theory revision in inductive logic programming has been achieved by adding or removing symbolic literals to improve logical models. NeTheR extends this by leveraging predicate invention to introduce neural concept representations whose meaning is not predefined, allowing the system to discover useful concepts directly from subsymbolic data and integrate them into the symbolic theory. These high-impact modifications, such as inserting a neural concept into a specific part of the model, are identified using a variant of the Sharpe ratio that estimates their risk-adjusted performance gain. Empirical evaluation demonstrates that NeTheR is the only method that consistently achieves statistically significant improvements over the initial theory across all evaluated settings. In contrast to competing approaches, which exhibit unstable or marginal gains, NeTheR robustly and reproducibly enhances predictive performance by effectively combining existing symbolic knowledge with the expressive power of neural networks.

Keywords

Theory revision, Neurosymbolic AI, Neural ILP

Prepared using sagej.cls *[Version: 2017/01/17 v1.20]*

1 Introduction

Neurosymbolic AI (NeSy) is an emerging subfield within artificial intelligence (Gartner 2025) that combines the strengths of two traditionally distinct approaches: symbolic reasoning and neural computations. Symbolic AI, which emphasizes logic, rules, and human-like reasoning, excels in handling abstract, structured knowledge and can be easily interpreted by humans (De Raedt 2008; Russell and Norvig 2020). Neural approaches, on the other hand, have contributed significant advances in pattern recognition, perception, and data-driven learning, especially with unstructured data such as images and text. Such neural approaches do, however, often lack explainability, logical consistency, and the ability to generalize from limited data (Rudin et al. 2021). NeSy aims to bridge these gaps, creating systems that cannot only learn from vast amounts of data, but also reason at the same time in a structured, interpretable manner. By integrating neural networks' learning capabilities with symbolic reasoning's precision and transparency, NeSy holds the potential to tackle complex problems more effectively.

Despite recent progress, integrating symbolic knowledge with neural learning remains challenging when the available symbolic knowledge is incomplete or partially incorrect. In many practical settings, domain experts can provide an initial set of logical rules or background knowledge, but these rules rarely capture the full complexity of the environment. While several neurosymbolic approaches can learn symbolic structures jointly with neural representations, they often treat the provided knowledge as fixed or rely primarily on learning new structure from data. However, discarding existing knowledge and relearning everything from scratch is undesirable, particularly in low-data regimes where prior knowledge can provide valuable inductive bias. Instead, it is often more natural to start from the available knowledge and refine it where necessary. This perspective aligns with theory revision in logic programming, where imperfect rule sets are incrementally corrected and extended. Bringing this principle to neurosymbolic systems allows models to remain faithful to prior knowledge while adapting it based on data.

We address these shortcomings by proposing NeTheR* (**Ne**urosymbolic **The**ory **R**evision), an approach to automatically revise imperfect symbolic knowledge through the introduction of new symbolic literals and neural concepts. Unlike many NeSy systems, where neural predicates correspond to predefined concepts (e.g., object or

[1]Department of Computer Science, KU Leuven, Belgium
[2]Leuven.AI - KU Leuven Institute for AI, Belgium
[3]Center for Applied Autonomous Systems, Örebro University, Sweden

Corresponding author:
Sieben Bocklandt
Email: sieben.bocklandt@kuleuven.be
*All code will be made publicly available upon paper acceptance.

Prepared using sagej.cls

attribute detectors), NeTheR can invent neural predicates whose meaning is not specified beforehand. These neural concepts are learned directly from subsymbolic data and integrated into the symbolic theory where they provide the greatest benefit. This approach combines the advances in neurosymbolic research with the principles of theory revision (Mooney and Shavlik 2021), a field that focuses on refining knowledge bases such as logical rules (Muggleton and Buntine 1988; De Raedt 1992; Ourston and Mooney 1990), or probabilistic networks (Mahoney and Mooney 1993; Ramachandran and Mooney 1996). NeTheR can refine the existing symbolic structure with new, learned concept representations, thus compensating for incomplete knowledge.

Example 1.1. *As an example task, consider a binary classification problem over a collection of photographs. The goal is to classify whether a given picture belongs to the positive class, defined as images that were taken by either Alice or Bob and that depict either a bird or a fish. Additionally, animals photographed by Alice should not have the ability to fly. All other images belong to the negative class. The only annotated information is the available set of Boolean encoded categorical variables:* `Photographer`*,* `Bones`*, and* `Gills`*, abbreviated as P, B, and G respectively. An expert attempts to solve this task using the logic formula below, which is based on expert knowledge that birds have hollow bones and all fish have gills.*

$$(P = Alice \vee P = Bob) \wedge (B = Hollow \vee G = True)$$

This solution is imperfect as it does not cover birds without hollow bones, such as penguins, nor the extra constraint for Alice. Existing theory revision methods are not able to correct this as they are limited to symbolic data. In contrast, NeTheR introduces neural concepts whose value can also depend on subsymbolic data, like images, thereby increasing expressivity. An example of the result of NeTheR is shown in Figure 1, which includes the introduction of a neural concept and a new symbolic literal to refine the formula. For example, given enough data, the neural concept learns to recognize whether an animal has wings, thus is probably a bird.

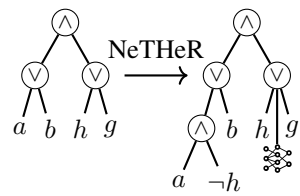


Figure 1. Example initial theory and its revision by NeTheR. We use propositional symbols to describe properties of each image: a denotes that the picture was taken by Alice ($P =$ Alice), b denotes that the picture was taken by Bob ($P =$ Bob), h denotes that the depicted animal has hollow bones ($B =$ Hollow), and g denotes that the depicted animal has gills (Gills $=$ True). NeTheR introduces a new symbolic literal $\neg h$ to exclude many of the flying animals, and additionally incorporates a neural concept to revise the initial theory.

Our key contribution is the introduction of NeTheR, a framework that (1) improves neurosymbolic models by refining imperfect background knowledge in a data-driven manner, and (2) extends theory revision with neural predicate invention, allowing the system to introduce neural concepts that do not have a predefined semantic interpretation. This enables the model to discover useful intermediate concepts directly from subsymbolic data and incorporate them into symbolic rules. To efficiently identify impactful modifications to the theory, we further propose a variant of the Sharpe ratio that prioritizes revisions with high expected performance gains relative to their uncertainty.

2 Background

We first introduce propositional logic as a foundation for formal reasoning, arithmetic circuits for efficient exact inference on propositional logic, and neurosymbolic AI, which integrates symbolic reasoning with neural networks.

2.1 Propositional Logic

We consider a symbolic input space $\mathcal{X}_S$ consisting of a finite set of feature variables $\mathcal{X}_S = \{X_1, X_2, \ldots, X_n\}$, where each feature variable X_i is associated with its own domain $\mathcal{D}_i$. A feature variable may be either categorical, with a finite domain of discrete values, or continuous, with a domain that is a subset of $\mathbb{R}$. An interpretation $I_{\mathcal{X}_S}$ assigns to each feature variable $X_i \in \mathcal{X}_S$ a value $I(X_i) \in \mathcal{D}_i$. To reason about feature variables in propositional logic, we introduce Boolean variables defined by predicates over feature variables, which we call *symbolic literals* in the remainder of this work. A symbolic literal is a Boolean expression of the form $(X_i \bowtie v)$, where $X_i \in \mathcal{X}_S$, $\bowtie$ an (in)equality operator and $v \in \mathcal{D}_i$; each symbolic literal evaluates to true or false under an interpretation I depending on whether the predicate holds for the assigned value $I(X_i)$. A propositional logic formula F is inductively defined as a symbolic literal, a disjunction $F_1 \vee F_2$, a conjunction $F_1 \wedge F_2$, or a negation $\neg F_1$, with the usual Boolean semantics, and therefore acts as a Boolean classifier over the input space $\mathcal{X}_S$, evaluating to true or false for any interpretation $I_{\mathcal{X}_S}$. An interpretation I is said to *satisfy* a formula F, denoted $I \models F$, if F evaluates to true under I. For example, the interpretation $I = \{Colour \mapsto Red,\ Height \mapsto 175\}$ satisfies the symbolic literals $(Colour = Red)$ and $(Height < 180)$, and consequently satisfies the formula $(Colour = Red) \wedge (Height < 180)$.

A *logic circuit* represents a logic formula as a directed acyclic graph of leaf nodes (literals) and inner nodes with an associated type (either disjunction, conjunction, or negation), as shown in Figure 1. Logic circuits explicitly support the reuse of subcircuits, i.e., a node can have more than one parent node. We use the terms logic circuit and logic formula interchangeably throughout this work.

2.2 Tractable Probabilistic Inference in Logic Circuits

While propositional logic provides a deterministic classification mechanism, many real-world applications require reasoning under uncertainty, which can be incorporated by compiling the logic circuit into an equivalent circuit with specific structural properties.

These properties allow the compiled circuit to be evaluated under probabilistic semantics, yielding a probabilistic (arithmetic) circuit, and crucially enable tractable probabilistic inference by guaranteeing that circuit evaluation is linear in the size of the circuit. Specifically, a formula must satisfy three structural properties (Darwiche and Marquis 2002): it is in *negation normal form* (NNF) if negation occurs only directly on Boolean variables; it is *decomposable* if, for every conjunction node, the conjunct branches do not share variables; and it is *deterministic* if, for every disjunction node, the disjunct branches are mutually exclusive, meaning that no interpretation satisfies more than one branch simultaneously.

Knowledge compilation algorithms transform an arbitrary logic formula into an equivalent logic circuit satisfying these properties, commonly referred to as decomposable deterministic negation normal form (d-DNNF). These properties guarantee that the resulting circuit admits tractable probabilistic inference. In particular, a d-DNNF circuit can be systematically converted into an arithmetic circuit, enabling efficient computation of probabilities. We use the PySDD (Meert 2017) software package to perform this compilation.

2.3 Arithmetic Circuits

To perform probabilistic inference, each literal l in the compiled circuit is associated with a *probability function* $p_l(\mathbf{x}) \in [0, 1]$, representing the probability that the literal is true given input $\mathbf{x}$. For symbolic literals, this reduces to a Boolean evaluation on symbolic input $I_{\mathcal{X}_S}$:

$$p_l(I_{\mathcal{X}_S}) = \begin{cases} 1 & \text{if } I_{\mathcal{X}_S} \models l \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Given these probability functions, the compiled logic circuit can be evaluated as an *arithmetic circuit (AC)*. This replaces logical operators with arithmetic operators: conjunction nodes become multiplication, and disjunction nodes become addition, while literal nodes evaluate to their probability functions.

Formally, the arithmetic circuit rooted at node n evaluates as:

$$p_n(\mathbf{x}) = \begin{cases} p_l(\mathbf{x}) & \text{if } n \text{ is a literal } l \\ \prod_{d \in in(n)} p_d(\mathbf{x}) & \text{if } n \text{ is a conjunction} \\ \sum_{d \in in(n)} p_d(\mathbf{x}) & \text{if } n \text{ is a disjunction} \end{cases} \tag{2}$$

with $in(n)$ the input nodes of node n. This arithmetic circuit computes the probability that the logical formula is satisfied given the literal probabilities, and enables exact probabilistic inference in time linear in the circuit size.

2.4 Neurosymbolic AI

In a probabilistic logic circuit, each literal l is evaluated using its associated probability function $p_l(\mathbf{x})$. In a pure inference setting, these probability values are fixed and specified a priori. In a learning setting, however, the probability functions can instead be treated

as learnable parameters that are optimized from data. In this case, the circuit defines a differentiable computational structure through which gradients can propagate, allowing the parameters governing the literal probabilities to be learned.

Extending this idea, Manhaeve et al. (2021) proposed parametrizing some literals with the output of a neural network. In their framework, the neural network represents the probability function of the literal. These neural literals, referred to as *neural predicates* in the logic programming setting, allow the system to infer symbolic properties from subsymbolic inputs such as images, while integrating seamlessly into the arithmetic circuit.

In this work, we adopt a similar mechanism, and use the term *neural concepts* to refer to literals nc whose probability function $p_{nc}(\mathbf{x})$ is represented by a neural network.

We use the term neural concepts to emphasize a key difference with traditional neurosymbolic systems such as DeepProbLog (Manhaeve et al. 2021), where neural predicates are assumed to correspond to predefined symbolic concepts with known semantic meaning, and the neural network is trained to recognize these concepts. In contrast, neural concepts in our setting do not have predefined semantics. Instead, their meaning emerges entirely through training, guided by the circuit structure and supervision signal. The neural network thus learns to output a literal's probability in a way that is consistent with the logical constraints encoded in the circuit.

In addition to the symbolic input space $\mathcal{X}_S$, we thus consider a neural input space $\mathcal{X}_N$, which consists of subsymbolic data such as images, multivariate time series, or other high-dimensional signals. Formally, $\mathcal{X}_N$ is a set of tensors, where each element $\mathbf{x} \in \mathcal{X}_n$ may represent, for example, a single image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$, a collection of multiple images $\mathbf{x} \in \mathbb{R}^{K \times H \times W \times C}$, or a multivariate time series $\mathbf{x} \in \mathbb{R}^{T \times d}$. Figure 2 shows an example evaluation of an arithmetic circuit combining symbolic and neural probability functions.

Because the resulting arithmetic circuit consists entirely of differentiable operations, the neural network parameters can be trained end-to-end using gradient-based optimization. This enables learning abstract concepts from indirect supervision, while leveraging the logical structure of the circuit to constrain and guide learning.

3 Problem Setting

We consider a binary classification task defined over symbolic and subsymbolic data, in the presence of a possibly imperfect background model that the user may wish to revise.

Binary classification. Let $\mathcal{X} = \mathcal{X}_S \times \mathcal{X}_N$ denote the input space, consisting of a symbolic input space $\mathcal{X}_S$ and a subsymbolic input space $\mathcal{X}_N$. An input instance is denoted by $\mathbf{x} = (\mathbf{x}_S, \mathbf{x}_N) \in \mathcal{X}$, thus consisting of a symbolic and subsymbolic part. Let $\mathcal{Y} = \{0, 1\}$ denote the output space of Boolean labels. We assume the existence of an unknown target function $M^* : \mathcal{X} \to \mathcal{Y}$, which represents the true labeling mechanism. In practice, the true function M^* is not accessible. Instead, we are given a training set $E \subseteq \mathcal{X} \times \mathcal{Y}$ consisting of labeled instances. Each element $(\mathbf{x}_i, y_i) \in E$ contains an input instance $\mathbf{x}_i \in \mathcal{X}$ and its corresponding observed label $y_i \in \mathcal{Y}$. The label y_i may be

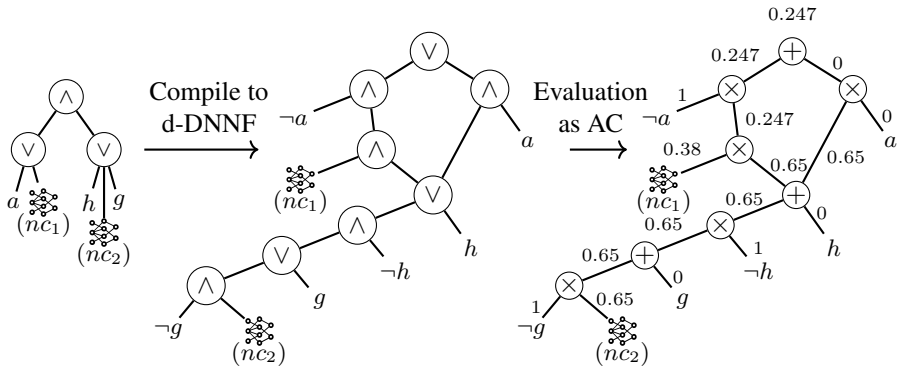


Figure 2. A logic formula is compiled into a d-DNNF and evaluated as an arithmetic circuit given the interpretation $\{a \rightarrow false, h \rightarrow false, g \rightarrow false\}$ with subsymbolic data feeding the neural networks, by which they evaluate to 0.38 and 0.65. These weights, and those of the symbolic input variables, are denoted above them. The weights propagate upwards, and those results are similarly denoted above the intermediate nodes.

provided by a human annotator, a measurement process, or another labeling system, and is assumed to be generated by the unknown target function M^*.

The goal of binary classification is to learn a model $M : \mathcal{X} \rightarrow \mathcal{Y}$ that approximates M^* and correctly predicts labels for both seen and unseen instances. Since M^* is unknown, the model must be learned from the labeled training set E. The hypothesis space $\mathcal{H}$ consists of all logic circuits whose leaf nodes are either symbolic literals or neural concepts, as defined in Section 2.

Each hypothesis $M \in \mathcal{H}$ defines a function $M : \mathcal{X} \rightarrow [0, 1]$, where inference is performed by evaluating the arithmetic circuit obtained from compiling the logic circuit as described in Section 2.3. This choice mirrors well-known neurosymbolic systems such as DeepProbLog (Manhaeve et al. 2021) and Scallop (Li et al. 2023), which use arithmetic circuits to compute probabilistic outputs during inference and learning. The output $M(\mathbf{x})$ represents the predicted probability of the positive class. During evaluation, this probabilistic output can be converted into a Boolean prediction by thresholding at 0.5 yielding a model $M : \mathcal{X} \rightarrow \mathcal{Y}$.

Additional knowledge. Our setting also comprises an initial, imperfect model $M' \in \mathcal{H}$ that serves as additional information to find M^*. This addition is motivated by the scenario wherein an expert already has an idea, albeit incorrect or incomplete, of the target model or a previously learned model, and aims to automatically revise it in a data-driven manner. The goal is thus to stay close to M' whilst increasing performance. We therefore also include a limit d_{max} on the allowed maximal distance between the input model M' and model M, measured using a distance function $d : \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{R}_{\geq 0}$.

Summary. **Given** training set E, initial model $M' \in \mathcal{H}$, and distance bound d_{max}, we must **find** $M \in \mathcal{H}$ such that

$$\underset{M \in \mathcal{H}}{\operatorname{argmax}} \; F_1(M, M^*) \quad \text{s.t. } d(M, M') \leq d_{max}, \tag{3}$$

where

$$F_1(M, M^*) = \mathbb{E}_{\mathbf{x} \sim D}[F_1\big(M(\mathbf{x}), M^*(\mathbf{x})\big)]$$

denotes the expected F_1-score of M compared to M^*, with respect to input distribution $\mathcal{D}$ over $\mathcal{X}$. We use the F_1-score because the input theory is intended to distinguish the positive class from the remaining classes, making performance on the positive class more important. If both classes are equally important, the F_1-score can be replaced by balanced accuracy in both problem setting and the following method. A natural choice for d is the minimal number of modification steps required to obtain M from M'. The types of modifications we consider are discussed in the next section.

4 Our Approach: NeTheR

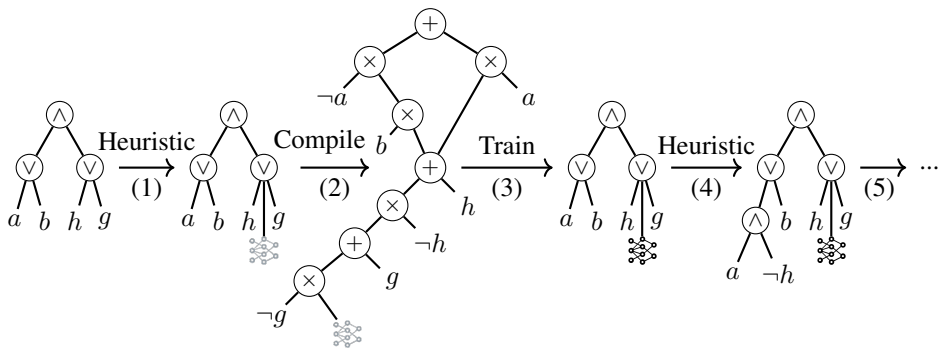


Figure 3. Example of the revision process by NeTheR. (1) An untrained neural concept is added; (2) The result is compiled to an arithmetic circuit; (3) This circuit allows us to efficiently compute probabilities and is end-to-end differentiable, such that the neural concept(s) are trained as part of the logical circuit; (4) Once trained, NeTheR performs again a new modification which may add a new symbolic literal; (5) This cycle repeats until the maximum number of modifications is reached or until modifications would decrease performance. An important task is then to efficiently identify high-impact modifications: where do we insert a new neural concept or symbolic literal; or what existing concepts should we remove?

We now introduce *NeTheR*, a greedy theory revision approach that incrementally improves a logic circuit through the incorporation of new symbolic literals and neural concepts, inspired by classical symbolic theory revision. The goal is to identify a sequence of structural modifications that improves predictive performance while remaining close to the original model. An overview is shown in Figure 3, where both a neural concept and a symbolic literal are introduced to refine the formula.

We formalise theory revision as an optimisation problem over circuit nodes and their structural replacements. Let $M^{(0)} = M'$ denote the initial logic circuit, and let $\mathcal{N}^{(t)}$ denote the set of nodes in the circuit $M^{(t)}$ at revision step t.

A modification at node n replaces the subcircuit rooted at n with a new subcircuit n', producing a revised model $M^{(t+1)} = M^{(t)}[n \leftarrow n']$. Our objective is to construct a sequence of revisions $M^{(0)} \to M^{(1)} \to \cdots \to M^{(T)}$ that maximises predictive performance while limiting deviation from the original model $M^{(0)}$.

Because the space of possible revision sequences is combinatorial, ***NeTheR*** adopts a greedy strategy that selects, at each step, the locally optimal modification according to a performance objective subject to a proximity constraint. Each greedy step solves:

$$M^{(t+1)} = M^{(t)}[n \leftarrow n'] \text{ with } \operatorname*{argmax}_{\substack{n \in \mathcal{N}^{(t)} \\ n' \in \mathcal{M}(n)}} F_1\big(M^{(t)}[n \leftarrow n'](X), Y\big) \text{ and } t < d_{\max}. \quad (4)$$

where $\mathcal{M}(n)$ denotes the set of admissible replacement subcircuits at node n.

4.1 Modification Types

Let M be a logic circuit and n a node in this circuit. Theory revision operates by locally modifying subcircuits rooted at n. These modifications fall into two categories:

- **Generalisation**, which makes the model less restrictive and may increase the set of inputs classified as positive. Structurally, generalisation is achieved by replacing n with *true* (removal) or by introducing a new node m in disjunction, $n \vee m$.
- **Specialisation**, which makes the model more restrictive and may decrease the set of inputs classified as positive. Structurally, specialisation is achieved by replacing n with *false* (removal) or by introducing a new node m in conjunction, $n \wedge m$.

To determine which modifications are most promising, we analyse the influence of each node through intervention. We consider two types of intervention. Either a node n is replaced by *true*, $M[n \leftarrow \mathit{true}](x)$, or n is replaced by *false*, $M[n \leftarrow \mathit{false}](x)$. These intervention models allow us to identify the examples whose classification depends on node n, which we refer to as the influence set:

- **Generalisation influence set**

$$E^{(n)}_{\text{gen}} = \{(x, y) \in E \mid M(x) = 0 \ \wedge \ M[n \leftarrow \mathit{true}](x) = 1\}. \quad (5)$$

  These are the examples that are not covered by M, but can be covered by generalising node n.
- **Specialisation influence set**

$$E^{(n)}_{\text{spec}} = \{(x, y) \in E \mid M(x) = 1 \ \wedge \ M[n \leftarrow \mathit{false}](x) = 0\}. \quad (6)$$

  These are the examples that are covered by M, but not anymore when specialising node n.

These influence sets identify exactly the examples whose classification can be changed through modification of node n, and are used to learn new symbolic literals and neural concepts. We now formalise the three modification types.

1. **Removal of a literal.**
   The literal n is replaced with a constant:
   $$n' \in \{true, false\}.$$
   Replacing n with *true* performs generalisation, while replacing it with *false* performs specialisation.
2. **Symbolic literal insertion.**
   A new leaf node m representing a symbolic literal is introduced and combined with n:
   $$n' = \begin{cases} n \vee m & \text{(generalisation)} \\ n \wedge m & \text{(specialisation).} \end{cases}$$
   The literal of leaf node m is learned from data using a decision tree of depth 1 (i.e., a decision stump) trained on the corresponding influence set. For generalisation, the stump is trained on $E^{(n)}_{\text{gen}}$ to identify conditions under which the prediction should become positive. For specialisation, it is trained on $E^{(n)}_{\text{spec}}$ to identify conditions under which the prediction should become negative.
   Decision stumps provide sufficient expressivity to generate Boolean atoms over both categorical and continuous variables, as explained in Section 2.1.
3. **Neural concept insertion.**
   A new leaf node nc representing a neural concept is introduced and combined with n:
   $$n' = \begin{cases} n \vee nc & \text{(generalisation)} \\ n \wedge nc & \text{(specialisation).} \end{cases}$$
   The neural concept is implemented as a neural network trained on the corresponding influence set. For generalisation, the network is trained on $E^{(n)}_{\text{gen}}$ to recognise additional positive patterns. For specialisation, it is trained on $E^{(n)}_{\text{spec}}$ to recognise patterns that should be excluded.
   These neural concepts represent properties that are not directly available in symbolic form, but must be inferred from subsymbolic inputs such as images or time series. Importantly, unlike traditional neurosymbolic systems such as DeepProbLog (Manhaeve et al. 2021), where neural predicates have predefined semantic meanings and knowledge is assumed to be specified perfectly, in our setting, neural concepts do not have a predefined meaning. They acquire meaning through training, guided by the circuit structure and supervision signal.

To ensure a structured and interpretable hypothesis space, neural concepts cannot be directly combined with other neural concepts. For example, expressions such as

$nc_1 \wedge nc_2$ and $nc_1 \vee nc_2$ are not allowed, while structured combinations such as $(x \vee nc_1) \wedge nc_2$ remain valid.

This restriction ensures that the number of candidate modifications grows linearly with circuit size. For a circuit with l symbolic literals, n neural concepts, and i inner nodes, the number of candidate modifications is at most $3l + 2i + n + 2$ (see Appendix A). Finally, if no candidate modification improves performance, the current model is considered locally optimal and the revision process terminates.

4.2 Modification Selection

When revising a logic circuit, symbolic and neural modifications differ fundamentally in how their impact can be evaluated. For symbolic revisions, the effect of a modification can be determined exactly prior to applying it. This is because symbolic modifications introduce only deterministic circuit structure, and the resulting model can be evaluated tractably using standard arithmetic circuit inference.

In contrast, inserting a neural concept introduces a component whose behavior depends on learnable parameters. While the circuit structure determines how the new node nc influences the model, its probability function $p_{nc}(x)$ is represented as a neural network with initially unknown parameters. These parameters must be learned from data through training. As a result, the performance of the modified model $M[n \leftarrow n']$ cannot be determined exactly without performing this training procedure.

Consequently, evaluating candidate neural revisions is significantly more computationally expensive than evaluating symbolic revisions, as each candidate requires training a neural network to determine its impact. This makes exhaustive evaluation of all possible neural modifications infeasible in practice. Therefore, enabling tractable revision requires estimating the potential impact of neural modifications prior to training, allowing promising revisions to be identified efficiently without incurring the full computational cost of neural optimization.

Upper and lower bounds. Let n be a candidate insertion node. The sets $E^{(n)}_{\text{gen}}$ and $E^{(n)}_{\text{spec}}$, defined above in equations 5 and 6 respectively, characterise the examples whose classification can be changed by generalising or specialising node n, respectively. Since the new neural concept nc is initially untrained, its exact predictions on the influence set are unknown. However, we can derive bounds on the achievable performance by considering upper and lower bound probability functions p^* and $p_\perp$.

- *Best case (B).* The neural concept correctly identifies all examples in $E^{(n)}_{\text{gen/spec}}$:

$$p^*(x) = y \quad \text{for all } (x, y) \in E^{(n)}_{\text{gen/spec}}. \tag{7}$$

- *Worst case (W).* The neural concept makes incorrect predictions on all examples:

$$p_\perp(x) = 1 - y \quad \text{for all } (x, y) \in E^{(n)}_{\text{gen/spec}}. \tag{8}$$

These probability functions define the best-case and worst-case modified subcircuits n^* and $n_\perp$, respectively. The corresponding performance bounds are:

$$B = F_1\big(M[n \leftarrow n^*](X), Y\big), \quad W = F_1\big(M[n \leftarrow n_\perp](X), Y\big). \tag{9}$$

By construction, the performance of any neural concept inserted at node n is bounded by:

$$W \;\leq\; F_1\big(M[n \leftarrow n'](X), Y\big) \;\leq\; B. \tag{10}$$

These bounds allow us to estimate the potential benefit and risk of inserting a neural concept without training the neural network.

Selecting the modification with maximal best-case score B is optimistic and favours expressive revisions, analogous to model selection criteria such as AIC and BIC (Akaike 1998; Schwarz 1978). In contrast, selecting the modification with maximal worst-case score W is conservative and favours safer symbolic revisions.

Sharpe ratio. To consider both the best and worst case scenarios simultaneously, we create a more sophisticated heuristic based on the *Sharpe ratio* (Pav 2021). The Sharpe ratio S is a widely used metric in finance to assess the risk-adjusted performance of an investment p. It is derived from the statistical t-test and is mathematically defined as the difference between R_p, which is the expected return of the investment p, and R_f, which is the return from a theoretical risk-free investment f. The denominator σ_p is the standard deviation of p's excess returns, quantifying the variability in p's performance. I.e., $S = (R_p - R_f)/\sigma_p$. In essence, this formula highlights how much excess return is achieved per unit of risk taken. A higher Sharpe ratio S indicates a better risk-adjusted performance. As the metric is particularly useful for comparing investments with varying levels of risk, it is frequently used in the context of multi-armed bandit problems (Cassel et al. 2018; Khurshid et al. 2025; Xi et al. 2021). Similar to the investment p, we do not know the performance of a neural concept prior to training and only know its best and worst case impact. In other words, we can use the Sharpe ratio S to objectively trade off the risk (W) and reward (B) of each circuit modification, and select the best modification as the one with the highest S.

Using the distribution X_{nc} to denote the performance of neural concept nc in a specific location of the logic circuit, we interpret X_{nc} as the distribution over all possible trained outcomes of the modified model i.e., over all attainable F_1 scores between the worst-case bound W and the best-case bound B.

Its expectation $\mathbb{E}[X_{nc}]$ serves as a substitute for R_p, while the standard deviation $\sigma[X_{nc}]$ replaces σ_p. Finally, the risk-free rate R_f corresponds to the F_1-score of the best modification with a precisely known impact: either the removal of a literal or neural concept, the addition of a new symbolic literal, or performing no modification at all:

$$S = \frac{\mathbb{E}[X_{nc}] - R_f}{\sigma[X_{nc}]}. \tag{11}$$

The distribution X_{nc} depends on both the neural network's architecture and the classification task. In our empirical evaluation, we assume that X_{nc} is a two-point weighted random variable whose mean is $\mu = \alpha B + (1-\alpha)W$ and variance is $Var(X_{nc}) = \alpha(B-\mu)^2 + (1-\alpha)(W-\mu)^2$, and we use these as parameters for a

shifted normal distribution. This results in

$$S = \frac{\alpha B + (1-\alpha)W - R_f}{\sqrt{\alpha((1-\alpha)B - (1-\alpha)W)^2 + (1-\alpha)(\alpha W - \alpha B)^2}} \tag{12}$$

where $\alpha \in (0,1)$ shifts the distribution closer to the best case when $\alpha > 0.5$. This equation assumes the presence of risk, i.e., $B \neq W$. Risk-free modifications are only selected when the best S is negative, i.e., when the expected F_1-score $\mathbb{E}[\mathbf{X}_{nc}]$ is worse than the best risk-free modification. A full example of a single modification selection is shown in Appendix B.

Dynamic Sharpe ratio. Instead of choosing a fixed value for α, we determine it dynamically using an exponential smoothing scheme: after training a neural network nc, we adjust α such that the estimated $\mathbb{E}[X_{nc}]$ better matches nc's actual performance. Denoting α_t as the correct value according to the neural network's performance, we move α closer to this value by updating it $\alpha \leftarrow (\alpha + \alpha_t)/2$. In addition to the periodic updates of α, we also choose a more informed initial value by randomly selecting k insertion modifications and evaluating the predicted performance with its actual performance. We select a modification with worst case W and best case B and train the network, resulting in F_1-score x. α is then $\frac{x-W}{B-W}$, showing how close we are to the best case. When $k > 1$, α is the average of all k estimates.

By dynamically adjusting α, the proposed heuristic provides a more accurate estimate of risk and captures performance trends across different scenarios. In contrast, the strategies that always select the worst-case performance W, the best-case performance B, or a fixed value $\alpha = 0.5$ constitute trivial alternatives. Always selecting W systematically undervalues the neural concept, leading it to be consistently rejected in favor of symbolic revision, as it is predicted to worsen the theory. Conversely, always selecting B overvalues the neural concept, causing it to be chosen in all cases. Fixing $\alpha = 0.5$ similarly fails to adapt to changing performance conditions. The ablation study in Appendix C.1 confirms that the dynamic Sharpe-ratio-based selection outperforms these trivial baselines.

4.3 Algorithm

We now discuss the outline of NeTheR using Algorithm 1. First, we determine a suitable, initial α parameter (line 3) to better estimate the potential impact of each possible neural concept, as explained in the previous paragraph. Then, we iteratively apply the modification with the highest Sharpe ratio (line 6 and 7) until the maximum distance is reached (line 5), or until the next best modification is to make no change at all (line 8). Note that it is expensive to compute distance $d(M, M')$ in equation 3, so we approximate it by counting the number of performed modifications, potentially overestimating the true distance and thus stopping too early. After this selection, we recompile the theory into a d-DNNF representation, and perform end-to-end parameter learning to update the neural network weights in case the modification added a neural concept (line 9). Once the learning process is complete, we can verify whether the estimated performance was

correct, and update α accordingly to improve future estimates (line 9). Finally, if the F_1-score improved the best found result, we update M and continue to further modify the result (line 10). In case the F_1-score did not improve, M did not change and the while loop repeats with the next best untried modification, using the best model found so far. Finally, we return the best found model (line 11).

Algorithm 1 NeTheR

Input:
1: Imperfect logic circuit M', Maximum distance d_{max},
2: Training and validation set $T, V \subseteq E$, Number of modifications to initialize k
Output: Revised model M
3: $\alpha \leftarrow$ initialise_alpha(M', k)
4: $M \leftarrow M'$
5: **while** $d(M', M) \leq d_{max}$ **do**
6: $\quad \mathcal{M} \leftarrow$ all untried modifications $(n \leftarrow n')$ for M
7: $\quad M'' \leftarrow argmax_{M[n \leftarrow n'] | (n,n') \in \mathcal{M}}\ S(M[n \leftarrow n'], \alpha)$
8: $\quad$ **if** $M'' = M$ **then** *break*
9: $\quad$ **if** M'' includes a neural concept **then** $train(M'', T)$, update α
10: $\quad$ **if** $F_1(M'', V) > F_1(M, V)$ **then** $M \leftarrow M''$
11: **return** M

Training the model. When inserting a neural concept, we train the underlying neural network in an end-to-end fashion with the rest of the logic circuit, using indirect supervision. For example, if our logic circuit is $M = nc_1 \wedge (x \vee nc_2)$, with nc_1 and nc_2 neural concepts and x a symbolic literal, then we train M using our training dataset; there is no direct label for the neural network. By compiling the logic circuit into d-DNNF prior to training its arithmetic version, we have the benefit of probabilistic semantics (cf. Figures 2 and 3, and Sections 2.3 and 2.4). Integrating a new neural concept enables the logic circuit to capture additional useful features. This is particularly advantageous when the subsymbolic data contains information not present in symbolic form, while also increasing robustness against noise in the symbolic data as the inserted neural concept may mitigate inconsistencies.

During training, NeTheR only updates the parameters of the newly inserted neural network and keeps the other parameters frozen. It is however possible to also update the previously trained parameters alongside the newly inserted ones, such that they can be better tuned with respect to each other. To test this hypothesis, we performed an ablation study that is included in Appendix C.2, which showed that not freezing the earlier parameters did not impact classification performance, but significantly increased runtime. Therefore we keep freezing the previously existing parameters in the rest of our empirical evaluation.

5 Evaluation

We empirically evaluate NeTheR on relevant benchmarks, assessing its effectiveness at solving the revision problem (cf. Section 3) compared to alternative approaches. In the theoretical setting, the target model M^* is unknown and must be approximated using labeled data. For the purpose of experimental evaluation, we will define a ground-truth target model M^* and generating labeled instances from it. This allows us to quantitatively assess how well the revised model approximates the true underlying model.

We investigate two scenarios: a structurally informed scenario where the initial model M' is structurally close to the target model M^*, and a structurally uninformed scenario where this is not the case.

5.1 Experimental Setup

We consider 9 open-source benchmark datasets, as listed in Table 1, with more details included in Appendix D.

Table 1. The 9 open-source datasets that are used in the experimental evaluation, with d_S denoting the number of symbolic features.

Dataset	#instances	d_S	type of subsymbolic data
Austin Texas Housing (ATH) (Pierce 2021)	5508	37	image
AWA2 (Xian et al. 2019)	5000	85	image
CelebA (Liu et al. 2015)	10000	40	image
CUB (Wah et al. 2011)	11788	312	image
derm7pt (Kawahara et al. 2019)	1011	13	image
FMA (Defferrard et al. 2017)	9217	14	audio + echonest features
Motifs, based on TSMD-bench (Van Wesenbeeck et al. 2024)	5000	180	time series
Pascal VOC (Everingham et al. 2010)	17125	20	image
StarLightCurves (Rebbapragada et al. 2009)	9236	21	time series

Six of those datasets have images as subsymbolic data, whilst three datasets demonstrate our method's ability to also operate on non-image data, such as music and time series.

The open-source datasets do neither include an imperfect initial model M', nor a target model M^* from which we can extract a structurally close M'. Hence, we generate for each dataset 24 logic circuits M^*, resulting in 216 binary classification problem instances in total. From M^* we obtain an initial model M' that is imperfect but structurally close, by systematically removing variables, lowering the F_1-score until it is in the range of 0.6 to 0.9. Importantly, we also remove these variables from the training dataset, to simulate the setting one encounters in practice where this data is not present. This means NeTheR must revise M' using only the subsymbolic features, and the symbolic features already present in M'.

For the experiment without structural information, we use as initial model M' a decision tree that is learned for each problem instance. After filtering on imperfect initial models M', i.e., those with an F_1-score below 0.9, we obtain 171 instances. These represent an expert's imprecise knowledge that is not necessarily structurally close to M^*. Similar to the experiment with structural information, we restrict the symbolic features to only those present in M'. More information on the generation of these problem instances is given in Appendix D. Appendix H shows an instance of the problem for both the structurally informed and uninformed experiment, together with the corresponding revised model by NeTHeR. Appendix E includes an additional, similar, experiment on the derm7pt dataset (Kawahara et al. 2019), wherein we use the existing dataset labels instead of M^*.

As our distance function d tracks the number of modifications, we set d_{max} to the number of systematic removals from M^* to M'. Each dataset was split into 80% training, 10% validation and 10% test data. For NeTheR, we use $k = 3$ to estimate an initial α, and freeze the parameters of previously added neural concepts when training.

Used neural networks. Depending on the type of subsymbolic data, we use a different type of neural network to parameterize the neural concepts. For images, we use a pre-trained ResNet-50 and train a 6-layer multi-layer perceptron (MLP) concatenated to the last fully connected ResNet-50 layer; for music (FMA dataset), we only use and train the MLP; and for time series (StarLightCurves and Motifs), we use and train the InceptionTime network (Fawaz et al. 2020). All of these are trained for 50 epochs with patience set to 5. Sklearn is used to learn the decision stump in case of a new symbolic literal.

5.2 Baseline Comparison

To evaluate the effectiveness of NeTheR, we compare against the following five baseline methods:

CMR A state-of-the-art concept-based model that was initialised with M', and that was allowed to learn 20 additional rules (Debot et al. 2024).

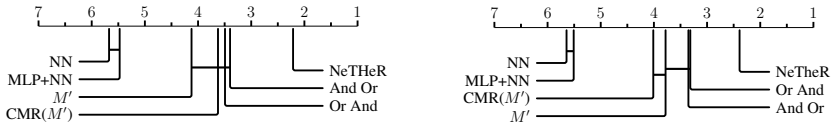
And Or A simple but very general extension of M': $(M' \vee nc_1) \wedge nc_2$, with nc new neural concepts.

Or And A simple but very general extension of M': $(M' \wedge nc_1) \vee nc_2$, with nc new neural concepts.

NN A single neural network exploiting only subsymbolic data. For each dataset, we use the same neural network architecture as used to parameterize the neural concepts, but without the logic circuit.

MLP+NN A multimodal network combining the single neural network with an MLP to also exploit symbolic data (Ahsan et al. 2020; Gessert et al. 2020).

The *NN* approach learns a single neural network, and takes on average 6 minutes. The remaining baselines, *CMR*, *And Or*, *Or And*, and *MLP+NN* each learn two networks, and complete on average in 9, 7, 8, and 12 minutes respectively. In comparison, NeTheR



(a) M' with structural information **(b)** M' without structural information

Figure 4. Critical difference diagrams (Demšar 2006; Benavoli et al. 2016), demonstrating the effectiveness of NeTheR. It plots the average rank (closer to 1 is better) and connects with a horizontal line those methods whose performance is not significantly different. We include input model M'. More detailed numerical results are included in Table G. 6 and Table G. 7 in Appendix G, which shows that using NeTheR increases the F_1-score with 0.07 on average.

trains an average of 1.5 neural networks during the search phase, and another 2.2 to estimate parameter α[†]. Again on average, NeTheR completes the full revision process in approximately 22 minutes: 12.5 minutes are spent estimating an initial α, 3.5 minutes are spent training neural networks during the search phase, and the remaining time is spent enumerating the possible modifications and compiling the revised circuit. In case a sufficient estimate for α is available, the initial step of the algorithm can be avoided entirely. Information on the used computing infrastructure is found in Appendix F.

Figure 4 contains the critical difference diagrams for both experiments, showing that NeTheR clearly outperforms the other approaches. Interestingly, the multimodal neural network *MLP+NN* does not perform significantly better than the single neural network *NN*. This indicates that the additional complexity does not necessarily translate to better results, because the subsymbolic data may provide a sufficient signal to replace the symbolic features. We also observe that both of these methods frequently perform worse than M'. This illustrates well that starting from an existing imperfect model M' has obvious performance benefits. The other three methods, *CMR*, *And Or*, and *Or And*, do exploit the existing information from M'. Still, their performance often significantly lags behind NeTheR's. We believe this is because NeTheR is more robust against bad performing neural concepts through a more fine-grained placement. In the other methods neural networks have a central role, heavily influencing the performance: for *And Or* and *Or And* this is obvious, in case of *CMR*, a neural predictor is key in deciding which logical rule determines the prediction. This also explains why NeTheR is more performant in data-poor settings compared to the other methods, as shown in Appendix C.3.

To conclude, NeTheR outperforms its competitors on the revision task in both the structurally informed and uninformed case. Furthermore, the results underscore the effectiveness of NeTheR in combining imperfect knowledge M' with subsymbolic information.

[†]Even though we use $k{=}3$ when evaluating NeTheR, some instances have fewer locations to insert a neural network such that the average k is lower.

6 Related Work

NeTheR is based on three key features: (1) it leverages an imperfect initial model M' by refining it with (multiple) neural concepts or symbolic literals, (2) it introduces neural concepts that are not pre-defined, (3) thereby combining symbolic and subsymbolic data.

Deep Probabilistic Logic Programming NeSy systems that integrate probabilistic logic programming with neural networks have gained significant attention for their ability to combine reasoning with perception. DeepProbLog (Manhaeve et al. 2021), DeepStochLog (Winters et al. 2022), NeurASP (Yang et al. 2020), and Scallop (Li et al. 2023) are just some examples of systems that enable probabilistic reasoning over the outputs of neural networks. Each of these systems is used under the assumption of perfect symbolic knowledge, where the neural concepts have a predefined intended meaning. In contrast, our contribution does not assume perfect knowledge (key feature 1), and we introduce neural concepts that do not have a pre-envisioned meaning (key feature 2).

Neural predicate invention. Predicate invention systems extend an initial vocabulary with new higher-level predicates that are defined in relation to the existing ones, to improve rule learning. In the neural version, the vocabulary includes neural predicates (Liang et al. 2025; Sha et al. 2024). For example, Sha et al. (2024) explores predicates representing shapes, and other visual features, to learn higher-level concepts in structured visual scenes. Similar to the NeSy systems, these (neural) predicates have a pre-envisioned meaning, and are typically learned before performing predicate invention. In contrast, our work introduces concepts whose meanings are not known a priori (key feature 2) but are instead learned from data alongside the theory, requiring the system to both discover and contextualize their roles without explicit semantic grounding from the user. Recent work on neurosymbolic decision tree learning also introduces neural predicates during structure induction (Möller et al. 2025). In this setting, neural predicates are added as part of a top-down tree construction process and trained jointly with the induced structure. In contrast, our approach uses existing knowledge (key feature 1), allowing neural concepts to be inserted at arbitrary nodes. This enables explicit reasoning about the structural impact of each invented predicate, and allows estimating best- and worst-case performance prior to training the underlying neural network.

Interpretable concept-based models. Concept-based models (CBM) are machine learning models that use human-interpretable concepts as an intermediate layer in its decision-making process. The canonical CBM, named the Concept Bottleneck Model, extracts concepts from raw data using a concept encoder, the results of which are passed to a task predictor that makes the final prediction (Koh et al. 2020). These models have been explored in settings with incomplete or imperfect concepts, aligning with challenges related to imperfect knowledge. Two notable neurosymbolic CBMs that are developed for such settings, DCR (Barbiero et al. 2023) and CMR (Debot et al. 2024), learn logic rules over concepts and utilize neural representations to enhance their accuracy, akin to our notion of neural concepts. Among these, only CMR has the ability to both learn rules and incorporate pre-defined logic rules (key feature 1). This means that, even though CMR was not presented as such, it is a viable solution for our problem setting.

However, CMR's use of neural concepts is close to the behavior of *And Or*, while NeTheR allows for a much more refined placement of neural concepts. This explains our empirical observation, that NeTheR is more effective in terms of improving F_1-score while maintaining the integrity of the original input.

Theory revision. Research on theory revision combines human-engineered, rule-based knowledge with empirically learned knowledge using symbolic, probabilistic, and neural approaches (Ginsberg et al. 1988; Mooney and Shavlik 2021). Systems such as DUCE and (N)EITHER (Baffes and Mooney 1993; Muggleton 1987; Ourston and Mooney 1990) refined propositional rule bases using abductive reasoning and inductive learning, while CIGOL, FOCL, FORTE, CLINT and MIS (Muggleton and Buntine 1988; Richards and Mooney 1991; De Raedt 1992; Kókai et al. 1997) tackled the revision of first-order Horn clause theories by leveraging inductive logic programming. In contrast to NeTheR, these systems are not fit to capture concepts available in subsymbolic data (key feature 3).

7 Discussion

In this work, we explore the revision of propositional theories by augmenting them with neural concepts. We characterize the trade-off this introduces between interpretability and predictive performance and identify the methodological advances required to generalize the approach to first-order theories.

Interpretability of neural concepts. Neurosymbolic AI methods exploit existing knowledge to increase accuracy, robustness, with less data. Interpretability also increases compared to purely neural models, because it enables the use of symbolic information and makes their combination symbolically understandable (i.e., and, or, negation). E.g., "$Gills = True \wedge neural\_concept$" is more interpretable than only "neural_concept". NeTheR introduces neural concepts that do not have a prior intended meaning (Section 2.4). This trades-off interpretability for accuracy compared to a fully symbolic initial model M'. In case the user wishes to better understand the neural concepts, standard explainability methods can be applied, such as saliency maps, gradient-based attribution methods (e.g., Integrated Gradients (Sundararajan et al. 2017), DeepLIFT (Shrikumar et al. 2017), and Layer-wise Relevance Propagation (Bach et al. 2015)), model-agnostic approaches like LIME (Ribeiro et al. 2016) and SHAP (Lundberg and Lee 2017), and concept-based techniques such as TCAV (Kim et al. 2018) and activation maximization (Erhan et al. 2009).

Future work: generalizing to first-order. First-order theories are out of scope for this work, but an important aspect to consider is the type of modifications that are explored. In traditional systems like FORTE (Richards and Mooney 1991), there are two types of modifications: (1) changing existing rules by removing antecedents or through inverse resolution and (2) adding antecedents/rules through predicate invention. The first type should not be a problem to expand with neural concepts. The second type however increases the search space significantly when compared with NeTheR, so the question of which constraints we can apply on the possible neural concepts merits more research.

For example, (1) would variables be allowed to become part of the neural concept and how would these be chosen; (2) Is it performant to only add neural concepts to existing rules, thus not allowing new rules to be made? Depending on the answers, the search space may become much larger and new heuristics have to be developed to approximate the performance without training each option.

8 Conclusion

We introduced NeTheR, an approach for revising imperfect propositional theories that introduces neurally learned concept representations, leveraging neural predicate invention. An altered version of the Sharpe ratio guides the revision, identifying and selecting the modification with the highest impact. The covered problem setting is inspired by earlier work on theory revision, but more suited to modern research fields such as NeSy by combining subsymbolic data with categorical symbolic data. Empirical evaluation demonstrates that using the Sharpe ratio to dynamically adapt to the observed performance bounds of neural concepts yields more reliable improvements than relying on static assumptions. Most importantly, in contrast to related methods that either fail to improve reliably or exhibit unstable behaviour across tasks, NeTheR is the only approach that consistently achieves a statistically significant increase in predictive performance across all considered settings.

9 Reproducibility

The code for NeTheR and for the experimental evaluation will be released upon paper acceptance. The experimental setup is detailed in Section 5.1, and all evaluations are conducted on open-source datasets, with preprocessing steps described in Appendix D. The proposed algorithm is presented in Section 4.3, and the results of ablation studies are provided in Appendix C.

Acknowledgements

This research received funding from the Flemish Government under the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” programme , from the KU Leuven Research Fund (C14/24/092) and from the European Union (ERC, DeepLog, 101142702). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. LDR is also supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

Ahsan MM, Alam TE, Trafalis TB and Huebner P (2020) Deep MLP-CNN model using mixed-data to distinguish between COVID-19 and non-covid-19 patients. *Symmetry* 12(9): 1526.

Akaike H (1998) *Information Theory and an Extension of the Maximum Likelihood Principle*. New York, NY: Springer New York. ISBN 978-1-4612-1694-0, pp. 199–213. DOI:10.1007/978-1-4612-1694-0_15. URL https://doi.org/10.1007/978-1-4612-1694-0_15.

Bach S, Binder A, Montavon G, Klauschen F, Müller KR and Samek W (2015) On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE* 10(7): 1–46. DOI:10.1371/journal.pone.0130140. URL https://doi.org/10.1371/journal.pone.0130140.

Baffes PT and Mooney RJ (1993) Symbolic revision of theories with m-of-n rules. In: *IJCAI*. Morgan Kaufmann, pp. 1135–1142.

Barbiero P, Ciravegna G, Giannini F, Zarlenga ME, Magister LC, Tonda A, Lio P, Precioso F, Jamnik M and Marra G (2023) Interpretable neural-symbolic concept reasoning. In: *ICML*, *Proceedings of Machine Learning Research*, volume 202. PMLR, pp. 1801–1825.

Benavoli A, Corani G and Mangili F (2016) Should we really use post-hoc tests based on mean-ranks? *J. Mach. Learn. Res.* 17(1): 152–161.

Cassel AB, Mannor S and Zeevi A (2018) A general approach to multi-armed bandits under risk criteria. In: *COLT*, *Proceedings of Machine Learning Research*, volume 75. PMLR, pp. 1295–1306.

Christ M, Braun N, Neuffer J and Kempa-Liehr AW (2018) Time series feature extraction on basis of scalable hypothesis tests (tsfresh - A python package). *Neurocomputing* 307: 72–77.

Darwiche A and Marquis P (2002) A knowledge compilation map. *J. Artif. Int. Res.* 17(1): 229–264.

De Raedt L (1992) *Interactive Theory Revision: An Inductive Logic Programming Approach*. Academic Press Ltd.

De Raedt L (2008) *Logical and relational learning*. Cognitive Technologies. Springer.

Debot D, Barbiero P, Giannini F, Ciravegna G, Diligenti M and Marra G (2024) Interpretable concept-based memory reasoning. In: *NeurIPS*.

Defferrard M, Benzi K, Vandergheynst P and Bresson X (2017) FMA: A dataset for music analysis. In: *18th International Society for Music Information Retrieval Conference (ISMIR)*. URL https://arxiv.org/abs/1612.01840.

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* 7: 1–30.

Erhan D, Courville A and Vincent P (2009) Visualizing higher-layer features of a deep network. *Technical Report, Univeristé de Montréal* .

Everingham M, Van Gool L, Williams CKI, Winn J and Zisserman A (2010) The pascal visual object classes (voc) challenge. *International Journal of Computer Vision* 88(2): 303–338.

Fawaz HI, Lucas B, Forestier G, Pelletier C, Schmidt DF, Weber J, Webb GI, Idoumghar L, Muller P and Petitjean F (2020) Inceptiontime: Finding alexnet for time series classification. *Data Min. Knowl. Discov.* 34(6): 1936–1962.

Gartner (2025) Hype cycle for artificial intelligence, 2025. URL https://www.gartner.com/en/newsroom/press-releases/2025-08-05-gartner-hype-cycle-identifies-top-ai-innovations-in-2025.

Accessed:2025-02-05.

Gessert N, Nielsen M, Shaikh M, Werner R and Schlaefer A (2020) Skin lesion classification using ensembles of multi-resolution efficientnets with meta data. *MethodsX* 7: 100864. DOI:https://doi.org/10.1016/j.mex.2020.100864. URL https://www.sciencedirect.com/science/article/pii/S2215016120300832.

Ginsberg A, Weiss SM and Politakis P (1988) Automatic knowledge base refinement for classification systems. *Artificial Intelligence* 35(2): 197–226. DOI:https://doi.org/10.1016/0004-3702(88)90012-4. URL https://www.sciencedirect.com/science/article/pii/0004370288900124.

Kawahara J, Daneshvar S, Argenziano G and Hamarneh G (2019) Seven-point checklist and skin lesion classification using multitask multimodal neural nets. *IEEE Journal of Biomedical and Health Informatics* 23(2): 538–546. DOI:10.1109/JBHI.2018.2824327.

Khurshid S, Abdulla MS and Ghatak G (2025) Optimizing sharpe ratio: risk-adjusted decision-making in multi-armed bandits. *Mach. Learn.* 114(2). DOI:10.1007/s10994-024-06680-2. URL https://doi.org/10.1007/s10994-024-06680-2.

Kim B, Wattenberg M, Gilmer J, Cai C, Wexler J, Viegas F and sayres R (2018) Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In: Dy J and Krause A (eds.) *Proceedings of the 35th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, volume 80. PMLR, pp. 2668–2677. URL https://proceedings.mlr.press/v80/kim18d.html.

Koh PW, Nguyen T, Tang YS, Mussmann S, Pierson E, Kim B and Liang P (2020) Concept bottleneck models. In: *ICML*, *Proceedings of Machine Learning Research*, volume 119. PMLR, pp. 5338–5348.

Kókai G, Harmath L and Gyimóthy T (1997) Algorithmic debugging and testing of prolog programs. In: *LPE*. pp. 14–21.

Li Z, Huang J and Naik M (2023) Scallop: A language for neurosymbolic programming. *Proc. ACM Program. Lang.* 7(PLDI): 1463–1487.

Liang Y, Kumar N, Tang H, Weller A, Tenenbaum JB, Silver T, Henriques JF and Ellis K (2025) Visualpredicator: Learning abstract world models with neuro-symbolic predicates for robot planning. In: *ICLR*. OpenReview.net.

Liu Z, Luo P, Wang X and Tang X (2015) Deep learning face attributes in the wild. In: *Proceedings of International Conference on Computer Vision (ICCV)*.

Lundberg SM and Lee SI (2017) A unified approach to interpreting model predictions. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781510860964, p. 4768–4777.

Mahoney JJ and Mooney RJ (1993) Combining connectionist and symbolic learning to refine certainty factor rule bases. *Connection Science* 5(3-4): 339–364. DOI:10.1080/09540099308915704. URL https://doi.org/10.1080/09540099308915704.

Manhaeve R, Dumancic S, Kimmig A, Demeester T and Raedt LD (2021) Neural probabilistic logic programming in deepproblog. *Artif. Intell.* 298: 103504.

Meert W (2017) Pysdd. *Recent Trends in Knowledge Compilation (Dagstuhl Seminar 17381),Dagstuhl Reports* 7(9): 81.

Mooney RJ and Shavlik JW (2021) A recap of early work on theory and knowledge refinement. In: *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, *CEUR Workshop Proceedings*, volume 2846. CEUR-WS.org.

Muggleton SH (1987) Duce, an oracle-based approach to constructive induction. In: *IJCAI*. Morgan Kaufmann, pp. 287–292.

Muggleton SH and Buntine WL (1988) Machine invention of first order predicates by inverting resolution. In: *ML*. Morgan Kaufmann, pp. 339–352.

Möller M, Norlander A, Martires PZD and Raedt LD (2025) Neurosymbolic decision trees. URL https://arxiv.org/abs/2503.08762.

Ourston D and Mooney RJ (1990) Changing the rules: A comprehensive approach to theory refinement. In: *AAAI*. AAAI Press / The MIT Press, pp. 815–820.

Pav SE (2021) *The Sharpe Ratio: Statistics and Applications*. Chapman and Hall/CRC.

Pierce E (2021) Austin Housing Prices. https://www.kaggle.com/datasets/ericpierce/austinhousingprices. [Accessed 15-11-2024].

Ramachandran S and Mooney RJ (1996) Revising bayesian network parameters using backpropagation. In: *Proceedings of the International Conference on Neural Networks (ICNN-96), Special Session on Knowledge-Based Artificial Neural Networks*. Washington DC, pp. 82–87. URL http://www.cs.utexas.edu/users/ai-lab?ramachandran:icnn96kbai.

Rebbapragada U, Protopapas P, Brodley CE and Alcock CR (2009) Finding anomalous periodic time series: An application to catalogs of periodic variable stars. *CoRR* abs/0905.3428.

Ribeiro MT, Singh S and Guestrin C (2016) "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, San Francisco, CA, USA, August 13-17, 2016*. pp. 1135–1144.

Richards BL and Mooney RJ (1991) First-order theory revision. In: *ML*. Morgan Kaufmann, pp. 447–451.

Rudin C, Chen C, Chen Z, Huang H, Semenova L and Zhong C (2021) Interpretable machine learning: Fundamental principles and 10 grand challenges. *CoRR* abs/2103.11251.

Russell S and Norvig P (2020) *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson.

Schwarz G (1978) Estimating the Dimension of a Model. *The Annals of Statistics* 6(2): 461 – 464. DOI:10.1214/aos/1176344136. URL https://doi.org/10.1214/aos/1176344136.

Sha J, Shindo H, Delfosse Q, Kersting K and Dhami DS (2024) EXPIL: explanatory predicate invention for learning in games. *CoRR* abs/2406.06107.

Shrikumar A, Greenside P and Kundaje A (2017) Learning important features through propagating activation differences. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17. JMLR.org, p. 3145–3153.

Song Y, Demirdjian D and Davis R (2011) Tracking body and hands for gesture recognition: NATOPS aircraft handling signals database. In: *FG*. IEEE Computer Society, pp. 500–506.

Sundararajan M, Taly A and Yan Q (2017) Axiomatic attribution for deep networks. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*,

ICML'17. JMLR.org, p. 3319–3328.

Van Wesenbeeck D, Yurtman A, Meert W and Blockeel H (2024) Quantitative evaluation of motif sets in time series. URL https://arxiv.org/abs/2412.09346.

Wah C, Branson S, Welinder P, Perona P and Belongie S (2011) The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Winters T, Marra G, Manhaeve R and Raedt LD (2022) Deepstochlog: Neural stochastic logic programming. In: *AAAI*. AAAI Press, pp. 10090–10100.

Xi G, Tao C and Zhou Y (2021) Near-optimal MNL bandits under risk criteria. In: *AAAI*. AAAI Press, pp. 10397–10404.

Xian Y, Lampert CH, Schiele B and Akata Z (2019) Zero-shot learning - A comprehensive evaluation of the good, the bad and the ugly. *IEEE Trans. Pattern Anal. Mach. Intell.* 41(9): 2251–2265.

Yang Z, Ishay A and Lee J (2020) Neurasp: Embracing neural networks into answer set programming. In: *IJCAI*. ijcai.org, pp. 1755–1762.

A Number of possible modifications

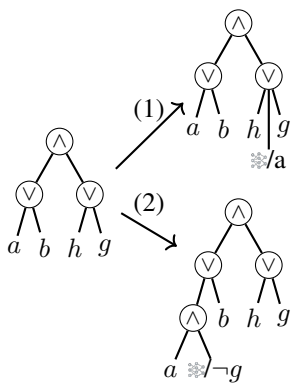
Given a circuit with l literals, n neural concepts and i inner nodes, we consider the following modifications:

- A removal modification can apply to every leaf node, thus giving $l + n$ possibilities.
- A new symbolic literal can be added to each leaf node, to the root node and as a child node to each inner node, thus resulting in $(l + n + i + 1)$ possibilities
- A new neural concept can be added to each leaf node, to the root node and as a child node to each inner node that does not already have a neural concept as a child, thus resulting in $(l + i - n + 1)$ possibilities

This sums up to $3 * l + 2 * i + n + 2$. In the edge cases that the circuit consists of a single neural concept or a single literal, the formula is $2 * l + 2 * i + 2$ as removal operations are not allowed in this case.

B Modification Selection Example

To clarify how the Sharpe ratio is used to select the next modification, consider the following toy example. This is based on the circuit used in Figure 1.



ID	a	b	h	g	Label
1	✓	✓	✓	✓	✓
2	✓	✗	✓	✗	✓
3	✗	✓	✗	✓	✓
4	✓	✓	✗	✗	✓
5	✓	✗	✗	✗	✓
6	✗	✓	✗	✗	✓
7	✓	✗	✗	✓	✗
8	✓	✗	✓	✗	✗
9	✗	✓	✗	✗	✗
10	✗	✗	✗	✗	✗

The initial circuit covers instances $\{1, 2, 3, 7, 8\}$, thus yielding:

$$TP = \{1, 2, 3\}, \quad FP = \{7, 8\}, \quad FN = \{4, 5, 6\}, \quad TN = \{9, 10\}.$$

This leads to an initial performance:

$$F_1 = \frac{2|TP|}{2|TP| + |FP| + |FN|} = \frac{6}{11} \approx 0.55.$$

We consider two candidate neural modifications:

1. inserting a neural concept in disjunction with the existing subcircuit $(h \vee g)$,
2. inserting a neural concept in conjunction with literal a.

Influence sets. The influence sets determine exactly which examples can be affected by each modification.

Modification 1 (generalisation). This replaces a node n with $n' = n \vee m$. The generalisation influence set is:

$$E^{(1)}_{\text{gen}} = \{4, 5, 6, 9\}.$$

These are precisely the examples currently classified as negative that would become positive if node n evaluated to true:

$$M[n \leftarrow True](x) = 1.$$

Examples outside this set, such as instance 10, remain unaffected.

Modification 2 (specialisation). This replaces a node n with $n' = n \wedge m$. The specialisation influence set is:

$$E^{(2)}_{\text{spec}} = \{2, 7, 8\}.$$

These are precisely the examples currently classified as positive that would become negative if node n evaluated to false:

$$M[n \leftarrow False](x) = 0.$$

Examples outside this set remain unaffected.

Best and worst case performance. The best and worst cases correspond to probability functions acting optimally or adversarially on the corresponding influence set, while leaving all other predictions unchanged.

Modification 1 (generalisation).

Only examples in $E^{(1)}_{\text{gen}}$ can change.

Best case.

The optimal probability function satisfies: $p^*(x) = \begin{cases} 1 & x \in \{4, 5, 6\} \\ 0 & x = 9 \end{cases}$

This yields: $TP = \{1, 2, 3, 4, 5, 6\}, \quad FP = \{7, 8\}, \quad FN = \emptyset.$

Thus, $B_1 = \frac{12}{12+2+0} = \frac{12}{14} \approx 0.86.$

Worst case.

The adversarial probability function satisfies:$p_\perp(x) = \begin{cases} 1 & x = 9 \\ 0 & x \in \{4, 5, 6\} \end{cases}$

This yields: $TP = \{1, 2, 3\}, \quad FP = \{7, 8, 9\}, \quad FN = \{4, 5, 6\}.$

Thus, $W_1 = \frac{6}{6+3+3} = \frac{6}{12} = 0.5.$

Modification 2 (specialisation).

Only examples in $E^{(2)}_{\text{spec}}$ can change.

Best case.

The optimal probability function satisfies: $p^*(x) = \begin{cases} 0 & x \in \{7,8\} \\ 1 & x = 2 \end{cases}$

This yields: $TP = \{1,2,3\}, \quad FP = \emptyset, \quad FN = \{4,5,6\}$.

Thus, $B_2 = \frac{6}{6+0+3} = \frac{6}{9} \approx 0.66$.

Worst case.

The adversarial probability function satisfies: $p_\perp(x) = \begin{cases} 0 & x = 2 \\ 1 & x \in \{7,8\} \end{cases}$

This yields: $TP = \{1,3\}, \quad FP = \{7,8\}, \quad FN = \{2,4,5,6\}$.

Thus, $W_2 = \frac{4}{4+2+4} = \frac{4}{10} = 0.4$.

Risk-free modification. To calculate the risk-free rate for the Sharpe ratio, we measure the performance of all symbolic operations. In this example, we consider all removal operations and two operations adding a symbolic literal in the same places as the neural concepts, with the decision stump learned on the influence sets from above:

Removing a from the circuit results in $\mathrm{F}_1 = \frac{4}{4+0+4} = \frac{4}{8} = 0.5$
Removing b from the circuit results in $\mathrm{F}_1 = \frac{4}{4+2+4} = \frac{4}{10} = 0.4$
Removing h from the circuit results in $\mathrm{F}_1 = \frac{4}{4+1+4} = \frac{4}{9} \approx 0.44$
Removing s from the circuit results in $\mathrm{F}_1 = \frac{4}{4+1+4} = \frac{4}{9} \approx 0.44$
Adding a literal (a) in disjunction with $h \vee g$ results in $\mathrm{F}_1 = \frac{10}{10+2+1} = \frac{10}{13} \approx 0.77$
Adding a literal $(\neg g)$ in conjunction with a results in $\mathrm{F}_1 = \frac{6}{6+1+3} = \frac{6}{10} = 0.6$
Performing no modification: $f_1 = 0.55$

The best risk-free modification $R_f = 0.77$ is thus adding a symbolic concept in disjunction as in modification 1.

Calculating the Sharpe ratio. Say that we do not have prior information over the performance of the neural network, thus setting $\alpha = 0.5$. The Sharpe ratios of both modifications are then calculated as shown in equation (4):

$$S_1 = \frac{\alpha B_1 + (1-\alpha)W_1 - R_f}{\sqrt{\alpha((1-\alpha)B_1 - (1-\alpha)W_1)^2 + (1-\alpha)(\alpha W_1 - \alpha B_1)^2}} = -0.44$$

In similar fashion, we calculate $S_2 = -1.77$, thus showing that applying the risk-free symbolic modification would result in the best estimated performance gain. In the case that there is prior information that the neural network performs better than average, $\alpha = 0.8$, the Sharpe ratios change. The first neural modification would then have $S_1 = 0.19$, the second would have $S_2 = -1.46$, thus showing that in this case applying the first neural modification has a higher estimated gain than performing the risk-free symbolic modification in the same place.

C Ablation Studies

The ablation studies were conducted using the experimental setup with structural information.

C.1 Effectiveness of the Selection Heuristic

This experiment studies the effectiveness of dynamically updating α, compared to the static alternatives as described in Section 4.2. The results, shown in Figure 5 and Table 2, demonstrate that our proposed variant of the Sharpe ratio with a dynamic α outperforms the alternative approaches. By dynamically adjusting α, and thereby updating the variance and expected reward of the neural concepts, the Sharpe ratio based heuristic effectively captures performance trends across different performance scenarios.

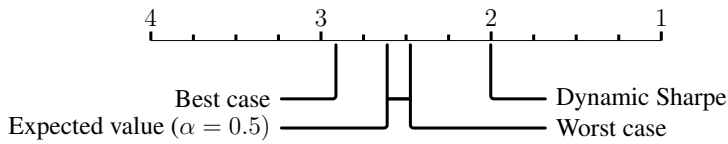


Figure 5. Critical difference diagram, identifying the dynamic Sharpe ratio as the best heuristic. Closer to 1 is better.

Table 2. F_1-score results of ablation study 1, which studies the used selection heuristic. The dynamic Sharpe ratio outperforms the static approaches.

Datasets	Sharpe dynamic	Worst case	Sharpe $\alpha = 0.5$	Best case
ATH	0.87±0.11	0.87±0.11	0.87±0.11	0.86±0.11
AWA2	0.87±0.09	0.84±0.10	0.84±0.10	0.86±0.09
CelebA	0.89±0.11	0.87±0.11	0.88±0.11	0.88±0.10
CUB	0.89±0.09	0.87±0.10	0.88±0.10	0.89±0.10
derm7pt	0.84±0.12	0.82±0.14	0.83±0.14	0.82±0.12
FMA	0.83±0.11	0.82±0.11	0.63±0.25	0.51±0.19
motifs	0.83±0.13	0.79±0.14	0.76±0.17	0.57±0.23
Pascal VOC	0.89±0.10	0.85±0.11	0.87±0.11	0.88±0.09
StarLightCurves	0.84±0.11	0.84±0.11	0.82±0.13	0.61±0.27

C.2 Freezing Existing Neural Concepts

When adding a neural concept, the underlying neural network is trained using indirect supervision while freezing the parameters of the previously trained neural networks. Table 3 shows the results of running NeTheR with only neural modifications allowed, in the cases that the result was modified with at least two neural concepts. It is clear that this choice, of freezing previously learned parameters does not significantly affect classification performance. In contrast to runtime, where it enhances computational

efficiency, as the number of trainable parameters is reduced in subsequent iterations, thereby decreasing the computational load and memory requirements for optimization.

Table 3. Results of ablation study 2, which studies the effect of freezing existing learned neural concepts. The results show that there are minor differences in F_1-score, but major differences in runtime.

Datasets	**Freeze**		**No Freeze**	
	F_1	time (s)	F_1	time (s)
ATH	0.86 ± 0.12	2579	0.87 ± 0.13	3080
AWA2	0.87 ± 0.10	784	0.86 ± 0.10	850
CelebA	0.95 ± 0.03	1084	0.95 ± 0.03	1351
CUBB	0.90 ± 0.10	2150	0.90 ± 0.10	2331
derm7pt	0.86 ± 0.13	303	0.87 ± 0.11	322
FMA	0.81 ± 0.17	718	0.78 ± 0.21	863
motifs	0.66 ± 0.28	883	0.62 ± 0.30	1196
Pascal VOC	0.89 ± 0.08	3637	0.89 ± 0.08	3913
StarLightCurves	0.86 ± 0.18	2727	0.85 ± 0.19	4468

C.3 Data-efficiency

We evaluate the data-efficiency by varying the size of the training dataset, for all datasets. Based on the previous performance results, we excluded *And Or* and *Or And* from the comparison because they are worse versions of NeTheR. The results of the data-efficiency experiment are shown in Figure 6 and Table 4. Both NeTheR and *CMR* perform much better in low data regimes because they can rely on M'. Compared to *CMR*, NeTheR shows slightly better but similar performance.

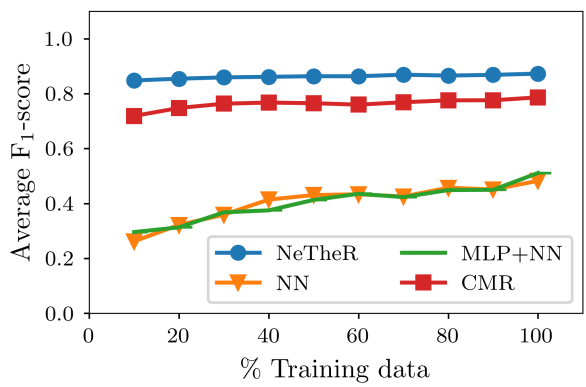


Figure 6. NeTheR is robust to limited training data.

Table 4. Numerical results showing the average F_1-score over all datasets per fraction (%) of available training data. It is clear that both NeTheR and CMR are robust against limited training data.

Methods	F_1-score per fraction of available data									
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
NeTheR	0.85	0.85	0.86	0.86	0.86	0.86	0.87	0.87	0.87	0.88
CMR	0.72	0.75	0.76	0.77	0.77	0.77	0.77	0.78	0.78	0.79
NN	0.26	0.32	0.36	0.41	0.43	0.43	0.43	0.46	0.46	0.48
MLP+NN	0.30	0.31	0.37	0.37	0.41	0.44	0.44	0.45	0.45	0.51

D Experiment details

We consider multiple benchmark datasets, the details of which we list below. We also clarify the generation of ground truth theories M^* and imperfect theories M'.

D.1 Benchmark Datasets

Austin Texas Housing (ATH) (Pierce 2021). All images were resized to 64×64 pixels, and irrelevant attributes (zpid, description, streetAddress, zipcode, latitude, longitude, latest_saledate) were removed to streamline the dataset for the task.

AWA2 (Xian et al. 2019). All images were resized to 64×64 pixels and used as subsymbolic data. The provided predicates associated with each class were utilized as symbolic data.

CelebA (Liu et al. 2015). We used the first 10,000 images in this dataset, all resized to 64x64 pixels. The attributes related to each picture were used as symbolic data.

CUB (Wah et al. 2011). All images were resized to 64×64 pixels and used as subsymbolic data. The provided attributes associated with each picture that had $certainty \in \{guessing, probably, definitely\}$ were utilized as symbolic data.

derm7pt (Kawahara et al. 2019). The clinical images were resized to 64x64 pixels and used as subsymbolic data. The given metadata and 7-point structure attributes were used as symbolic data.

FMA (Defferrard et al. 2017). For this dataset, we used the 'large' subset and discretized both the track data and Echonest data to obtain symbolic representations. The provided audio features were retained as subsymbolic data for analysis.

Motifs. This dataset comprises time series that were generated using TSMD-Bench (Van Wesenbeeck et al. 2024), constructed from the six core motifs from the NATOPS time series (Song et al. 2011). We extended these six core motifs to 18, by shifting their amplitude with +2 or -2. This simulates a common occurrence, for example, leakages cause a fixed amplitude to existing patterns. This is a challenging component in time series classification tasks, as instances have to be differentiated in both amplitude and shape. Then, using these 18 motifs, we generated 10,000 random instances by

concatenating 10 random motifs per instance. Each instance thus comprises 180 Boolean values as symbolic data, indicating whether a specific motif appears in a particular position. Figure 7 illustrates half of an instance with its symbolic representation.

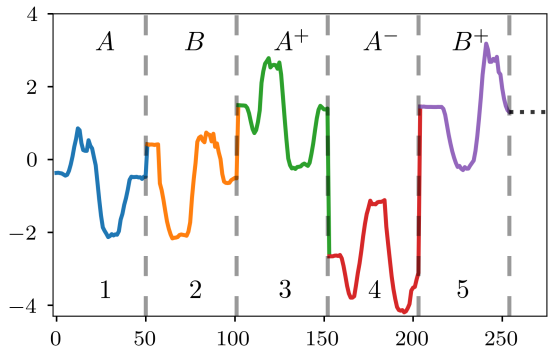


Figure 7. Visualization of five windows of one instance in the motifs dataset. X_i^+ and X_i^- both denote motif X in window i, with respectively a positive and negative shift in amplitude. The symbolic data of this instance would be $\{A_1{=}\top, B_2{=}\top, A_3^+{=}\top, A_4^-{=}\top, B_5^+{=}\top\}$, with the remaining variables $\bot$.

Pascal VOC (Everingham et al. 2010) All images were resized to 64×64 pixels and used as subsymbolic data. We used the presence of an object in the image as symbolic data.

StarLightCurves (Rebbapragada et al. 2009). Each instance in this time series dataset was divided into three equal windows. Within each window, the following features were identified: large peak, deep trough, low variability, high variability, sinusoidal shape, high average and large difference between min and max. Such features can be computed automatically and are often used in deployed systems (Christ et al. 2018). Each feature-window combination was represented as a Boolean variable to mimic expert detection of anomalies or behaviours based on patterns in specific parts of the time series.

D.2 Generating M and M'.*

For each dataset, we generated 24 logic circuits M^* and corresponding imperfect circuits M'. The construction process for M^* is top-down, starting from a disjunctive root node and iteratively refining the lower nodes. Each branch of the root disjunction contains nested conjunction nodes, and the leaf nodes are atomic variable comparisons using categorical or continuous columns from the data (e.g., $age < 30$ or $temperature = High$). Each node is generated with a degree of randomness, becoming either a nested subtree or a leaf node. In case of the latter, a random valid option (attribute, comparison operator and value) is chosen based on the dataset. The structurally informed imperfect theory M' is obtained from M^* by selecting a set of random attributes that are present

within M^*, and by removing all leaf nodes using those variables. If the resulting M' has an F_1-score that is not between 0.6 and 0.9, the process for generating an M^* and corresponding M' starts over. This resulted in M^* circuits that range from 3 to 47 nodes with a mean and standard deviation of 14 ± 9 nodes, with corresponding M' circuits ranging from 1 to 39 nodes and a mean and standard deviation of 8 ± 8 nodes. To learn the decision trees for the experiment without structural information, we use the training data and alter the parameters of the sklearn decision tree learner until the tree has an F_1-score smaller than 0.9.

E Experiment On Existing Labels

The experiments in Section 5 are conducted on the generated target model M^*. In this section we again repeat the non-informed experiment, but now using the existing labels of the dataset. We focus on derm7pt as it is a binary classification task, inferring the presence of melanoma. Instead of using the multitask approach that is proposed in the original paper Kawahara et al. (2019), we learned a classifier (ResNet-50) for each of the 7-point structures using both the dermoscopic and clinical images.

After that, we use the predicted values for the 7-point structures to construct the 7-pt score, which is combined with the metadata to serve as the data to learn a decision tree on. This decision tree can then be revised using NeTheR with both the dermoscopic and clinical images as subsymbolic data. We include as reference the models that were specifically designed for this task, and that were trained on unbalanced datasets: the 7pt-Unbalanced and Direct-unbalanced results that can be found in the benchmark paper (Kawahara et al. 2019). Table 5 summarizes the results, showing that the learned decision tree already outperforms the originally proposed approaches, and that NeTheR further improves performance.

Table 5. AUROC scores on the melanoma inferring task of the derm7pt dataset.

Methods	AUROC-score
Kawahara et al. (7pt-Unbalanced)	76.8
Kawahara et al. (Direct-Unbalanced)	83.2
Learned Decision Tree (M')	85.0
NeTheR	86.9

F Hardware

The experiments are performed on a machine running Ubuntu 20.04.6 LTS with 128 GB memory, an Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz CPU and a NVIDIA GeForce GTX 1080 Ti GPU.

G Detailed experimental results

Table 6 and 7 summarize the numerical results of the structurally informed and uninformed experiment respectively, by providing for each dataset the mean improvement of F_1-score over the initial model M'. The F_1-score for M' itself is provided in Table 8.

Table 6. Comparison with other methods, demonstrating the effectiveness of NeTheR when M' is structurally informed. The values are the mean differences in F_1-score between the learned model M and the input model M', i.e., $F_1(M(x), M^*(x)) - F_1(M'(x), M^*(x))$, aggregated per dataset. Higher is better. The F_1-scores for the input model M' are shown in Table 8.

Datasets	Methods					
	NeTheR	And Or	Or And	NN	MLP+NN	CMR
ATH	**+0.10±0.1**	-0.13±0.2	-0.15±0.2	-0.32±0.2	-0.32±0.3	+0.00±0.1
AWA2	+0.10±0.1	**+0.13±0.1**	+0.12±0.1	+0.04±0.1	-0.07±0.1	+0.10±0.1
CelebA	**+0.11±0.1**	+0.04±0.1	+0.04±0.1	-0.15±0.2	-0.25±0.3	+0.00±0.1
CUB	**+0.10±0.1**	-0.02±0.1	-0.02±0.1	-0.28±0.3	-0.38±0.4	-0.01±0.1
derm7pt	**+0.07±0.1**	-0.26±0.3	-0.31±0.3	-0.58±0.3	-0.56±0.3	-0.06±0.1
FMA	**+0.08±0.1**	+0.04±0.1	+0.04±0.1	-0.42±0.2	-0.05±0.2	-0.05±0.1
motifs	**+0.07±0.1**	-0.09±0.2	-0.13±0.3	-0.61±0.2	-0.63±0.2	-0.06±0.2
Pascal VOC	+0.15±0.1	**+0.17±0.1**	**+0.17±0.1**	+0.06±0.2	+0.07±0.2	+0.14±0.1
StarLightCurves	+0.08±0.1	-0.15±0.4	+0.04±0.3	-0.53±0.3	-0.10±0.4	**+0.12±0.1**

Table 7. Comparison with other methods, demonstrating the effectiveness of NeTheR when M' is structurally uninformed. The values are the mean differences in F_1-score between the learned model M and the input decision tree M', i.e., $F_1(M(x), M^*(x)) - F_1(M'(x), M^*(x))$, aggregated per dataset. Higher is better. The F_1-scores for the input model M' are shown in Table 8.

Datasets	Methods					
	NeTheR	And Or	Or And	NN	MLP+NN	CMR
ATH	**+0.12±0.1**	-0.08±0.2	-0.03±0.1	-0.4±0.3	-0.38±0.2	-0.28±0.2
AWA2	**+0.08±0.1**	+0.02±0.1	+0.02±0.1	-0.02±0.1	-0.13±0.1	+0.06±0.1
CelebA	**+0.07±0.1**	+0.05±0.1	+0.04±0.1	-0.13±0.2	-0.25±0.3	-0.05±0.2
CUB	**+0.07±0.1**	-0.02±0.1	+0.02±0.1	-0.26±0.3	-0.36±0.4	-0.23±0.2
derm7pt	**+0.05±0.1**	-0.02±0.1	-0.02±0.1	-0.64±0.3	-0.64±0.3	-0.14±0.1
FMA	-0.03±0.2	**+0.01±0.1**	**+0.01±0.1**	-0.48±0.2	-0.13±0.3	-0.05±0.1
motifs	-0.05±0.3	-0.19±0.4	-0.18±0.4	-0.47±0.3	-0.54±0.3	**+0.00±0.3**
Pascal VOC	**+0.16±0.1**	+0.11±0.1	+0.10±0.1	+0.06±0.2	+0.07±0.2	**+0.16±0.1**
StarLightCurves	-0.10±0.3	+0.06±0.2	+0.01±0.3	-0.51±0.2	-0.12±0.3	**+0.12±0.1**

Table 8. Average F_1-score of input model M' for each dataset. M'_1 is the model used for the structurally informed experiment, while M'_2 is the learned imperfect decision tree that is used for the uninformed experiment.

Dataset	$F_1(M'_1(x), M^*(x))$	$F_1(M'_2(x), M^*(x))$
ATH	0.76±0.1	0.80±0.1
AWA2	0.77±0.1	0.84±0.1
CelebA	0.78±0.1	0.74±0.1
CUB	0.80±0.1	0.78±0.2
derm7pt	0.77±0.1	0.81±0.1
FMA	0.74±0.1	0.78±0.1
motifs	0.75±0.1	0.64±0.2
Pascal VOC	0.74±0.1	0.70±0.2
StarLightCurves	0.77±0.1	0.75±0.1

H Qualitative examples

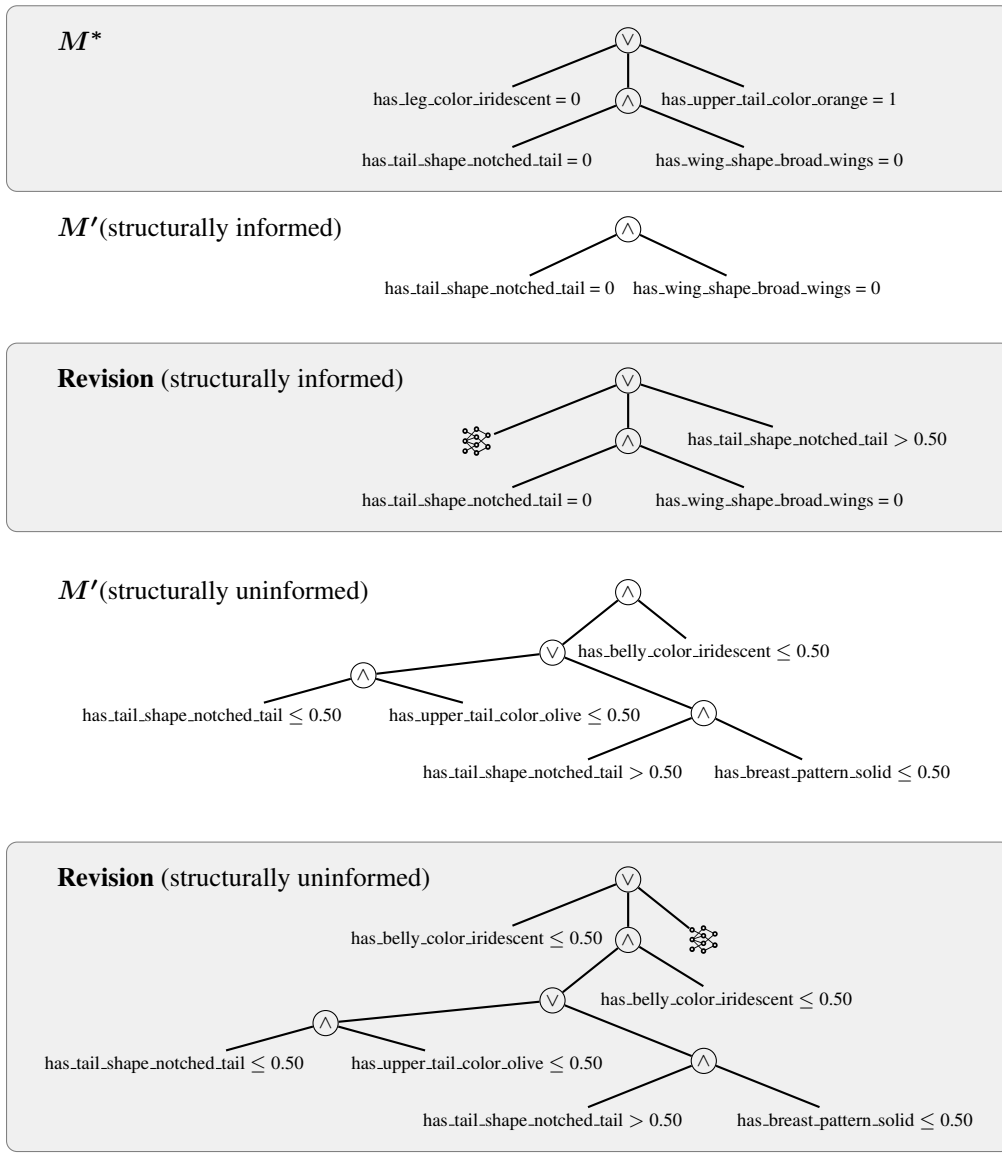


Figure 8. An example of an instance of the CUB dataset. The F_1-score for the structurally informed experiment increased from 0.78 for M' to 0.99 for the revised model of NeTheR that introduced a neural concept and a new symbolic literal. For the uninformed experiment, the learned decision tree reached an F_1-score of 0.89, while its revision using NeTheR outperforms with an F_1-score of 0.99.