Neural Reasoning Networks: Efficient Interpretable Neural Networks with Automatic Textual Explanations

Stephen Carrow[1], Kyle Erwin[1], Olga Vilenskaia[1], Parikshit Ram[1], Tim Klinger[1], Naweed Khan[1], Ndivhuwo Makondo[1], Alexander G. Gray[2]

[1]IBM,
[2]Centaur AI Institute
{steve.carrow, kyle.erwin, olga.vilenskaia, parikshit.ram, naweed.khan, ndivhuwo.makondo}@ibm.com,
tklinger@us.ibm.com, alexander.gray@centaurinstitute.org

Abstract

Recent advances in machine learning have led to a surge in adoption of neural networks for various tasks, but lack of interpretability remains an issue for many others in which an understanding of the features influencing the prediction is necessary to ensure fairness, safety, and legal compliance. In this paper we consider one class of such tasks, tabular dataset classification, and propose a novel neuro-symbolic architecture, Neural Reasoning Networks (NRN), that is scalable and generates logically sound textual explanations for its predictions. NRNs are connected layers of logical neurons that implement a form of real valued logic. A training algorithm (R-NRN) learns the weights of the network as usual using gradient descent optimization with backprop, but also learns the network structure itself using a bandit-based optimization. Both are implemented in an extension to PyTorch that takes full advantage of GPU scaling and batched training. Evaluation on a diverse set of 22 open-source datasets for tabular classification demonstrates performance (measured by ROC AUC) which improves over Multilayer Perceptron (MLP) and is statistically similar to other state-of-the-art approaches such as Random Forest, XGBoost and Gradient Boosted Trees, while offering 43% faster training and a more than 2 orders of magnitude reduction in the number of parameters required, on average. Furthermore, R-NRN explanations are shorter than the compared approaches while producing more accurate feature importance scores.

1 Introduction

Classifying tabular data has long been a fundamental task in ML/AI and recent advances in AI have led to increased adoption in various industries, but the complexity of these systems has made them black boxes that are difficult to understand and interpret (Adadi and Berrada 2018; Linardatos, Papastefanopoulos, and Kotsiantis 2021; Barredo Arrieta et al. 2020). This lack of transparency is a major obstacle to the adoption of AI in sensitive domains such as healthcare, and increasingly practitioners and researchers aim to explain black-box model predictions (Rudin 2019). A variety of tools have been designed towards that end (Guidotti et al. 2018), however popular methods such as SHAP (Lundberg and Lee 2017) and LIME (Ribeiro, Singh, and Guestrin 2016) have been shown to be susceptible to adversarial attacks (Yuan and Dasgupta 2023; Slack et al. 2020) and fail to effectively explain neural networks and generalized additive models (GAMs) (Carmichael and Scheirer 2023). As a result, these post-hoc explainers can produce misleading explanations, especially when users misunderstand or over trust them. Recent work even considers the use of black-box models to explain other black-box models, such as with a pre-trained LLM (Kroeger et al. 2023) or Diffusion model (Madaan and Bedathur 2023), further obfuscating the explanation generation process.

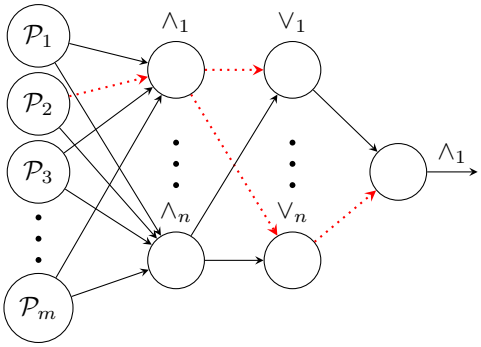

Figure 1: A two layer sparsely connected NRN for the R-NRN training algorithm with dropped edges in red.

One potential solution to address these drawbacks is to use inherently interpretable models, which can produce their own explanations that faithfully represent the model's computation (Rudin 2019). This approach is explored in, for example, Neural Additive Models (NAM) (Agarwal et al. 2021), and CoFrNet (CFN) (Puri et al. 2021), which have performance that is statistically similar to deep-tree-ensemble-based methods that are known to perform well for tabular classification.

However, inherently interpretable models are not all equal in their explanation quality, predictive accuracy, or practical utility such as training speed or parameter counts. For example, our experiments show that the best NAM and CFN models after hyper-parameter tuning have approximately 861K and 37K parameters respectively. In this paper, we introduce the NRN architecture and the R-NRN supervised classifi-

cation algorithm. NRNs use Weighted Lukasiewicz Logic, which forms the foundation of interpreting the resulting network after training. Each node is interpreted as either an "And" or "Or" operation, so the model itself is composed of logical operations over the input features. R-NRN, is constructed with sparsely connected logical nodes, as depicted in Figure 1, requires only 1K parameters and trains, on average, 43%[1] faster compared NAM and CFN, while its performance in terms of ROC AUC is statistically similar to RF, XGB, and GBT as summarized in Table 3.

Furthermore, while methods like NAM and CFN are interpretable, explanations take the form of directional feature importance showing only relative contribution from features, in contrast to our approach that produces a natural language description of the logical rule set for a prediction. This distinction is important in that the rule set produced contains not only direction of influence and relative importance for each rule, but also the conditions on a sample that must be satisfied, such as relational constraints of the form $f_1 > X$, for f_1 a real-valued feature, and X a learned boundary value. In addition, explanation sizes for methods based on feature importance, including NAM, CFN and those produced with SHAP, increase with the number of features in the data set while R-NRN explanation size is proportional to the size of the learned rules, which may have much fewer features. We show experimentally that R-NRN training results in NRN explanations that are 31% smaller on average than feature-importance-based methods for sample level explanations.

While the NRN learned using R-NRN may be rather complex, we achieve concise sample level explanations by simplifying verbose logic with straight forward rules, such as $AND(x > a; x > b) = x > \max(a, b)$, and we can ignore all disjunction branches which are not true for a sample leaving only the logic where all conditions are met. Since the sample under analysis will satisfy all the logic that remains following such transformations, the explanation can be simplified to a single conjunction. Details of explanation generation are discussed in **Section 3**.

In summary, we contribute the following in this work:

1. **Neural Reasoning Networks**: A neuro-symbolic AI architecture that uses a Modified Weighted Lukasiewicz Logic constructing interpretable networks that leverage AutoGrad, batched training, and GPU acceleration.
2. **Bandit Reinforced Neural Reasoning Network (R-NRN)**: A *supervised classification* algorithm for *tabular data* with predictive performance that is statistically similar to RF, XGB, and GBT and which produces compact sample level explanations of logical rules sets.
3. **Automatic explanation generation**: A novel algorithm that leverages the structure of NRNs, algebraic manipulations of Weighted Lukasiewicz Logic, and logical interpretation of NRN nodes to produce natural language explanations of the models' predictions.
4. **Rigorous evaluation**: An elaborate evaluation with various models and tabular datasets, highlighting the competitiveness of our proposed R-NRN. Relevant to the research on neural networks for tabular data, our results show that R-NRN, NAM, CFN, and DAN outperform various tabular NN with a significant margin and are on par with RF, XGB, and GBT, contrary to existing results.

[1]Average R-NRN trial time versus the mean of average trial times for NAM and CFN in Table 3.

The remainder of the paper presents the details of NRNs and R-NRN along with our experimental results and analysis of our algorithm. In **Section 2**, we review the related work in explainable machine learning and discuss limitations of the previously published techniques. In **Section 3**, we introduce Neural Reasoning Networks, detail the R-NRN algorithm for classification, and our explanation algorithm. In **Section 4**, we share the details of an extensive empirical study comparing the predictive performance of R-NRN to existing algorithms. In addition, we analyze R-NRN explanation quality. In **Section 5** we discuss limitations and conclude our work with future directions for this research.

2 Related Work

In this section we review related works and discuss how they are positioned within the landscape of AI/ML algorithms used for supervised tabular classification, as well as their relation to explainable AI (XAI).

We begin with post-hoc explanation of a trained AI/ML model, as this is a highly common (Rudin 2019) and flexible approach to XAI in practice. Application of post-hoc explainers like SHAP and LIME are traditionally used to explain *non-interpretable models*. Well-known instances of non-interpretable models are Random Forests (RF) (Breiman 2001), Gradient Boosted Trees (GBT) (Friedman 2001), eXtreme Gradient Boosting (XGB) (Chen and Guestrin 2016), and Multilayer Perceptron (MLP). More recently, research has focused on development of neural-methods for tabular data leveraging modern architectures like Transformer. A recent example is the FTTransformer (FTT) (Gorishniy et al. 2021). FTT is an adaptation of the Transformer architecture for tabular data that embeds all features (categorical and numeric) and applies a stack of Transformer layers to the embeddings. FTT outperforms other deep learning methods according to the authors.

Chen et al. proposed another non-interpretable neural method for learning from tabular data. They contribute a flexible neural component for tabular data called the Abstract Layer, which learned to explicitly group correlative input features and generate higher-level features for semantics abstraction (Chen et al. 2022). This approach is somewhat similar to self-attention in that it learns sparse feature selections. The authors also proposed Deep Abstract Networks (DANets) which use these Abstract Layers. The authors show that DANets are effective and have superior computational complexity compared to competitive methods.

Many more neural methods for tabular data exist and works like (Borisov et al. 2022) provide extensive surveys.

Another branch of research focuses on *inherently interpretable models*. These methods should themselves produce faithful explanations representing the model's computation (Rudin 2019). Such models produce explanations and allow for interpretation in a variety of ways. GAMs, one such class

	R-NRN	BRCG	DIF	NAM	CFN	FTT	DAN	MLP	RF	XGB	GBT
Logical	**Yes**	**Yes**	**Yes**	No	No	No	No	No	No	No	No
Linear Scoring	**Yes**	No*	No*	**Yes**	**Yes**	No*	No*	No*	No*	No*	No*
Case Reasoning	**Yes**	**Yes**	No	No	No	No	No	No	No	No	No
GPU Scaling	**Yes**	No	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	No	No
Uses AutoGrad	**Yes**	No	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	No	No
Batched Training	**Yes**	No	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	No	No

Table 1: Assessment of algorithms against (Rudin 2019) challenges for interpretability and assessment of scalability attributes AutoGrad and Batched Training. Methods with *No** listed for linear scoring can be estimated using SHAP.

of interpretable models, are a type of statistical model that extends the concept of generalized linear models (GLMs) by allowing for more flexible, non-parametric relationships between the target variable and the predictor variables (features) (Hastie 2017). Formally, GAMs are defined as

$$g(\mathbb{E}[y]) = \beta + f_1(x_1) + f_2(x_2) + \cdots + f_K(x_K) \quad (1)$$

where $\mathbf{x}$ is the input vector with K features, represented as $(x_1, x_2, ..., x_K)$, y is the target variable, g represents the link function, and each f_i is a smoothing function. Models in this class aim to produce explanations using the weights assigned to the linear combination. In the case of Neural Additive Models (NAMs), this is a combination of neural networks each focused on a single feature. NAM can also be explained at a global level by visualizing the shape functions learned by each neural network (Agarwal et al. 2021).

Scalable Interpretability via Polynomials (SPAM) (Dubey, Radenovic, and Mahajan 2022) are similar to NAM, except that they efficiently model higher order feature interactions with polynomials. While the authors show that this approach does improve AUC and RMSE on a variety of datasets, they only evaluate interpretability for the SPAM-linear that does not model those higher level feature interactions. Furthermore, the authors point out that such polynomial feature interactions are less interpretable.

Finally, Puri *et al*. proposed a new neural network architecture called Continued Fraction Network (CFN) which was inspired by continued fractions from number theory (Puri et al. 2021). This method is another approach to producing models with interpretable weights. The authors show that CFNs are efficient learners and because they represent linear combinations of features they can be interpreted in a similar manner to NAM, SPAM, and other GLMs.

Logic or rule based models are another common approach to creating interpretable models. These methods produce a rule set that describes the positive class for tabular classification problems. For example, the Boolean Rule Column Generation (BRCG) approach learns interpretable Boolean rules in disjunctive normal form (DNF) or conjunctive normal form (CNF) for classification (Dash, Gunluk, and Wei 2018). It uses column generation to efficiently search over an exponential number of candidate clauses without pre-mining or restrictions. The resulting model can be interpreted by examining the rule set induced during training.

Rule based models have also been combined with linear scoring, meaning scores or weights are assigned to rules or features based on their impact within in the model. For instance, Wei et al. proposed a new approach to building generalized linear models using rule-based features, also known as rule ensembles, for regression and probabilistic classification. The approach uses the same binary column generation technique as BRCG. The method is shown to obtain better accuracy-complexity trade-offs than existing rule ensemble algorithms and is competitive with less interpretable benchmark models (Wei et al. 2019). One interpreting the model can understand both the rules that lead to a prediction and the relative influence of those rules.

Another example of combining logic and weights is Logical Neural Networks (LNN) (Riegel et al. 2020), a framework that combines neural networks and Weighted Lukasiewicz Logic to score logical nodes with a linear weight. The framework can minimize logical contradiction, enabling it to handle inconsistent knowledge and make open-world assumptions. This approach is highly related to our work, however unlike our work, the authors do not propose a *supervised classification* method to solve traditional ML problems on tabular data, and leverage First Order Logic rather than the Propositional Logic of NRNs.

Not all logic or rules based models maintain interpretability, however. Deep Differentiable Logic Gate Networks (DIF), for example, learn a real-valued representation of logic gates that can then be discretized to a traditional output logic gate network for fast inference (Petersen et al. 2022). DIF classifies samples by counting gates that predict each class. These networks may be inspected post training to examine the learned logic, however, the authors do not propose a method to interpret the network, which may become arbitrarily large and complex making it transparent in some sense, but un-interpretable under real world conditions.

Our work is aimed at developing a logic based model that uses linear weighting such that the importance of each rule can be understood. In addition, our approach should be not only transparent, but also interpretable, such that it can be used in practice. Rudin clearly articulates three challenges in developing interpretable models (Rudin 2019) that align closely to our work. They are described as *logical conditions* (Challenge 1), *linear modeling* (Challenge 2), and *case-based reasoning* (Challenge 3). Logical models, such as decision trees and rule lists, employ logical conditions like "If-Then" statements, "Or", and "And" to make predictions or classify data. These models consist of statements that are combined to form a logical rule, enabling transparent decision-making. Linear modeling, on the other hand, involves assigning scores or weights to features based on their relevance to the model's predictions. This scoring system

helps identify the most important features contributing to the model's outcomes. Lastly, case-based reasoning involves explaining the decision-making process for each sample by highlighting the relevant parts of the input data. This approach provides a clear understanding of how the model arrives at its conclusions (Rudin 2019). Please refer to Table 1 to see how the aforementioned methods measure up to these challenges and an assessment of their scalability; our proposed R-NRN address all three challenges.

3 Neural Reasoning Networks

In this section, we motivate our *architecture*, Neural Reasoning Network (NRN), and the benefits of our approach to building inherently interpretable AI systems. We then show how NRN is used to instantiate a *supervised classification* algorithm for learning on *tabular data* and close by detailing how this trained network is explained.

Neuro-symbolic AI techniques, such as (Riegel et al. 2020), have demonstrated capable of producing interpretable models that use logic. LNN showed excellent results for Inductive Logic Programming (Sen et al. 2022), Entity Linking (Jiang et al. 2021), KBQA (Kapanipathi et al. 2020), and reinforcement learning (Kimura et al. 2021). However, LNN lacks a method for logic induction from data, and cannot leverage GPU acceleration. We therefore leverage Weighted Lukasiewicz Logic activation functions introduced in (Riegel et al. 2020) to construct a Neural Reasoning Network with "blocks" of Modified Weighted Lukasiewicz Logic, which are implemented as PyTorch modules, to produce an interpretable neuro-symbolic network of connected layers that is differentiable, leverages AutoGrad, GPU acceleration, and distributed GPU computation.

NRN Intuitively, a NRN is similar to a traditional Neural Network (NN), except that each node in the network is interpreted as either an *"And"* (*Conjunction*), or *"Or"* (*Disjunction*) operation as shown in Figure 1, which depicts a two-layer Neural Reasoning Network with alternating Conjunction and Disjunction blocks, although different architectures can be easily developed. This interpretation is achieved by representing each node with Weighted Logic that models those operations. Weighted Logic, including Weighted Lukasiewicz Logic introduced in (Riegel et al. 2020), are a form of fuzzy real valued logic that produce a confidence estimation that the given logical operation is *True*.

Formally, a NRN is defined as a composition of a Modified Weighted Lukasiewicz Logic, $f := g \circ f_k \circ \ldots f_2 \circ f_1$, that maps n real valued inputs to p real valued outputs, $f : \mathcal{R}^n \rightarrow \mathcal{R}^p$. Both $\mathcal{R}^n$ and $\mathcal{R}^p$ are in the range $[0, 1]$. Each of the functions g and $f_1 \ldots f_k$ can take on a tensor form of one of the two Modified Weighted Lukasiewicz Logic functions corresponding to Conjunction in Expression 2, i.e.

$$f\left(\beta - \sum_j |w_j|\left[1 - \left[m_j x_j + (1 - m_j)(1 - x_j)\right]\right]\right), \quad (2)$$

or Disjunction in Expression 3, i.e.

$$f\left(1 - \beta + \sum_j |w_j|\left[m_j x_j + (1 - m_j)(1 - x_j)\right]\right), \quad (3)$$

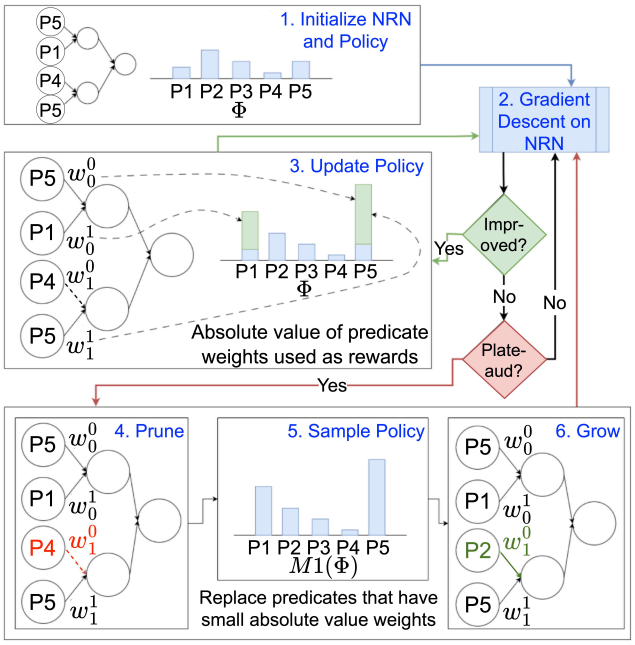

Figure 2: Depiction of iteration using R-NRN logic induction algorithm.

where β represents a bias, which is typically fixed and set to 1 to aid interpretation of the network, and for which $\beta \geq 0$. j corresponds to the number of inputs to the logical node, $w \in \mathcal{R}$ is a vector of weights for each input, and $x \in [0, 1]$ is a vector of inputs. Finally, f clamps the result in the range $[0, 1]$ to produce a *truth value*.

The key distinction from (Riegel et al. 2020) is $m_j = \mathbf{I}_{w_j > 0}$, which is an indicator identifying if the weight $w_j \in \mathcal{R}$ is greater than 0 and used as a mask in the operation. Negative weights correspond to negation of an input, and the indicator vector acts by flipping those input values to satisfy the negation enabling the network to learn negations without any additional parameters. We detail how this approach enables GPU scaling in (Carrow et al. 2024) Appendix N.

A "predicate", $\mathcal{P}$, is the third node type in an NRN and is a mapping between a feature in an input vector $\mathbf{x}$ and a predicate node; the leaves in an NRN. A single feature may be mapped to multiple predicate nodes. In addition, predicate nodes contain the feature's description used during explanation of a trained NRN. Therefore, *one role of predicates is to aid in interpretability of NRN explanations*.

Bandit Reinforced Neural Reasoning Network (R-NRN) We now proceed to describe our supervised classification algorithm, R-NRN, that induces logic used for prediction from tabular data using NRN. In addition, we will share a description and intuition behind important hyper-parameters and heuristic methods used in the R-NRN algorithm.

R-NRN employs an iterative training algorithm that leverages Gradient Descent and a Multi-Armed Bandit that we depict in Figure 2 and describe in detail in the (Carrow et al. 2024) Appendix in Algorithm 8. Model-1, $M_1(\Phi)$, is a system that generates a proposed NRN, thereby constructing

Model-2, $M_2(\Theta)$. Model-2 is subsequently used to predict a target, or multiple targets, and it's parameters, Θ, are updated using Gradient Descent. Finally, if a pre-determined update criteria are met, then a reward function, R, computes a reward proportional to the desired performance of Model-2 on the training data and the parameters of Model-1, Φ, are updated with respect to this reward, r, using an UpdatePolicy method for Model-1 . The algorithm is trained for a set number of epochs, T, or is stopped using other desired criteria. Next, we describe each step in Figure 2.

In *Step 1* of R-NRN training, we initialize Model-1, $M_1(\Phi)$, as a Bayesian UCB Multi-Armed Bandit (BMAB) whose policy is the Maximal Information Coefficient score between features and targets. The NRN, Model-2, is initialized with a fixed structure resembling that of Figure 1 with blocks of interpretable nodes that alternate between conjunction and disjunction, the ordering of which is controlled by a hyper-parameter. Much like a traditional NN, one predefines the number of logical nodes in each layer, the number of inputs to the logical nodes in each layer, and the number of layers. On initialization, inputs to each logical node are randomly selected from the previous layer.

Following initialization, *Step 2* begins the training procedure and updates weights on each edge in the network using Gradient Descent. The training proceeds in a typical fashion computing loss over mini-batches until an epoch completes.

On each iteration the loss is evaluated, and if it's improved, we complete *Step 3* in which the BMAB policy updates with a reward appropriate for generating an improved Model-2 as training continues. We experiment with several reward functions for the BMAB. The *class* strategy, depicted in Figure 2, corresponds to the sum of the learned weights for each feature. While fast to compute, this strategy determines rewards at a feature level, rather than a logic level. Two additional reward strategies detailed in the (Carrow et al. 2024) Appendix S enable rewards at a logic level.

The decision of when to prune the existing logic and generate new logic (*Steps 4, 5, and 6*) is a central part of the R-NRN algorithm and is triggered by plateaus in model performance. As training continues, the learned logic should be closer to optimal and the structure may require less frequent updates. We therefore control the NRN structure update process using three hyper-parameters: κ, τ, and ι. The hyper-parameter, κ, sets the number of epochs for which improvement has plateaued that will trigger the model to prune the existing logic. The next two hyper-parameters act together to increase the value of κ, which is helpful to reduce the frequency of pruning as training progresses. ι indicates the number of additional epochs to add to κ, once τ is reached.

Once the performance plateau is reached, the structure of the NRN is updated with *Steps 4, 5, and 6*. In this update, R-NRN performs a logic pruning and generation process in which predicates with weights above a threshold are kept and those with weights below that threshold are pruned (*Step 4*). New features are sampled from the BMAB policy, modified by δ to decrease the likelihood of sampling un-pruned features (*Step 5*). Finally, those sampled features are placed into the predicate nodes that were pruned (*Step 6*). We note that multiple predicate nodes may reference the same feature

Algorithm 1: Produce un-simplified explanation

1: initialize $l = NRN.root$, $t = NRN(x)$, $n =$ False
2: **Procedure:** $expl(l,\ t,\ n)$:
3: initialize $\epsilon =$ ""
4: **for** j, C **in** $enumerate(l.children)$
5: **if** n
6: $t = 1 - t$
7: $v_c \leftarrow ComputeRequiredChildValue(C, t)$
8: **if** n
9: $M = C.value \times |l.weights_j| < v_c$
10: **else**
11: $M = C.value \times |l.weights_j| \geq v_c$
12: **if** M
13: **if** $HasChildren(C)$
14: **if** $l.weights_j < 0$
15: $n =$ **not** n
16: $\epsilon \leftarrow Cat(\epsilon, AddN(expl(C,\ v_c,\ n), l, n, j))$
17: **else**
18: **if** $l.weights_j < 0$
19: $v_c = 1 - v_c$
20: $\epsilon_{\mathcal{P}} \leftarrow AddN(Cat(C.name, \geq, v_c), l, n, j)$
21: $\epsilon \leftarrow Cat(\epsilon, \epsilon_{\mathcal{P}})$
22: **if** $\epsilon <>$ ""
23: **return** $Cat(l.logic\_type, "(", \epsilon, ")")$
24: **return** ""

but in separate Conjunction or Disjunction nodes. Additionally, if the newly generated predicate references a feature already referenced in the un-pruned predicate set, the weights for that predicate are re-initialized with a sign opposite to the average weight across all logics with un-pruned predicates referencing that feature. The effect of flipping the weight is to introduce the negated version of the un-pruned predicate, enabling the Model-2 NRN to represent $l \leq x \leq u$ and $x \leq l \ \& \ u \leq x$. The details of this procedure are in the (Carrow et al. 2024) Appendix in Algorithm 9.

Once trained, the learned structure and weights reveal the induced logic and can be inspected with our explanation algorithm, described next.

Explanation generation As mentioned, each node in an NRN is interpretable as an "And" or "Or", and we examine a *trained* network to generate an explanation of the model. To interpret the trained network, we implement an algorithm that performs a depth first traversal of an NRNs nodes to produce a sample level explanation. The recursive explanation algorithm consists of evaluating the Weighted Lukasiewicz Logic activation functions of each node to identify the value for each input to a logical node that is required for that input to produce the nodes *truth value* given all other input values, as in Algorithm 1, line 7; and surfacing only those nodes for which the specific sample value meets this threshold, lines 8-11. In Algorithm 1, l is the root node of an NRN, t is the output from a trained NRN, and n is a boolean assisting with handling negations during traversal. (Carrow et al. 2024) Appendix I includes details of procedures for $ComputeRequiredChildValue$, $AddN$ which adds "NOT" to the explanation string if required, Cat which

Positive Class	Negative Class
The average number of bedrooms was between 1.053 and 1.114	The average number of bedrooms was between 1.053 and 1.114
The average number of rooms was greater than 6.986	The average number of rooms was less than or equal to 4.201
The average occupancy was between 3.001 and 7.886	The average occupancy was between 3.001 and 7.886
The house age was between 16.5 and 26.5	The house age was greater than 26.5
The latitude was less than or equal to 34.45	The latitude was less than or equal to 33.945
The longitude was between -118.035 and -117.635	The longitude was between -118.295 and -118.035
The median income was between 5.349 and 6.091	The median income was between 2.192 and 2.578
The population was less than or equal to 656	The population was greater than 656

Table 2: Explanations from trained R-NRN on randomly selected samples from the `California` dataset. Each explanation shows 100% of the simplified model explanation, which is a single conjunction of the rules that produce the sample's prediction.

performs string concatenation, and $expl$, which produces a string form of the explanation.

Explanation simplification The logic extracted with Algorithm 1 may be complex and we therefore perform a simplification process, which is of critical importance to ensure NRNs are interpretable in practice. The simplification process implies applying logical rules to shorten and simplify the explanations. Our goal is to make the explanations as simple as possible while preserving all the logic of the model. As in Algorithm 2, we apply several recursive sub-algorithms until the following conditions are met 1) standardize the logical structure to conjunctive (or disjunctive) normal form that only contains negations at the leaves – represented by "predicates" (line 2) by traversing the explanation, represented as a tree, and applying logical rules to push negations down to the leaves level, 2) identify propositional equivalences and apply carefully crafted simplification rules to remove unnecessary logic such as collapsing sequences of consecutive conjunctions or disjunctions to a single operation, and removing redundant predicates (lines 3, 4, 5, 6), and 3) aggregate all the conditions which are true for the sample into a single conjunction while ignoring all the logic parts which do not hold for the sample (lines 7, 8). The resulting sample explanation is a single conjunction of rules as shown in Table 2. Additional details of the simplification functions and rules are located in the (Carrow et al. 2024) Appendix section J and additional example outputs are shown in section G.

4 Results

In this section we present the results of an extensive empirical study comparing R-NRN to tree-based algorithms, traditional deep learning, and specialized deep learning models designed for tabular data. In addition, we present example explanations from R-NRN and evaluate their quality.

Algorithm 2: Explanation simplification

1: initialize $n = Explanation.root$
2: $push\_negations\_down(n)$ {(Rules # 1, # 4, # 5, # 6)}
3: $collapse\_repeated\_operands(n)$ {(Rules # 7, # 8)}
4: $remove\_redundant\_predicates(n)$ {(Rules # 2, # 3)}
5: $collapse\_single\_operands(n)$ {(Rule # 9)}
6: $remove\_redundant\_predicates(n)$ {(Rules # 2, # 3)}
7: $collapse\_sample\_explanation(n)$
8: $remove\_redundant\_predicates(n)$ {(Rules # 2, # 3)}

Predictive performance To evaluate our method, we train R-NRN, as well as one or more representative models from each class of models described in the related work, on the benchmark for tabular datasets proposed by (Grinsztajn, Oyallon, and Varoquaux 2022) as detailed in (Carrow et al. 2024) Appendix E. The authors performed extensive analysis on these datasets to ensure they represent a variety of problems that are not easily solved and are similar to a typical application of AI/ML to real world tabular data. We conduct our own hyper-parameter search with baseline models using the same hyper-parameter ranges as in (Grinsztajn, Oyallon, and Varoquaux 2022) and a larger validation split size, as the method described in (Grinsztajn, Oyallon, and Varoquaux 2022) used too small a validation set for optimal hyper-parameter selection and did not use the methods described in (Kadra et al. 2021) to tune the MLP based models.

Our study focuses on classification and we evaluate all algorithms using micro averaged AUC so as to require good performance on all classes and to not impact performance with less than optimal decision boundary selection. The results of our hyper-parameter search, shown in Table 3, show the average test AUC score *normalized to the AUC for Random Forest* for each algorithm from 5 models trained with different random seeds (1 to 5) using the best hyper-parameters identified with an extensive, approximately 400 trial search guided by a TPESampler (Bergstra, Yamins, and Cox 2013) from the Optuna (Akiba et al. 2019) library. For R-NRN and MLP, we allow the TPESampler to choose if FeatureBinarizationFromTrees (Arya et al. 2019) pre-processing is used, explained further in (Carrow et al. 2024) Appendix L. For Boolean Rules via Column Generation (BRCG), DiffLogic (DIF), NAM, CoFrNet (CFN), DANet (DAN), and FT-Transformer (FTT) we choose hyper-parameter ranges suggested by the authors if available, or infer the ranges from the author's code bases.

We examine the average scores[2] along with the results of the two-tailed Mann-Whitney U test with $\alpha = 0.05$ and find that R-NRN performs similarly to XGB, RF, GBT, NAM, CFN, and DAN. R-NRN also outperforms other logic based methods BRCG (+7%) and DIF (+16%), some neural methods designed for tabular data FTT (+28%), and traditional

[2]DIF reported using only seed 42. All tuning used seed 42 and DIF test AUC is random for all other seeds.

	R-NRN	BRCG	DIF	NAM	CFN	FTT	DAN	MLP	RF	XGB	GBT
Average	0.980	0.913$^{\blacktriangle 7\%}$	0.844$^{\blacktriangle 16\%}$	<u>1.028</u>$^{\blacktriangledown 5\%}$	0.991$^{\blacktriangledown 1\%}$	0.763$^{\blacktriangle 28\%}$	0.959$^{\blacktriangle 2\%}$	0.865$^{\blacktriangle 13\%}$	1.000$^{\blacktriangledown 2\%}$	0.979$^{\blacktriangle 0\%}$	0.993$^{\blacktriangledown 1\%}$
Avg. Time	114	143	77	189	213	293	176	174	37	**10**	21
Avg. Params	**1**	-	6	861	37	182	1463	62	-	-	-
Ours vs W/L/T		18/0/4	21/0/1	0/20/2	4/9/9	20/2/0	8/2/12	12/2/8	5/12/5	5/11/6	7/9/6
Ours vs p-value		0.047*	0.001**	0.130	0.614	0.000***	0.460	0.027*	0.734	0.769	0.991

Table 3: Experimental results showing average test AUC of 5 models trained with different random seeds and *normalized to RF scores*. <u>Underline</u> is highest average test AUC. Average trial training time in seconds and average number of parameters in thousands (for neural methods, otherwise "-") computed over a benchmark of 22 tabular classification datasets across a variety of algorithms. We show Win/Loss/Ties pairwise comparisons and p-values from two-tailed Mann-Whitney U test comparing R-NRN to other algorithms. Average test score of R-NRN better $^{\blacktriangle \%}$ / worse $^{\blacktriangledown \%}$ vs Other.

deep learning MLP (+13%).

Explanation quality Quantitative evaluations of model explanation are aimed at producing measures that indicate the quality of a produced explanation and can be defined in a variety ways, each measuring different aspects of an explanation (Nauta et al. 2022).

Size is a measure of explanation compactness and gives information regarding the presentation of the explanation (Nauta et al. 2022). Smaller explanations are generally considered better since those will be easiest to understand, although computing the size of an explanation is dependent on the explanations format. For NAM, RF, and SHAP we count features, since an explanation consists of a feature importance value for each input feature. For R-NRN, we compute explanation size as the average of the count of rules produced by a sample of explanations on the test set.

Our second metric, Single Deletion, is a measure of correctness, which is an evaluation of how faithfully the explanation describes the model (Nauta et al. 2022). A Single Deletion score can be calculated for any algorithm that produces feature importance; for R-NRN see (Carrow et al. 2024) Appendix K. We report both Spearman and Pearson Correlation between the feature importance score produced by each algorithm and the change in AUC on the test set when that single feature is perturbed.

While there are many more metrics, these two provide essential information when evaluating explanation quality. Namely, can a human reasonably understand the explanation (size), and can a human trust that the explanation is correctly describing the model (single deletion). Table 4 shows R-NRN explanations are 31% smaller, while also more accurate in terms of Single Deletion when compared with both NAM and RF. While Single Deletion Pearson Correlation is better for SHAP applied to RF, R-NRN explanations are generated 96% faster than SHAP on RF. We choose these algorithms as our baselines since the comparison covers other inherently interpretable methods (NAM), our best performing deep-tree-ensemble model (RF), and a post-hoc explanation technique (SHAP).

Table 2 presents an example of sample level explanation produced by R-NRN on the California dataset for qualitative analysis. This data consists of predicting if the price of a house is above or below the median and includes features such as the number of rooms, number of bedrooms, latitude, longitude etc. The example compares randomly selected explanations for the positive and negative class and demonstrates the ease with which one can interpret explanations using R-NRN. Furthermore, the explanations match our intuition about the factors that impact housing prices. (Carrow et al. 2024) Appendix H provides additional qualitative analysis comparing R-NRN and NAM explanations.

	R-NRN	NAM	RF	SHAP(RF)
Size	14.8	21.3$^{\blacktriangledown 31\%}$	21.3$^{\blacktriangledown 31\%}$	21.3$^{\blacktriangledown 31\%}$
SD Spearman	0.80	0.74$^{\blacktriangle 8\%}$	0.73$^{\blacktriangle 10\%}$	0.79$^{\blacktriangle 1\%}$
SD Pearson	0.83	0.74$^{\blacktriangle 12\%}$	0.80$^{\blacktriangle 4\%}$	0.91$^{\blacktriangledown 9\%}$
Avg. Time	270	194$^{\blacktriangle 39\%}$	0.0$^{\blacktriangle \text{N/A}}$	6005$^{\blacktriangledown 96\%}$

Table 4: Explanation evaluation with *Size* and *Single Deletion* Spearman/Pearson Correlation, and average time in seconds to compute feature importance, reported against benchmark datasets (Grinsztajn, Oyallon, and Varoquaux 2022). R-NRN *Size* and *Avg. Time* better $^{\blacktriangledown \%}$ / worse $^{\blacktriangle \%}$ vs Other. R-NRN *Single Deletion* score better $^{\blacktriangle \%}$ / worse $^{\blacktriangledown \%}$ vs Other.

5 Conclusion

Limitations R-NRN shows good performance on the benchmark datasets for tabular classification, but it is not yet clear if this approach will work for regression or other fundamental problems in AI such as with more complex domains like Computer Vision or Natural Language Processing. Furthermore, the flexibility to modify the initialized structure is somewhat limited in R-NRN, raising questions as to if this approach can be used on such complex domains. Also, while we present examples of explanations produced by R-NRN as well as quantitative analysis of their quality, we do not perform a user-study that would give more insight into the usefulness of such explanations in practice.

Conclusion NRNs are used to construct a supervised classification algorithm, R-NRN, with performance on tabular classification similar to RF, XGB, and GBT. R-NRN explanations are smaller and more accurate compared to feature importance based approaches including RF and NAM. Future work could leverage NRNs to develop inherently interpretable algorithms for NLP, computer vision, and reasoning domains.

Acknowledgements

A special thanks to Naoki Abe for sharing his experience and knowledge with us along our way.

References

Adadi, A.; and Berrada, M. 2018. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6: 52138–52160.

Agarwal, R.; Melnick, L.; Frosst, N.; Zhang, X.; Lengerich, B.; Caruana, R.; and Hinton, G. 2021. Neural Additive Models: Interpretable Machine Learning with Neural Nets. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Arya, V.; Bellamy, R. K. E.; Chen, P.-Y.; Dhurandhar, A.; Hind, M.; Hoffman, S. C.; Houde, S.; Liao, Q. V.; Luss, R.; Mojsilović, A.; Mourad, S.; Pedemonte, P.; Raghavendra, R.; Richards, J.; Sattigeri, P.; Shanmugam, K.; Singh, M.; Varshney, K. R.; Wei, D.; and Zhang, Y. 2019. One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques.

Barredo Arrieta, A.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; Garcia, S.; Gil-Lopez, S.; Molina, D.; Benjamins, R.; Chatila, R.; and Herrera, F. 2020. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *Information Fusion*, 58: 82–115.

Bergstra, J.; Yamins, D.; and Cox, D. 2013. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In Dasgupta, S.; and McAllester, D., eds., *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, 115–123. Atlanta, Georgia, USA: PMLR.

Borisov, V.; Leemann, T.; Seßler, K.; Haug, J.; Pawelczyk, M.; and Kasneci, G. 2022. Deep Neural Networks and Tabular Data: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21.

Breiman, L. 2001. Random Forests. *Machine Learning*, 45(1): 5–32.

Carmichael, Z.; and Scheirer, W. 2023. How Well Do Feature-Additive Explainers Explain Feature-Additive Predictors? In *XAI in Action: Past, Present, and Future Applications*.

Carrow, S.; Erwin, K.; Vilenskaia, O.; Ram, P.; Klinger, T.; Khan, N.; Makondo, N.; and Gray, A. G. 2024. Neural Reasoning Networks: Efficient Interpretable Neural Networks with Automatic Textual Explanations. arXiv:2410.07966.

Chen, J.; Liao, K.; Wan, Y.; Chen, D. Z.; and Wu, J. 2022. DANets: Deep Abstract Networks for Tabular Data Classification and Regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(4): 3930–3938.

Chen, T.; and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. *CoRR*, abs/1603.02754.

Dash, S.; Gunluk, O.; and Wei, D. 2018. Boolean Decision Rules via Column Generation. In *Advances in Neural Information Processing Systems*, volume 31.

Dubey, A.; Radenovic, F.; and Mahajan, D. 2022. Scalable Interpretability via Polynomials. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in Neural Information Processing Systems*.

Friedman, J. H. 2001. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5): 1189 – 1232.

Gorishniy, Y.; Rubachev, I.; Khrulkov, V.; and Babenko, A. 2021. Revisiting Deep Learning Models for Tabular Data. In *Advances in Neural Information Processing Systems*, volume 34, 18932–18943. Curran Associates, Inc.

Grinsztajn, L.; Oyallon, E.; and Varoquaux, G. 2022. Why do tree-based models still outperform deep learning on typical tabular data? In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2018. A Survey of Methods for Explaining Black Box Models. *ACM Computing Surveys*, 51(5): 93:1–93:42.

Hastie, T. J. 2017. Generalized additive models. In *Statistical models in S*, 249–307. Routledge.

Jiang, H.; Gurajada, S.; Lu, Q.; Neelam, S.; Popa, L.; Sen, P.; Li, Y.; and Gray, A. 2021. LNN-EL: A Neuro-Symbolic Approach to Short-text Entity Linking. In Zong, C.; Xia, F.; Li, W.; and Navigli, R., eds., *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 775–787. Online: Association for Computational Linguistics.

Kadra, A.; Lindauer, M.; Hutter, F.; and Grabocka, J. 2021. Regularization is all you Need: Simple Neural Nets can Excel on Tabular Data. *CoRR*, abs/2106.11189.

Kapanipathi, P.; Abdelaziz, I.; Ravishankar, S.; Roukos, S.; Gray, A. G.; Astudillo, R. F.; Chang, M.; Cornelio, C.; Dana, S.; Fokoue, A.; Garg, D.; Gliozzo, A.; Gurajada, S.; Karanam, H.; Khan, N.; Khandelwal, D.; Lee, Y.; Li, Y.; Luus, F. P. S.; Makondo, N.; Mihindukulasooriya, N.; Naseem, T.; Neelam, S.; Popa, L.; Reddy, R. G.; Riegel, R.; Rossiello, G.; Sharma, U.; Bhargav, G. P. S.; and Yu, M. 2020. Question Answering over Knowledge Bases by Leveraging Semantic Parsing and Neuro-Symbolic Reasoning. *CoRR*, abs/2012.01707.

Kimura, D.; Ono, M.; Chaudhury, S.; Kohita, R.; Wachi, A.; Agravante, D. J.; Tatsubori, M.; Munawar, A.; and Gray, A. 2021. Neuro-Symbolic Reinforcement Learning with First-Order Logic. *CoRR*, abs/2110.10963.

Kroeger, N.; Ley, D.; Krishna, S.; Agarwal, C.; and Lakkaraju, H. 2023. Are Large Language Models Post Hoc Explainers? In *XAI in Action: Past, Present, and Future Applications*.

Linardatos, P.; Papastefanopoulos, V.; and Kotsiantis, S. 2021. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy*, 23(1): 18.

Lundberg, S. M.; and Lee, S. 2017. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874.

Madaan, N.; and Bedathur, S. 2023. Diffusion-Guided Counterfactual Generation for Model Explainability. In *XAI in Action: Past, Present, and Future Applications*.

Nauta, M.; Trienes, J.; Pathak, S.; Nguyen, E.; Peters, M.; Schmitt, Y.; Schlötterer, J.; van Keulen, M.; and Seifert, C. 2022. From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI. *CoRR*, abs/2201.08164.

Petersen, F.; Borgelt, C.; Kuehne, H.; and Deussen, O. 2022. Deep Differentiable Logic Gate Networks. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 2006–2018. Curran Associates, Inc.

Puri, I.; Dhurandhar, A.; Pedapati, T.; Shanmugam, K.; Wei, D.; and Varshney, K. R. 2021. CoFrNets: Interpretable Neural Architecture Inspired by Continued Fractions. In *Advances in Neural Information Processing Systems*, volume 34, 21668–21680.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *CoRR*, abs/1602.04938.

Riegel, R.; Gray, A. G.; Luus, F. P. S.; Khan, N.; Makondo, N.; Akhalwaya, I. Y.; Qian, H.; Fagin, R.; Barahona, F.; Sharma, U.; Ikbal, S.; Karanam, H.; Neelam, S.; Likhyani, A.; and Srivastava, S. K. 2020. Logical Neural Networks. *CoRR*, abs/2006.13155.

Rudin, C. 2019. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence*, 1(5): 206–215.

Sen, P.; Carvalho, B. W. S. R. d.; Riegel, R.; and Gray, A. 2022. Neuro-Symbolic Inductive Logic Programming with Logical Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8): 8212–8219.

Slack, D.; Hilgard, S.; Jia, E.; Singh, S.; and Lakkaraju, H. 2020. Fooling LIME and SHAP: Adversarial Attacks on Post Hoc Explanation Methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, 180–186. New York, NY, USA: Association for Computing Machinery.

Wei, D.; Dash, S.; Gao, T.; and Gunluk, O. 2019. Generalized Linear Rule Models. In *Proceedings of the 36th International Conference on Machine Learning*, 6687–6696. PMLR.

Yuan, J.; and Dasgupta, A. 2023. A Simple Scoring Function to Fool SHAP: Stealing from the One Above. In *XAI in Action: Past, Present, and Future Applications*.