Towards a Partial Coverage of Neural Network Explanations Using Approximation Networks

Mathieu Brassart[1], Laurent Simon[1]

[1]Université de Bordeaux, Bordeaux INP, CNRS, LaBRI, UMR 5800, F-33400 Talence, France

Abstract

We introduce a neuro-symbolic verification framework that challenges the conventional trade-off between accuracy and explainability by maintaining model precision while enabling formal verification or explanation for the majority of decisions. Rather than attempting to verify an entire neural network - often computationally intractable — we train it to decompose its decision boundary into smaller, verifiable submodels, allowing for partial coverage with Reduced Ordered Binary Decision Diagrams representations. This ensures rigorous verification for the majority of cases, while acknowledging and identifying cases that remain unverifiable.

Our method allows us to formally analyze over 94% of decisions in standard benchmarks without sacrificing accuracy. Additionally, our system is designed to identify cases where a decision cannot be formally verified, ensuring that uncertain predictions are flagged for human intervention or additional scrutiny. This guarantees that AI-driven decisions remain not only precise and explainable but also trustworthy. Our approach provides a scalable and practical solution for deploying safe, accountable, and formally verifiable AI, offering a new verification pathway where accuracy and interpretability are simultaneously achievable rather than being at odds.

Keywords

Neuro-Symbolic AI, Reduced Ordered Binary Decision Diagrams, Formal Verification, Bounded Decisions

1. Introduction

The widespread adoption of Artificial Intelligence (AI) and Machine Learning (ML) in critical applications (such as healthcare, finance, and autonomous systems) has created a growing demand for trustworthy and verifiable AI. However, modern machine learning models, particularly deep neural networks and complex ensemble methods, often function as black boxes, making their decisions difficult to interpret or formally verify. This lack of explainability and verifiability is a major concern in high-stakes environments, where erroneous or unverifiable predictions can lead to severe consequences.

Existing explainability and verification methods typically rely on post-hoc interpretability techniques [1, 2], inherently interpretable models such as decision trees [3, 4, 5, 6, 7, 8], or formal methods using Boolean circuits [9, 10] and logical reasoning approaches [11, 12, 13]. While these methods offer insights into model behavior, they fail to scale when applied to large, high-dimensional models, limiting their practical use in real-world AI systems. Thus, a long-standing challenge in AI verification is the apparent trade-off between accuracy and explainability, leading to the belief that highly accurate models must remain opaque and unverifiable. We challenge this notion by demonstrating that formal verification and explainability can be achieved without degrading performance but by accepting to only partially covering explainable decisions. Rather than attempting to verify an entire neural network — which is often computationally infeasible - we train it to decompose its decision boundary into smaller, verifiable submodels, leveraging an Reduced Ordered Binary Decision Diagram (ROBDD) representation of neural network submodules. In practice, our goal is to ensure that a vast majority of decisions can be formally verified or explained, while cases that cannot be verified are explicitly identified.

To do so, we introduce a hierarchical verification structure, where approximation networks, trained alongside the main model, are able to provide formal guarantees for most decisions while allowing complex, unverifiable cases to be flagged for additional control. We use a neuro-symbolic [14] approach with an hybrid model combining Neural Networks and ROBDD representations, enabling efficient

AEQUITAS 2025: Workshop on Fairness and Bias in AI | co-located with ECAI 2025, Bologna, Italy

mathieu.brassart@labri.fr (M. Brassart); lsimon@labri.fr (L. Simon)

logical reasoning over neural network submodules. ROBDDs are particularly attractive for verification because they allow polynomial-time queries over Boolean functions, enabling efficient verification of logical constraints, bias mitigation, and adversarial robustness properties. Our ROBDD representation is based on prior work [9, 10], which considers neural networks as Boolean circuits, and extends it using binarized neural networks [15] on submodules, thanks to decision boundary decomposition [16]. We ensure that a majority of decisions (over 94% on some examples as we will see) can be formally verified (i.e. are decided by a submodule represented as a ROBDD), while maintaining full model accuracy. This new approach challenges the traditional trade-off between accuracy and explainability by offering a new pathway for verification. Interestingly, our system is capable of identifying cases where a decision cannot be formally verified, ensuring that uncertain predictions are flagged for human intervention or additional scrutiny. This selective verification mechanism makes AI systems more reliable and trustworthy, particularly in safety-critical applications.

The remainder of this paper is structured as follows. Section 2 examines the limitations of existing formal verification methods for neural networks. Section 3 introduces our approximation-based verification framework, detailing how smaller submodels constrain the overall decision space. Section 4 presents experimental results demonstrating that our approach preserves accuracy while allowing a significant proportion of decisions to be formally verified. Finally, we conclude by discussing the implications of our findings and potential future research directions.

2. Neural Networks Accessible to Formal Verification

Our approach applies formal verification (FM) to neural networks (NNs) by expressing them as Boolean circuits [9, 10]. However, for any formal method to be applicable, the AI model must first be expressed in a suitable formalism, close to propositional logic for FM, and, more importantly, must remain of manageable size. In our framework, we assume that a neural network (or one of its submodules) is verifiable if it can be expressed as a ROBDD. This assumption is motivated by the computational properties of ROBDDs, which allow efficient polynomial-time verification of Boolean functions within the Knowledge Compilation Map [17]. Unlike standard neural networks, which function as opaque black-box models, ROBDDs enable logical and constraint-based reasoning over decision functions. This makes them a powerful tool for formal AI verification.

Once expressed as an ROBDD, a (binarized) neural network allows for precise formal verification queries that are otherwise infeasible in conventional deep learning models, for instance bias elimination (ensuring that protected attributes such as gender or ethnicity do not affect predictions), safety verification in autonomous systems (guaranteeing that the AI model does not enter unsafe states under any possible input configuration) and even adversarial robustness. However, despite these advantages, applying ROBDDs to large-scale neural networks remains highly challenging. Converting a neural network into a ROBDD may have a time complexity of $O(n! \cdot n^2 \cdot 2^{n^2})$ and may result in a BDD of size $O(2^{n^2})$ (for a special family of function) [6] (Table 1), making it computationally infeasible for relatively small networks, and preventing its use in practical deep learning systems. Additionally, the process of binarizing neural networks to fit into a Boolean circuit framework is itself a challenging task. Previous studies [9] have highlighted that even when using binarized neural networks (BNNs) [15], the resulting Boolean structure remains too large for direct formal verification methods. The challenge, therefore, is not only in expressing neural networks as Boolean circuits, but also in managing the explosion of state-space complexity that arises during this transformation.

To address this scalability issue, we introduce approximation networks, which allow us to reduce the complexity of the ROBDD conversion while preserving the benefits of formal verification. Instead of verifying the full neural network, our approach selectively trains smaller subnetworks, designed to approximate specific decision boundaries while remaining tractable for ROBDD representation. These approximation networks act as verifiable surrogates, covering a large proportion of decisions that can be formally verified, while leaving only a limited set of complex cases unverifiable. By structuring the decision boundary in this way, we enable partial but scalable verification, ensuring that the majority

```
prev_layer ← layers_obdd[0]
p_vars ← prev_layer[0].vars
for each layer of layers_obdd[1:] do
    res_layer ← []
    for each obdd of layer do
        res_bdd ← obdd.compose(obdd.vars, prev_layer)
        res_bdd.vars ← p_vars
        append(res_layer, res_bdd)
    prev_layer ← res_layer
return prev_layer
```

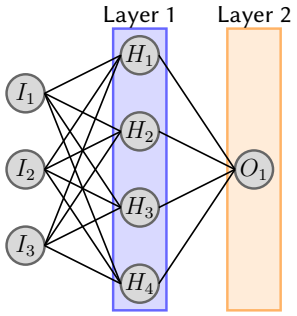

Figure 1: Algorithm combining a list of ROBDDs representing a neural network into a single ROBDD (left), and an example of a binarized neural network (right).

of AI decisions remain provably correct while explicitly flagging uncertain cases for further scrutiny. This approach provides a practical pathway for integrating formal verification into modern AI systems, balancing computational feasibility with the need for rigorous, logic-based guarantees in machine learning models.

2.1. Evaluating the Complexity of Neural Network Compilation

While it is possible to train a network directly in its binarized form, this often leads to optimization challenges due to the non-differentiability of the binarization function. Therefore, in practice, BNNs are usually trained using floating-point weights and activations, and then binarized afterward for compilation purposes. Binarized NN have binary inputs and outputs, but floating-point weights and biases. We use an activation layer between each fully connected layer that implements the threshold function 2.1. This ensures that every neuron input and output is binary, either 1 or 0 ($f(x) = 1$ if $x \geq 0$ otherwise 0). The binarization by itself is not particularly computationally expansive (if we accept to loose precision). The main challenge is to be able to "compile" a BNN into a ROBDD, in order to use formal verification on it.

During the training phase, the activation layers function as sigmoid activations, facilitating faster training. The compilation method follows these steps: first, each layer is processed sequentially, considering one neuron at a time, based on the algorithm of [5]. Each neuron is then converted into an ROBDD, expressed in terms of the outputs of the previous layer or the input features for the first layer. Then, by replacing the previous layer's outputs with the corresponding BDD, a single BDD is generated, representing the neural network's output in terms of the input features (see Algorithm 2.1).

This algorithm can be illustrated using the example of a binarized neural network in Figure 1: we sequentially express the outputs (h_1, h_2, h_3, h_4) of the first layer with inputs (i_1, i_2, i_3), and the output (o_1) of the second layer with the outputs (h_1, h_2, h_3, h_4) of the first layer. At the end of the algorithm, a single BDD will be constructed, representing the output in terms of the inputs.

Table 1
Compilation time for compiling a neural network into an ROBDD. Top: The NN is a fully connected layers of size (Input Size, 10) and (10, 1), and Input Size is varying. Bottom: The NN is a fully connected layers of size (15, Hidden Layer Size) and (Hidden Layer Size, 1), and Hidden Layer Size is varying.

Input Size	5	10	15	20	25
Average Time (s)	0.018	0.18	3.54	102.5	2693
Number of Runs	1000	500	250	100	10

Hid. Layer Size	5	10	15	20	25
Average Time (s)	0.80	3.39	18.40	115.3	956.8
Number of Runs	500	250	100	50	10

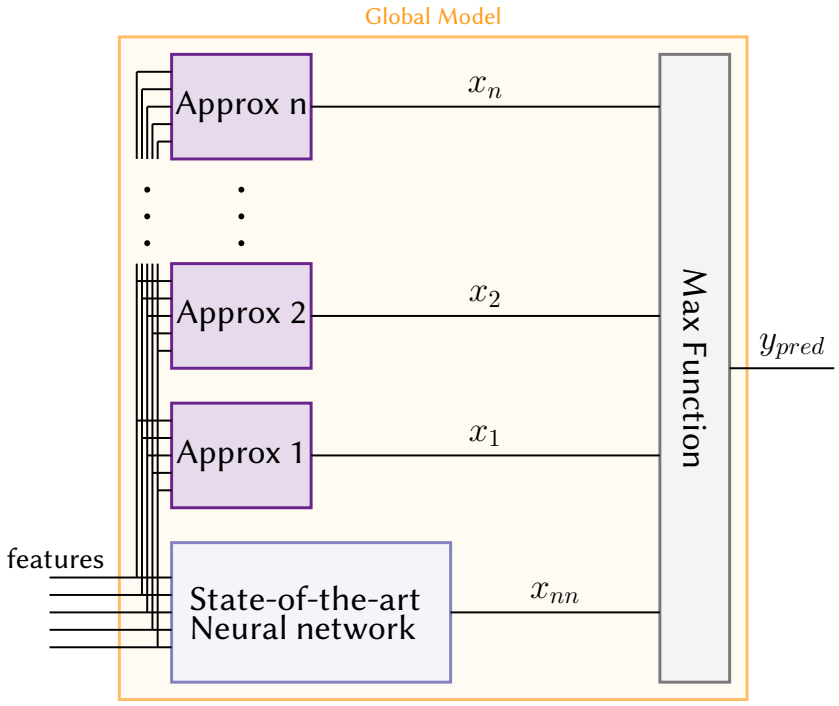

Figure 2: Model with a neural network and multiple approximation networks

Unfortunately, Table 1 illustrates our measured growth in computational cost required to rewrite a binarized neural network based on its size (also reported in [6]). In conclusion, while several prior approaches have attempted to address this problem, compiling a neural network into an ROBDD suitable for practical formal verification remains a major computational challenge (our network has only one hidden layer!).

3. Coverage of a Neural Network Model by Approximation Networks

We propose a novel structured verification approach that overcomes the limitations discussed earlier by integrating a logical layer that connects multiple neural networks, with controlled sizes. Our approach replaces a single large neural network with a Global Model that integrates multiple approximation networks (see Figure 2). Each approximation network is trained to capture a verifiable subset of decisions, while we expect the main neural network to remain responsible for handling more complex cases. The final prediction, denoted as y_{pred}, is computed as $y_{pred} = \max(x_1, x_2, \ldots, x_n, x_{nn})$, where x_i represents the output of an approximation network and x_{nn} is the output of the main neural network. This architecture ensures that if any approximator is confident in its decision, the final model prediction is immediately verifiable. If no approximator provides a decision, the main neural network takes over, handling more intricate cases that may not be directly verifiable.

The intuition behind our idea is to build on Decision Boundary Decomposition. We take advantage of the fact that many real-world decision boundaries are locally simple but globally complex: while the global decision boundary may be highly complex, many local subregions are governed by simple, well-separated decision rules. This phenomenon aligns with the Pareto principle, where (typically) 80% of the decision complexity arises from only 20% of cases. Our aim is to exploit this structure by assigning verifiable approximation networks to low-complexity regions, while more intricate cases are handled by the main neural network. This design follows a Pareto-like pattern, where the majority of decisions fall into simpler, verifiable subregions, significantly improving formal verification feasibility.

Note that, in the current work, we focus on explaining positive decisions (cases where the model responds "yes" or "1"), using the max function to capture verifiable subsets. While our framework is

currently designed for positive predictions, generalizing it to both "yes" and "no" decisions (0/1 outputs) is a natural extension and remains an ongoing area of research.

3.1. Guarantees on Explainability and Bias through Decision Composition

Since the final decision inherits the properties of at least one approximator when it takes precedence, we can ensure global model properties by controlling the behavior of its subcomponents. If all approximators are designed/proved to be fully explainable or bias-free, any decision one approximator covers will also be explainable at the global level: let's suppose that approximator x_i satisfies a given formal property ϕ (e.g., fairness, monotonicity, adversarial robustness), then for any input X where $x_i(X) = 1$, we can guarantee that the Global Model also satisfies ϕ on that input, since

$$y_{pred}(X) = \max(x_1(X), x_2(X), \ldots, x_n(X), x_{nn}(X)) \geq x_i(X)$$

For example, consider the case of fairness and explainability, where we construct multiple approximation networks that adhere to a specific constraint: *"The decision must not be influenced by the individual's gender."* Our decomposition of the decision-making process across multiple approximation networks (each one enforcing this fairness constraint) ensures that any decision made by one of these modules inherently satisfies the condition at the global level. If these approximation networks are trained to cover all positive cases, the fairness property extends to the entire model, guaranteeing that all decisions made by at least one approximator remain unbiased.

3.2. Training Strategies for the Global Model

To effectively integrate approximation networks into the global model, different training strategies may be proposed. One approach is **sequential training**, where approximation networks are trained first to handle simpler decisions before training the main neural network to focus on more complex cases. This method encourages approximation networks to learn the most verifiable decisions, improving explainability. Alternatively, **simultaneous training** allows all components to be trained together, ensuring consistency between models, though it may limit the effectiveness of approximation networks in capturing simple cases independently. Another option is **independent training with post-integration**, where each component is trained separately and later combined through the logical layer. This last method is only used in Section 4 as it is not relevant for the current experiment (each component is independently trained with classical methods that do not need to be studied here).

Gradient Propagation through the Max Layer For each of these methods, we also consider two different gradient propagation strategies through the max layer: **ALL** and **MAX**. The ALL method distributes the gradient to all inputs, while the MAX method propagates the gradient only to the maximum value. These methods may impact the performance of the global model, particularly in terms of coverage and accuracy.

Loss Function To optimize the training process, we employed a range of *ad hoc* loss functions in a preliminary study. The loss function that yielded the best results is a weighted binary cross-entropy function (see eq. 1), where p is a penalty applied to class 1. When the expected label is 0 but the predicted label is 1, the loss is p times larger than in the opposite case. The idea is to penalized submodules that wrongly predict class 1 (because the decision will be irrevocable): it is better for submodules to have a false negative, that can be recovered by the large NN.

$$L = -(p \cdot (1 - y) \cdot \log(1 - x) + y \cdot \log(x)) \tag{1}$$

Using this loss function provided better results compared to standard binary cross-entropy, particularly in terms of recall for class 0 and precision for class 1. However, this approach led to a reduction in precision for class 0 and a decrease in recall for class 1. Increasing the penalty p results in higher

confidence in class 1 predictions but at the expense of reduced coverage (for example, setting $p = 25$ often results in a model predicting only class 0). A suitable value must be chosen to balance precision and coverage (see below).

3.3. Tuning hyper-parameters on the Diabetes Dataset

We based our experimental study on all the available benchmarks on [18] for fairness that fitted our prerequisites. To adjust and test different scenarios for backpropagation mechanisms, loss functions and training methods of the different NNs within the global model, we first conducted a preliminary study on a simple and well-known simple tabular example: the "Diabetes" dataset [18, 19] (8 features and 2 classes). We used a model composed of 3 approximators and 1 neural network, trained with 100 runs of 500 epochs. The results of this preliminary study are presented in Table 2. The second step (section 4) is a more comprehensive study on additional datasets, using only the best method from this preliminary step, which will be presented in the next section.

The evaluation metrics used in our study include several key measures to assess the performance of the global model and its components. **Model Coverage** refers to the accuracy of the global model when predicting positive cases (class "1"). **Approximator Coverage** measures the accuracy considering only the predictions made by the approximator network. **Relative Coverage** represents the proportion of positive predictions made by the global model that are also predicted by the approximator, computed as the ratio of approximator coverage to model coverage ($\frac{\text{Approx coverage}}{\text{Model coverage}}$). **Neural Network Coverage** reflects the accuracy of the large neural network, though a lower value does not necessarily indicate poor performance, as the network is designed to handle only complex cases. Additionally, **Approximator Misclassifying Zeros** quantifies the percentage of false positives for class 1, capturing errors where the approximator incorrectly predicts a positive outcome. Finally, **Global Model Accuracy** accounts for the overall accuracy of the model across both classes, providing a global measure of system performance.

Table 2
Comparison of class 1 metrics for the two gradient propagation methods through the max layer. For this experiment, a model composed of 3 approximators and 1 larger neural network was used (Diabetes dataset, 100 runs, 500 epochs, p=2).

Training Method	Sequential		Simultaneous	
	ALL	MAX	ALL	MAX
Model Coverage	0.81	0.66	0.81	0.70
Approximator Coverage	0.72	0.66	0.73	0.24
Relative Coverage	0.89	1.00	0.89	0.34
Neural Network Coverage	0.63	0.00	0.34	0.50
Approximator Misclassifying Zeros	0.27	0.26	0.34	0.56
Global Model Accuracy	0.73	0.70	0.73	0.71

The results of this preliminary study are presented in Table 2. We observe that the ALL method performs best. However, it is quite surprising to see that the Simultaneous method also provides strong results. The main difference is seen in the incorrect classification of zeros by the approximator. This metric is particularly important because if an approximator incorrectly classifies a 0 as a 1, the large neural network will not be able to correct this decision, regardless of its size.

The Table 3 shows the results of the same experiment, but with different penalties applied to class 1. We can see that the best results are obtained with a penalty of 2, which provides a good balance between precision and recall for both classes.

4. Experimentation on Additional Datasets

We selected datasets containing tabular data and providing binary classification problems and we binarized their features that may be originally boolean, floating-point, or categorical. We included the datasets Compas ([20]), Diabetes ([19]), Titanic ([21]), and Adult ([22]), taken on [23]. Each dataset has

Table 3
Comparison of results obtained with different penalties applied to class 1 (Diabetes dataset, 100 runs, 500 epochs, sequential training with N approximator).

Penalty	N	F1-score	Precision		Recall	
			Class 0	Class 1	Class 0	Class 1
p = 1 (BCE)	1	0.69	0.73	0.69	0.67	0.73
p = 2	1	0.70	0.69	0.73	0.74	0.68
p = 5	1	0.73	0.70	0.79	0.82	0.65
p = 10	1	0.62	0.65	0.57	0.88	0.46
p = 25	1	0.34	0.51	0.01	1.00	0.01
p = 1 (BCE)	3	0.69	0.75	0.68	0.59	0.80
p = 2	3	0.72	0.77	0.71	0.65	0.81
p = 5	3	0.72	0.69	0.79	0.82	0.64
p = 10	3	0.66	0.67	0.65	0.86	0.53
p = 25	3	0.34	0.50	0.01	1.00	0.02

different feature dimensions and levels of binarization. They covers the main benchmarks available on the Fairness dataset [18]. The Compas dataset: 13 features, which expand to 18 binarized features, F1-score of 0.66 with a neural network. The Diabetes dataset: 8 features, increasing to 29 binarized features, F1-score of 0.76. The Titanic dataset: 7 features, which are transformed into 16 binarized features, yF1-score of 0.85. The Adult dataset: 14 features, expanded to 113 binarized features, F1-score of 0.81. An extensive search on Kaggle and other ML repositories was conducted to identify the most promising solutions and were selected for comparison (Table 4). This Table demonstrates that our approach yields very good results. Our method achieves F1-scores close to those of the other two methods, but, as we will see, it has the advantage of being able to provide explanations for and formally verify a large portion of positive predictions (75% in the worst-case dataset studied (Compas) and 94% in the best case (Adult). The worst case results (Compas) may be disappointing (75% of the positive predictions can be explained) but this is already a huge improvement compared to the 0% of overall neural network used as state of the art.

Dataset	F1	Precision		Recall	
		C0	C1	C0	C1
Compas	0.65	0.64	0.68	0.71	0.61
Diabetes	0.73	0.73	0.74	0.76	0.71
Titanic	0.80	0.77	0.84	0.85	0.75
Adult	0.80	0.82	0.79	0.78	0.83

	F1	Precision		Recall	
		C0	C1	C0	C1
...	0.66	0.65	0.68	0.71	0.62
...	0.75	0.75	0.77	0.77	0.74
...	0.80	0.78	0.82	0.84	0.76
...	0.82	0.83	0.81	0.81	0.83

Dataset	F1-Score	Precision		Recall	
		C0	C1	C0	C1
Compas	0.65	0.66	0.66	0.65	0.66
Diabetes	0.72	0.77	0.71	0.65	0.81
Titanic	0.77	0.79	0.77	0.75	0.79
Adult	0.78	0.87	0.73	0.66	0.90

Table 4
Perfomances of state of the art Neural Network (100 runs) Upper Left; Random Forests (100 estimators) Upper Right; and our method (100 runs, 500 epochs, 3 approximators, sequential training, p=2) Lower.

One limitation of our current work is that, for simplicity, both the neural network and the approximators are trained with binarized datasets, which decrease the performances of the large NN (we are working on this point). Our final observation concerns how the different neural networks are trained. The Table 5 shows that sequential and separate trainings are the best methods for covering most of the cases with approximation networks. Between these two methods, sequential training appears to yield slightly fewer misclassifications and a better coverage than separate training. As for simultaneous training, it generally results in significantly more "0" misclassifications and lower coverage. Additionally,

Table 5
Comparison of multiple training processes for a model containing 1 or 3 approximators and a neural network, evaluating performance on class 1. (p=1.2, average of 100 runs, 500 epochs per run)

Dataset	Model	1 Approx. + NN			3 Approx. + NN		
	Training Method	Seq.	Simult.	Sep.	Seq.	Simult.	Sep.
Adult	Model Coverage	0.88	0.83	0.92	0.91	0.84	0.97
Adult	Approx. Coverage	0.83	0.29	0.84	0.88	0.41	0.96
Adult	Relative Coverage	0.94	0.34	0.91	0.97	0.49	0.99
Adult	Approx. Misclassified Zeros	0.26	0.58	0.26	0.28	0.44	0.32
Adult	Model Precision	0.77	0.80	0.78	0.77	0.80	0.75
Compas	Model Coverage	0.62	0.58	0.65	0.70	0.60	0.70
Compas	Approx. Coverage	0.47	0.34	0.47	0.61	0.40	0.61
Compas	Relative Coverage	0.75	0.57	0.72	0.87	0.64	0.87
Compas	Approx. Misclassified Zeros	0.28	0.46	0.28	0.36	0.45	0.33
Compas	Model Precision	0.66	0.66	0.66	0.63	0.66	0.65
Titanic	Model Coverage	0.75	0.75	0.78	0.78	0.79	0.82
Titanic	Approx. Coverage	0.60	0.56	0.61	0.74	0.68	0.76
Titanic	Relative Coverage	0.80	0.75	0.78	0.95	0.86	0.93
Titanic	Approx. Misclassified Zeros	0.15	0.13	0.16	0.23	0.22	0.26
Titanic	Model Precision	0.78	0.79	0.77	0.76	0.77	0.74
Diabetes	Model Coverage	0.79	0.77	0.82	0.81	0.84	0.91
Diabetes	Approx. Coverage	0.69	0.66	0.68	0.79	0.79	0.89
Diabetes	Relative Coverage	0.87	0.86	0.83	0.97	0.93	0.98
Diabetes	Approx. Misclassified Zeros	0.32	0.22	0.31	0.26	0.30	0.38
Diabetes	Model Precision	0.71	0.74	0.71	0.70	0.73	0.67

using the sequential or separate training methods allows for easy extension at a later date by adding new approximation networks. Another advantage is the ability to change the loss function for each part of the global model, unlike the simultaneous method, which does not easily support this flexibility.

5. Discussion

The increasing deployment of neural networks in critical domains highlights the importance of formal verification. As these models are often used in safety-critical or ethically sensitive settings, providing guarantees about their behavior is essential. Formal methods, such as compilation into decision diagrams, offer a promising path for analyzing and reasoning about the internal logic of neural models in a principled and rigorous manner. One of the main advantages of ROBDDs lies in their ability to efficiently count models, which can be leveraged to assess fairness criteria—such as balancing positive outcomes across protected groups—through exact counting.

Our approach, however, is not limited to neural networks. In principle, other interpretable approximators could be used, such as decision trees or small random forests. These models are inherently explainable and can sometimes yield more compact logical representations. Nonetheless, random forests are more difficult to integrate into end-to-end differentiable pipelines and typically lack the flexibility needed for tasks such as knowledge distillation or joint optimization with other components.

By contrast, using a neural network as the approximation module offers significantly more modeling freedom. Neural networks can be trained jointly with other subsystems, fine-tuned for specific downstream tasks, and serve as targets for distillation from larger, more accurate models. This enables a hybrid strategy in which a complex model is first trained for accuracy and then distilled into a smaller, verifiable architecture.

Thus, retaining a neural representation—even if it is eventually binarized for compilation and not trivially amenable to formal analysis—provides greater flexibility in the learning pipeline. It supports a wider range of optimization strategies while still enabling formal guarantees through the compilation step.

An additional benefit of our approach is that it enables the integration of symbolic prior knowledge through knowledge compilation [17]. Logical rules or domain-specific constraints can be encoded directly into ROBDDs and composed with the compiled neural network. This makes it possible to combine learned behavior with formally verified knowledge, yielding a hybrid system that is both data-driven and constraint-aware. Such integration can be particularly valuable in domains where certain properties must hold by design (e.g., legal compliance, physical laws), and supports reasoning tasks such as consistency checking, abduction, or symbolic inference over the compiled knowledge base.

One final point of discussion concerns a limitation of our current work: we only target positive decisions. To overcome this, we are investigating the use of the ROBDD from the current approximator as a routing mechanism. Specifically, the 1/0 output would indicate whether to delegate the input to a specialized approximator—for handling multiple classes—or to the main neural network. In this setting, the router functions as a selector, determining whether a given instance is sufficiently simple to be handled by an auxiliary component, or requires the full capacity of the original model. Our preliminary results suggest that this routing strategy could achieve similar levels of coverage while extending the approach beyond binary classification.

6. Conclusion

The goal of our proposed method is to enable formal verification of simple cases decided by neural networks, which can be processed by reasonably sized subnetworks. We demonstrate that it is possible to integrate a small neural network within a larger neural network model in such a way that if any of the small networks decide "Yes" for a given decision, the global model will also decide "Yes". We show that even on non-trivial examples, very small neural networks (NNs) can capture a high proportion of decisions for which formal verification or explanation can be performed efficiently.

While our results demonstrate that up to 94% of decisions can be formally verified, there remain important limitations. The subset of unverifiable decisions (ranging from 6% to 25% depending on the dataset) may correspond to more complex or edge-case scenarios, potentially requiring additional scrutiny. Furthermore, our approach relies on a binarization step that can lead to feature explosion, especially in datasets with continuous variables. Future work will explore hybrid approaches combining our method with adversarial training techniques to improve coverage in high-complexity cases.

A key advantage of our method is its ability to identify unverifiable decisions—cases where the approximation networks fail to provide a provable answer—allowing these instances to be flagged for further scrutiny. In other words, our approach ensures that the system "knows when it doesn't know", a crucial property for safe and trustworthy AI deployment.

Looking forward, a natural extension of this work is to explore knowledge distillation as a means to apply our approach to existing neural networks (instead of training the approximator at the same time). By systematically transferring knowledge from the larger model to smaller, verifiable networks, we could refine the decision boundary decomposition even further, maximizing the proportion of formally verifiable decisions. Additionally, extending this framework to handle both "yes" and "no" decisions, and integrating it with adversarial robustness techniques, could further enhance its applicability in real-world AI deployment. We also want to study the exact nature of the cases that remain unverifiable. Are they edge cases, adversarial examples, or ambiguous samples?

By bridging the gap between machine learning and formal verification, this work paves the way for scalable, explainable, and provably reliable AI systems. This contributes to the broader goal of building trustworthy and accountable artificial intelligence.

Acknowledgments

This work has been partially funded by the "Chaire IA Digne de Confiance" (Chair on Trustworthy AI), operated by the Bordeaux University Foundation.

Declaration on Generative AI

All the experiments and code were developped without generative AI. The translation of a number of sentences in this paper, the english grammar and language were improved by text generation AI.

References

[1] G. V. den Broeck, A. Lykov, M. Schleich, D. Suciu, On the tractability of shap explanations, 2021. arXiv:2009.08634.

[2] Y. Nohara, K. Matsumoto, H. Soejima, N. Nakashima, Explanation of machine learning models using shapley additive explanation and application for real data in hospital, Computer Methods and Programs in Biomedicine 214 (2022) 106584.

[3] G. Audemard, S. Bellart, L. Bounia, F. Koriche, J.-M. Lagniez, P. Marquis, On the Explanatory Power of Boolean Decision Trees, Data and Knowledge Engineering 142 (2022) 102088. URL: https://hal.science/hal-03939107. doi:10.1016/j.datak.2022.102088.

[4] A. Shih, A. Choi, A. Darwiche, A symbolic approach to explaining bayesian network classifiers, 2018. arXiv:1805.03364.

[5] H. Chan, A. Darwiche, Reasoning about bayesian network classifiers, 2012. arXiv:1212.2470.

[6] Y. Tang, K. Hatano, E. Takimoto, Boosting-based construction of bdds for linear threshold functions and its application to verification of neural networks, 2023. arXiv:2306.05211.

[7] Y. Zhang, Z. Zhao, G. Chen, F. Song, T. Chen, Precise quantitative analysis of binarized neural networks: a bdd-based approach, ACM Transactions on Software Engineering and Methodology 32 (2023) 1–51.

[8] A. Shih, A. Darwiche, A. Choi, Verifying binarized neural networks by Angluin-Style learning, in: Theory and Applications of Satisfiability Testing–SAT 2019: 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9–12, 2019, Proceedings, volume 11628, Springer, 2019, p. 354.

[9] A. Choi, W. Shi, A. Shih, A. Darwiche, Compiling neural networks into tractable boolean circuits, intelligence (2017).

[10] W. Shi, A. Shih, A. Darwiche, A. Choi, On tractable representations of binary neural networks, 2020. arXiv:2004.02082.

[11] F. Leofante, N. Narodytska, L. Pulina, A. Tacchella, Automated verification of neural networks: Advances, challenges and perspectives, 2018. arXiv:1805.09938.

[12] J.-A. Goulet, L. H. Nguyen, S. Amiri, Tractable approximate gaussian inference for bayesian neural networks, 2021. arXiv:2004.09281.

[13] A. Ignatiev, N. Narodytska, J. Marques-Silva, Abduction-based explanations for machine learning models, 2018. arXiv:1811.10656.

[14] T. R. Besold, A. d. Garcez, S. Bader, H. Bowman, et al., Neural-symbolic learning and reasoning: A survey and interpretation, Cognitive Computation 9 (2017) 395–419. URL: https://doi.org/10.1007/s12559-017-9491-6.

[15] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, 2016. arXiv:1602.02830.

[16] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer, 2009.

[17] A. Darwiche, P. Marquis, A knowledge compilation map, Journal of Artificial Intelligence Research 17 (2002) 229–264. URL: http://dx.doi.org/10.1613/jair.989. doi:10.1613/jair.989.

[18] R. K. E. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, J. Martino, S. Mehta, A. Mojsilovic, S. Nagar, K. N. Ramamurthy, J. Richards, D. Saha, P. Sattigeri, M. Singh, K. R. Varshney, Y. Zhang, AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias, 2018. URL: https://arxiv.org/abs/1810.01943.

[19] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, R. S. Johannes, Using the adap learning algorithm to forecast the onset of diabetes mellitus, in: Proceedings of the Annual Symposium

on Computer Applications in Medical Care, American Medical Informatics Association, 1988, pp. 261–265. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2245318/.

[20] J. Angwin, J. Larson, S. Mattu, L. Kirchner, Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks, ProPublica (2016). URL: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[21] British Board of Trade, Report on the loss of the titanic (s.s.), 1912. URL: https://www.kaggle.com/c/titanic.

[22] D. Dua, C. Graff, UCI Machine Learning Repository - Adult Dataset, Technical Report, University of California, Irvine, School of Information and Computer Sciences, 2019. URL: https://archive.ics.uci.edu/ml/datasets/adult.

[23] Kaggle, Kaggle: Your machine learning and data science community, 2024. URL: https://www.kaggle.com.