Logic Gate Neural Networks are Good for Verification

Fabian Kresse FABIAN.KRESSE@IST.AC.AT
Emily Yu EMILY.YU@IST.AC.AT
Christoph H. Lampert CHL@IST.AC.AT
Thomas A. Henzinger TAH@IST.AC.AT
Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria

Editors: G. Pappas, P. Ravikumar, S. A. Seshia

Abstract

Learning-based systems are increasingly deployed across various domains, yet the complexity of traditional neural networks poses significant challenges for formal verification. Unlike conventional neural networks, learned Logic Gate Networks (LGNs) replace multiplications with Boolean logic gates, yielding a sparse, netlist-like architecture that is inherently more amenable to symbolic verification, while still delivering promising performance. In this paper, we introduce a SAT encoding for verifying global robustness and fairness in LGNs. We evaluate our method on five benchmark datasets, including a newly constructed 5-class variant, and find that LGNs are both verification-friendly and maintain strong predictive performance.

Keywords: Formal Verification, Logic Gate Networks, Global Robustness.

1. Introduction

Neural networks are increasingly utilized in emerging safety-critical domains such as self-driving cars, robotic systems, and healthcare devices, as they show great promise in solving complex real-world tasks (Jumper et al., 2021; Fawzi et al., 2022). However, ensuring correctness and reliability in these settings poses significant challenges, driven by the networks' inherent opacity and high complexity (Amodei et al., 2016). As a result, numerous verification methods have been put forward to address these issues by formally proving the robustness and fairness of neural networks. Among the commonly considered properties is local robustness, such that for a given input, a network is robust to a set of specified perturbations (Katz et al., 2017; Gehr et al., 2018; Singh et al., 2018). Another critical property is fairness, which requires that the network's predictions be free from bias with respect to attributes such as gender, ethnicity, and age. Extending these notions further, global robustness and fairness are 2-safety hyperproperties demanding that a network maintain robustness and fairness for all pairs of inputs (Athavale et al., 2024; Biswas and Rajan, 2023; Khedr and Shoukry, 2023).

Verification methods for neural networks can be grouped into two main categories: complete and incomplete. Incomplete approaches often employ semidefinite programming (Raghunathan et al., 2018; Dathathri et al., 2020; Fazlyab et al., 2022) or bound-propagation techniques (Wang et al., 2018a,b, 2021; Singh et al., 2019) to overapproximate the network's behavior. Although this yields efficient verification in practice, it may introduce approximation errors. In contrast, complete methods use mixed-integer linear programming (MILP) (Tjeng et al., 2019; Anderson et al., 2019) and satisfiability modulo theories (SMT) (Wu et al., 2024; Pulina and Tacchella, 2012; Katz et al., 2019) to provide definitive verdicts to verification queries by encoding the neural networks. However, the

exhaustive nature of these techniques often results in higher computational overhead, particularly for large-scale deep neural networks (DNNs).

Despite rapid advances in DNN verification, most efforts have focused on conventional neural networks, for which the verification task still faces significant challenges. As an alternative, some methods use quantized networks (Gholami et al., 2022; Lechner et al., 2023; Henzinger et al., 2021) and binary neural networks (Qin et al., 2020), whose verification can be performed using SMT/SAT solvers (Jia and Rinard, 2020). Recently, deep differentiable Logic Gate Networks (LGNs) have attracted attention for their potential to deliver fast, efficient inference while exhibiting promising performance (Petersen et al., 2022, 2024). Unlike binary neural networks, which discretize weights and activations, LGNs use a differentiable relaxation to learn a mixture over basic logic gate operators (e.g., NAND and XOR). LGNs can be directly deployed on FPGAs or ASICs, potentially yielding significant energy savings over conventional architectures and making them especially attractive for cyber-physical systems. Furthermore, due to their discrete and SAT-solving-friendly architecture, LGNs and related variants have shown promise for formal verification, particularly in the context of local robustness (Benamira et al., 2024).

In this paper, we take a first step toward SAT-based verification of learned logic gate networks by focusing on two global properties: robustness and fairness. Global robustness and fairness say that similar inputs should yield similar results; by comparing two input-output pairs, they are 2-safety hyperproperties (Athavale et al., 2024). Specifically, we propose a symbolic encoding that formalizes these hyperproperties in a confidence-based framework, enabling efficient verification by a SAT solver. We then conduct an experimental evaluation on five datasets, illustrating the promise of our approach for the practical verifiability and the competitive predictive performance of LGNs. To the best of our knowledge, this is the first exploration of using SAT solving for 2-safety hyperproperties such as global robustness and fairness in LGNs, and we hope that it will serve as an inspiration for further research in this direction.

Related work. We now briefly review related approaches to DNN verification for robustness. For a comprehensive overview, we refer to the survey paper of Liu et al. (2021). Most related to our work, Athavale et al. (2024) present an SMT encoding for both global robustness and fairness. A similar SMT-based approach for fairness has also been explored by Biswas and Rajan (2023) and Khedr and Shoukry (2023), but without incorporating a confidence-based verification framework. Ruan et al. (2019) propose a method for verifying global robustness by defining it as an expectation of the maximal safe radius over a test dataset. The work of Kabaha and Drachsler-Cohen (2024) proposes computing a minimal globally robust bound via solving a Mixed Integer Programming problem, and Wang et al. (2022) similarly encode the verification task as an MILP problem. Different from them, in this paper, we focus on verifying logic gate neural networks. The verification of LGN-related networks has been addressed by Benamira et al. (2024), but only for local robustness.

2. Verifying Logic Gate Networks

In this section, we study the verification problem of logic gate networks. We begin by introducing the notations for LGNs, and then present our SAT-based encoding for global robustness and fairness.

2.1. Logic Gate Networks for Multi-Class Classification

LGNs offer an alternative to conventional neural networks by replacing neurons with discrete logic gates. Formally, an LGN can be defined as a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose vertices $\mathcal{V}$ are grouped into layers. Each vertex $v \in \mathcal{V}$ in a layer corresponds to a two-input Boolean gate selected from a fixed set of 16 possible Boolean operations (e.g., NAND, XOR, etc.); the inputs are ordered since their positions affect non-commutative logic operations. The edges $\mathcal{E}$ define connections from gates in one layer to gates in the next layer, resulting in a sparse, netlist-like architecture rather than dense matrix multiplications. Note that each gate can have multiple outgoing edges, allowing its output to be fed into several subsequent gates. The connectivity is randomly initialized and remains fixed throughout training. During training, each gate is modeled as a continuous relaxation over the possible Boolean operators, mixed via a softmax. LGNs are inherently non-linear and thus do not rely on non-linear activation functions. Prior work has demonstrated the ability of LGNs to learn tasks such as CIFAR-10 and MNIST (Petersen et al., 2022, 2024).

As in previous work, we employ LGNs for multi-class classification. Let $\mathcal{X} \subseteq \mathbb{B}^d$ denote the input space (with inputs binarized using thermometer encoding for numerical features and one-hot encoding for categorical features), and let $\mathcal{Y} = \{1, 2, \ldots, C\}$ be the set of class labels. We consider a classification model $f : \mathcal{X} \to \mathcal{Y}$, where each input $x \in \mathcal{X}$ is mapped to a predicted label $f(x) \in \mathcal{Y}$. For C-class classification, $f(x)$ is obtained by an LGN's final layer producing O Boolean outputs, partitioned into C blocks of size L, where $O = C \times L$. Let $o_{j,k} \in \{0, 1\}$ denote the k-th output bit for class j. As in Petersen et al. (2022): $\text{score}(j) = \sum_{k=1}^{L} o_{j,k}$, for $j = 1, \ldots, C$. The predicted label is the class with the highest score: $f(x) = \arg\max_{1 \le j \le C} \text{score}(j)$.

We define the confidence for $f(x)$:

$$\text{conf}(f(x)) = \frac{\text{score}(f(x))}{\sum_{j=1}^{C} \text{score}(j)}. \tag{1}$$

2.2. Global Fairness and Robustness

We define our notions of global fairness and robustness using a confidence-based approach inspired by Athavale et al. (2024). Our setting assumes that inputs are discretized and encoded as binary vectors. This discretization leads to slight modifications in our definitions. To capture the aforementioned properties, we consider pairs of inputs, denoted x and x'. Comparing two inputs allows us to formalize the idea that if the network predicts with high confidence on one input, then any sufficiently similar input should receive the same label. This is also referred to as self-composition (Barthe et al., 2011). For a given confidence threshold $\kappa > 0$ and a numerical tolerance ϵ, given as an integer, we require that

$$\forall x,\, x' \;:\quad \Phi(x, x', \epsilon) \wedge \text{conf}(f(x)) > \kappa \quad \Longrightarrow \quad f(x) = f(x'), \tag{2}$$

where the definition of the similarity condition $\Phi(x, x', \epsilon)$ differs for global fairness and robustness.

Global Fairness. For fairness, we partition the input x into sensitive attributes and non-sensitive attributes. Subscripts are used to denote the ith bit of the input x. In this case, we require that the non-sensitive part satisfies the similarity condition, while the sensitive attributes differ. Formally, we define

$$\Phi(x, x', \epsilon) \;:=\; \left(\bigwedge_{i \in \mathcal{N}_n} d(x_i, x'_i) \le \epsilon \right) \wedge \left(\bigwedge_{j \in \mathcal{C}_n} (x_j = x'_j) \right) \wedge \left(\bigwedge_{k \in \mathcal{C}_s} (x_k \neq x'_k) \right), \tag{3}$$

where $\mathcal{N}_n$ and $\mathcal{C}_n$ denote non-sensitive numerical and categorical features, respectively, and $\mathcal{C}_s$ indexes sensitive attributes. For numerical features encoded using thermometer encoding, the distance function $d(\cdot, \cdot)$ counts the number of differing bits.

Global Robustness. For global robustness, we require that each numerical feature differs by at most ϵ and that the categorical features remain identical. Therefore, we can encode it like global fairness, with the number of sensitive attributes $\mathcal{C}_s$ set to the empty set.

These formulations, which capture conditions over pairs of inputs, are examples of hyperproperties (Clarkson and Schneider, 2010). More specifically, they are 2-safety properties. In the next section, we detail a SAT encoding to verify these properties for LGNs.

2.3. SAT Encoding

In this section, we present a SAT-based encoding for verifying global robustness and fairness defined above. As LGNs are fully discretized and operate on Boolean inputs and outputs, we can encode both the network behavior and the property constraints as Boolean formulas. In the following, we assume standard notations from classical Boolean logic. A literal is either a Boolean variable v or its negation $\neg v$. A formula in conjunctive normal form (CNF) is a conjunction of clauses, with each clause being a disjunction of literals. We write $\mathtt{f}(V)$ to denote a formula over sets of variables V and U, V denotes $U \cup V$. An assignment is a function that maps each variable to a Boolean value in $\{\bot, \top\}$. The satisfiability problem (SAT) is to determine, for a given CNF formula, whether there exists an assignment under which the formula is true. For clarity, in the following we use "$\simeq$" to denote syntactic equivalence between sets of variables, and "$\Rightarrow$" for semantic implication. We write $V[k]$ to denote the k-th variable in V.

First of all, we need to encode the network. Let V_{in} be the set of Boolean variables encoding an input $x \in \mathcal{X}$ and V_{out} be the output Boolean variables representing $f(x)$. Each logic gate in the LGN is encoded by its corresponding Boolean clause, so that the network is represented as: $V_{out} \simeq \mathtt{network}(V_{in})$. For a second input x', we define V'_{out} and V'_{in} symmetrically.

2.3.1. CONSTRAINTS ON INPUTS

Well-Formedness. We require that the assignments to V_{in} and V'_{in} conform to our encoding: numerical features must be correctly encoded using thermometer encoding, and categorical features must be one-hot encoded. We refer to this condition as $\mathtt{well\_formed}(V_{in})$ (and similarly for V'_{in}).

Numerical Proximity. For each numerical feature i (with $i \in \mathcal{N}$), let $V_{in}^{(i)}$ and $V_{in}^{\prime(i)}$ denote the thermometer encodings (each with B bits). We enforce that the two encodings differ by at most ϵ bit-flips. Formally, we define the proximity constraint for feature i as

$$\mathtt{prox}_\epsilon\big(V_{in}^{(i)}, V_{in}^{\prime(i)}\big) \;:=\; \bigwedge_{k=\epsilon}^{B-1} \Big[\Big(V_{in}^{(i)}[k] \Rightarrow V_{in}^{\prime(i)}[k-\epsilon]\Big) \wedge \Big(V_{in}^{\prime(i)}[k] \Rightarrow V_{in}^{(i)}[k-\epsilon]\Big)\Big]. \tag{4}$$

This ensures that the thermometer encodings for feature i in both inputs are within ϵ bit-flips.

Categorical. We enforce that categorical features remain identical across both inputs. This means that for each categorical feature i, every bit in $V_{in}^{(i)}$ and $V_{in}^{\prime(i)}$ must be the same. We refer to this condition as $\mathtt{same\_cat}(V_{in}^{(i)}, V_{in}^{\prime(i)})$.

2.3.2. CONSTRAINTS ON OUTPUTS

The output variables V_{out} are partitioned into C blocks, one for each class. Each block consists of a total of L bits. For class c (with $c \in \{1, \ldots, C\}$), let $V_{out}^{(c)}$ denote its corresponding block. To facilitate comparisons, we sort each block using a sorting network (Eén and Sörensson, 2006):

$$V_{out}^{(c),\text{sorted}} \simeq \texttt{sort\_net}\left(V_{out}^{(c)}\right). \tag{5}$$

The same procedure is applied to the corresponding block in the second output V'_{out}. When the superscript is omitted, we refer to sorting all output bits in order to encode the overall sum of ones.

Winning Condition. For each class c, we define a Boolean variable w_c that encodes the condition that class c is the winning (predicted) class. Specifically, we require that:

$$w_c \Rightarrow \Bigg(\underbrace{\bigwedge_{d<c} \bigwedge_{k=0}^{L-1} \left(V_{out}^{(d),\text{sorted}}[k] \Rightarrow V_{out}^{(c),\text{sorted}}[k] \right)}_{\text{class } c \text{ has confidence at least as high as classes } d<c} \wedge \underbrace{\bigwedge_{d>c} \bigvee_{k=0}^{L-1} \left(V_{out}^{(c),\text{sorted}}[k] \wedge \neg V_{out}^{(d),\text{sorted}}[k] \right)}_{\text{class } c \text{ has confidence higher than classes } d>c} \Bigg) \tag{6}$$

Similarly, we define w'_c for the primed output V'_{out}. Then, to capture the constraint that two inputs must be assigned different classes, we encode:

$$\texttt{diff\_class}(V_{out}, V'_{out}) \coloneqq \bigwedge_{c=1}^{C} \left(w_c \Rightarrow \neg w'_c \right). \tag{7}$$

Confidence. Finally, we define $\texttt{confidence}_{>\kappa}(V)$ to require that the network's output for input V exceeds a specified confidence threshold κ. We can write:

$$\texttt{confidence}_{>\kappa}(V_{out}) \coloneqq \bigwedge_{i=1}^{C \cdot L} \left(V_{out}^{\text{sorted}}[i-1] \Rightarrow \bigvee_{c=1}^{C} V_{out}^{(c),\text{sorted}}[\lfloor i \cdot \kappa \rfloor] \right). \tag{8}$$

2.3.3. OVERALL VERIFICATION CONDITION

To prove the validity of a propositional formula, we check whether its negation is unsatisfiable. We first define an overall formula:

$$\begin{aligned} \Psi \coloneqq\ & \texttt{well\_formed}(V_{in}) \wedge \texttt{well\_formed}(V'_{in}) \wedge \texttt{network}(V_{in}) \wedge \texttt{network}(V'_{in}) \\ & \wedge\ \texttt{confidence}_{>\kappa}(V_{out}) \wedge\ \texttt{diff\_class}(V_{out}, V'_{out}). \end{aligned} \tag{9}$$

We then specialize Ψ for our two properties as follows. First, Ψ_{fair} augments Ψ with the requirement that non-sensitive features are similar (numerically within ϵ and categorically identical), while the sensitive features differ. Formally,

$$\begin{aligned} \Psi_{\text{fair}} \coloneqq\ \Psi \wedge \Bigg(& \Big(\bigwedge_{i \in N_n} \texttt{prox}_{\epsilon}\left(V_{in}^{(i)}, V_{in}^{\prime(i)}\right) \wedge \bigwedge_{j \in C_n} \texttt{same\_cat}\left(V_{in}^{(j)}, V_{in}^{\prime(j)}\right) \Big) \\ & \wedge \Big(\bigwedge_{k \in \mathcal{C}_s} \neg\, \texttt{same\_cat}\left(V_{in}^{(k)}, V_{in}^{\prime(k)}\right) \Big) \Bigg). \end{aligned} \tag{10}$$

Similarly, for Ψ_{robust}, we use the same encoding as for Ψ_{fair}, with the set of sensitive attributes $\mathcal{C}_s$ being empty. Verification then reduces to proving that Ψ_{robust} or Ψ_{fair} is unsatisfiable.

Table 1: Dataset (after preprocessing) showing the dataset size, total number of features including the number of numeric features and the number of output classes.

Dataset	**#Samples**	**#Features** (Categorical/Numeric)	**#Classes**
German Credit (Hofmann, 1994)	1,000	16 (12/4)	2
Adult (Becker and Kohavi, 1996)	46,033	7 (4/3)	2
Law School (Wightman, 1998)	21,982	9 (5/4)	2
COMPAS (Larson, 2017)	60,798	8 (7/1)	3
Adult-5 Class (Ding et al., 2021)	195,665	7 (4/3)	5

3. Experiments

In this section, we present experimental results demonstrating the effectiveness of our proposed method for fairness and robustness verification of LGNs. We evaluate our approach on a range of classification tasks derived from five different datasets.

3.1. Datasets and Model Training Setup

Datasets. In Table 1, we summarize the key attributes of each dataset (number of instances, features, and classes). Four of these datasets—German Credit, Adult, Law School, and COMPAS—were used by prior work on DNN verification for global fairness and robustness verification (Athavale et al., 2024; Biswas and Rajan, 2023). We also investigate a five-class variant of the Adult dataset by partitioning annual income into five brackets, utilizing the Folktables dataset with data from the 2018 California census (Ding et al., 2021).

Data Splits and Encodings. All datasets are split into 64% training, 16% validation, and 20% test sets. Numeric attributes are discretized via thermometer encoding into a maximum of 20 buckets (except for German Credit, for which we use only 5 buckets due to a larger number of features). If the range of valid values is less than the maximum, fewer buckets are employed accordingly. Categorical features are one-hot encoded.

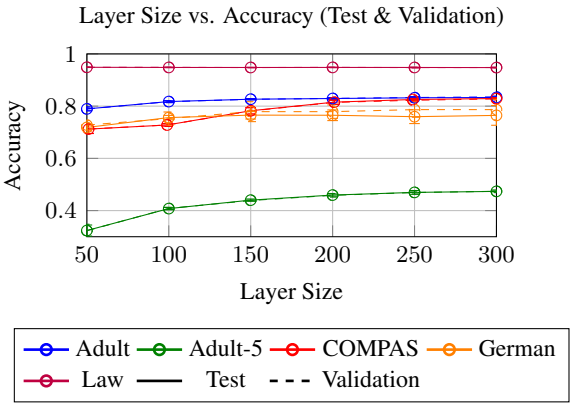

Figure 1: Validation and test accuracy achieved for five different classification tasks as a function of model layer size. Each point represents the mean accuracy over five runs with different random seeds. Error bars indicate standard deviation.

LGN Model Training. We train LGNs with three layers, each containing one of $\{50, 100, 150, 200, 250, 300\}$ gates to investigate different model sizes and the impact on verification runtime. Figure 1 illustrates the validation and test accuracy of our models, when hardened, across different layer sizes. For COMPAS, we adjust the layer sizes to the closest number divisible by 3 to accommodate ternary output blocks. For each layer size and dataset,

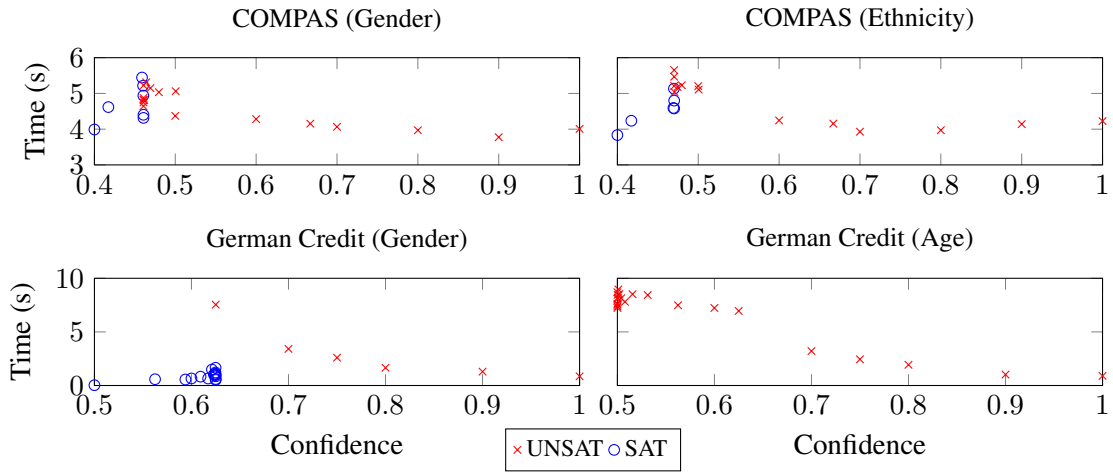

Figure 2: Solving time and result vs. confidence value for two example models (COMPAS with 300 and German Credit with 50 gates per layer).

we run five different random seeds (shuffling both data splits and initial gate parameters). We use the Adam optimizer with a constant learning rate of 0.01 for 200 epochs.

3.2. Verification Framework

In the experimental evaluation, we verify global fairness (GF) and global robustness (GR)—of LGN networks, as described in Section 2. In all experiments, we use the award-winning SAT solver Kissat (Biere et al., 2024) as the backend solver. All experiments are run on an AMD EPYC 9654 96-Core Processor. We conduct two types of experiments:

1. **Fixed-threshold experiments:** We perform queries at specific confidence thresholds (e.g., $\kappa = 0.5$, $\kappa = 0.99$) for GF. In these experiments, we measure the solver's runtime and record its outcome (UNSAT/SAT), corresponding to being globally fair or not, respectively.

2. **Binary search over κ:** For both GF and GR, we perform a binary search over κ (with a convergence threshold of 0.05) to identify the smallest safe confidence threshold at which no counterexample pair (x, x') exists. For each trial κ, we query the SAT solver. We then report the cumulative runtime across all queries as the total solving time.

3.3. Global Fairness Verification

For global fairness, we fix the allowed perturbation $\epsilon = 0$ on the non-sensitive attributes, meaning that only the sensitive attributes (gender, ethnicity, or age – for German Credit) may vary.

Solving time at fixed κ. Figure 2 shows solving times for one of our largest COMPAS models with 300 gates per layer and one of our smallest German Credit models with 50 gates per layer. We observe that solving time increase as we approach the globally robust boundary.

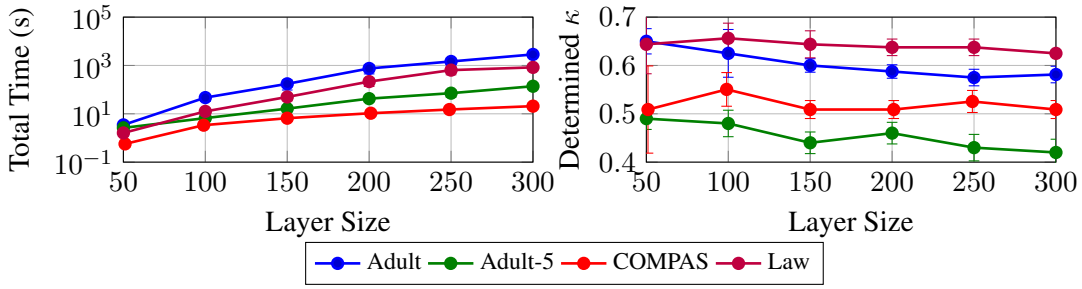

Figure 3: Comparison of runtime and determined confidence thresholds for the sensitive attribute ethnicity, as the number of gates per layer increases (standard deviation too small to see).

All verification queries for these two models are performed in under 10 seconds and, except for German Credit with the sensitive attribute age, which is globally robust for any confidence threshold, we find non-trivial confidence thresholds. We additionally verified that at least one input example x exists that results in a higher confidence output than the found boundary.

Minimum Confidence for Global Fairness. For the sensitive attributes gender and ethnicity, we perform a binary search over the confidence threshold κ to determine the smallest value at which the model is globally fair. We set the timeout to 8 hours, and all our models, except German Credit for layer sizes above 100, were successfully verified. In addition, for each found confidence threshold, we confirm that at least one input x that attains the corresponding confidence value exits, ensuring the threshold is not trivially satisfied. Figure 3 presents the cumulative runtime of the binary search (on a logarithmic scale) and the corresponding minimal confidence thresholds for the sensitive attribute ethnicity; results for gender are provided in the Appendix.

Our results suggest that two main factors influence the solver's performance. First, the number of output classes appears to affect runtime, with more classes generally leading to faster verification, likely due to a coarser resolution in the confidence computation. Second, the size of the input encoding—particularly the number of numerical features— impacts solving time. Although the Law School dataset has four numerical features, its values are represented using fewer buckets because the features have a limited range of integer values, which helps mitigate the expected slowdown. Further investigations on the effects of runtime are necessary in future work.

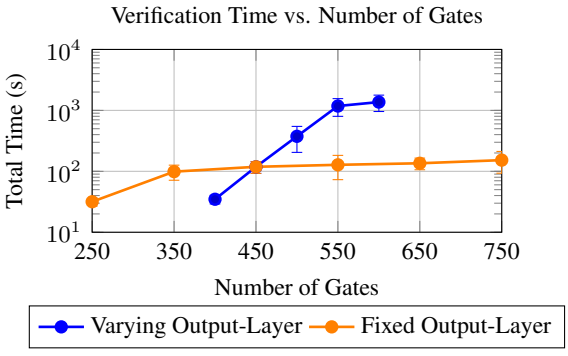

Figure 4: Adult dataset, varying vs. constant output layer size for 5 seeds. Either the output layer size or the size of the other layers is kept constant at 150 gates.

For all datasets the found confidence values tend to stay constant or decrease with increasing model capacity. However, this is not the case for the sensitive attribute gender, see Appendix.

Solving Time vs. Confidence Granularity. As the confidence score is computed as the ratio of activated gates in the winning class to the total number of activated gates, its granularity is limited

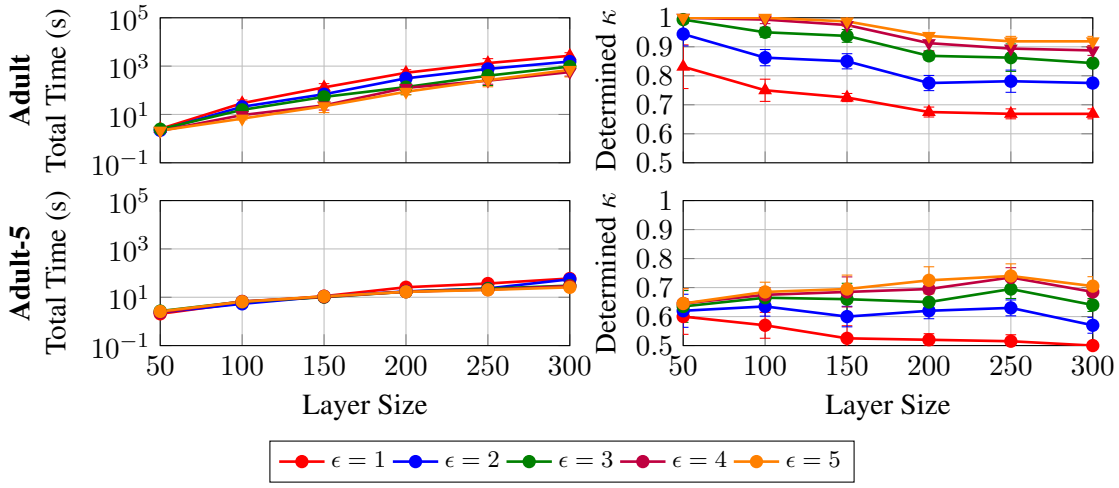

Figure 5: Comparison of runtime and determined confidence values, as the number of gates per layer increases (standard deviation too small to see).

by the number of output gates. For example, in a Adult configuration with 300 gates (i.e., 150 gates per output class), the highest attainable non-1.0 confidence is $\frac{150}{151} \approx 0.99$. Figure 4 compares two experimental settings: one in which the output layer size is varied while all preceding layers are fixed at 150 gates, and another in which the output layer size is held constant at 150 gates while the sizes of the preceding layers are varied. It can be observed that increasing the output layer size leads to a substantial increase in verification time, whereas maintaining a constant output layer size results in only a small increase. With respect to the models used in Figure 4, scaling the output layer yields a greater accuracy boost than adding gates earlier: the largest fixed output layer achieves 0.826 ± 0.001, versus 0.834 ± 0.003 when the final layer's gates are expanded to 300.

3.4. Global Robustness Verification

For global robustness, we allow up to ϵ flips in each binarized numerical feature while keeping categorical features fixed. As with fairness, we perform a binary search over the confidence threshold κ and record the cumulative solver runtime for all queries. Figure 5 shows the returned minimal confidence thresholds and corresponding runtimes for various ϵ values on the Adult and Adult-5 tasks. Results for other datasets are provided in the Appendix. In Figure 5, we annotate each confidence threshold with markers indicating whether valid inputs x exist that yield the reported confidence. A filled upward triangle (▲) denotes that all five model runs have at least one valid input resulting in the confidence value returned, a downward triangle (▼) indicates that no input x can produce the confidence value for any of the models, and a filled circle (●) signifies that valid inputs exist for some, but not all, models. As expected, the runtime for $\epsilon = 1$ is generally the highest, which is a tighter constraint on the inputs compared with higher values of ϵ. For higher ϵ values (e.g., $\epsilon = 4$ and $\epsilon = 5$), no valid inputs exist for the Adult dataset; in contrast, for the Adult-5 class task, such inputs exist, at least for some of our models.

3.5. Comparison with previous work

In general, direct comparisons are challenging as our approach employs LGNs and a SAT-based verification encoding, whereas the most closely related works Athavale et al. (2024) and Biswas and Rajan (2023) verify conventional floating-point neural networks using SMT-based methods.

Comparison with Athavale et al. (2024). Even our smallest LGN models (with the exception of COMPAS, which requires models with more than 100 gates) achieve classification accuracies comparable to or *higher than* those reported by Athavale et al. (2024). Although they reported verification times for most instances under 60 seconds, our reproduction attempts sometimes yield runtimes on the *order of minutes* (despite substantial communication with the authors, we could not reproduce their runtime numbers; possibly due to undocumented changes). Given these inconsistencies, we omit an experimental comparison. In addition, their SMT-based verification relies on a softmax approximation that results in potentially spurious counterexamples. This limits the number of output classes due to increasing errors, whereas our method scales to a multi-class setting, as demonstrated by our 5-class variant of the Adult dataset. Lastly, Athavale et al. (2024) did not verify the existence of an input x that satisfies the minimum confidence threshold.

Comparison with Biswas and Rajan (2023). As Biswas and Rajan (2023) consider only binary classification and do not incorporate explicit confidence scores, hence operating at a confidence threshold of 0.5. Their investigated networks for the Adult and German Credit tasks yield similar accuracies to our LGNs. In contrast to our work, their experiments do not report any UNSAT instances, only observing time-out or SAT. Furthermore, they report that only one network can be verified with a straightforward SMT encoding (with a timeout of three days), necessitating the use of both sound and heuristic pruning schemes; the latter however may compromise accuracy.

4. Conclusion

In this paper, we have taken a first step toward SAT-based verification of learned Logic Gate Networks (LGNs), focusing on the ease of verifying global fairness and robustness while retaining favorable predictive performance. Our initial experiments, including a 5-class dataset, show that LGNs can be verified efficiently using purely Boolean encodings, suggesting that such networks are well-suited for applications requiring strong correctness guarantees. Looking ahead, we plan to investigate more challenging datasets (e.g., complex image classification tasks) and tighten the integration between SAT solving and the verification tasks, for example, exploring the use of incremental SAT solving. In addition, we aim to explore real-world domains, such as neural network controllers for cyber-physical systems, where both efficient inference and reliable verification are critical. We hope this work will inspire further research into leveraging the inherently symbolic structure of LGNs for broader verification scenarios.

Acknowledgments

This work is supported in part by the ERC grant under Grant No. ERC-2020-AdG 101020093 and the Austrian Science Fund (FWF) [10.55776/COE12]. This research was supported by the Scientific Service Units (SSU) of ISTA through resources provided by Scientific Computing (SciComp).

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety, 2016. URL https://arxiv.org/abs/1606.06565.

Ross Anderson, Joey Huchette, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. In *Integer Programming and Combinatorial Optimization (IPCO)*, 2019.

Anagha Athavale, Ezio Bartocci, Maria Christakis, Matteo Maffei, Dejan Nickovic, and Georg Weissenbacher. Verifying global two-safety properties in neural networks with confidence. In *International Conference on Computer Aided Verification (CAV)*, 2024.

Gilles Barthe, Pedro R D'argenio, and Tamara Rezk. Secure information flow by self-composition. *Mathematical Structures in Computer Science (MSCS)*, 2011.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996.

Adrien Benamira, Thomas Peyrin, Trevor Yap, Tristan Guérand, and Bryan Hooi. Truth table net: Scalable, compact & verifiable neural networks with a dual convolutional small boolean circuit networks form. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. CaDiCaL, Gimsatul, IsaSAT and Kissat entering the SAT Competition 2024. In *Proceedings of SAT Competition 2024 – Solver, Benchmark and Proof Checker Descriptions*, 2024.

Sumon Biswas and Hridesh Rajan. Fairify: Fairness verification of neural networks. In *International Conference on Software Engineering (ICSE)*, 2023.

Michael R Clarkson and Fred B Schneider. Hyperproperties. *Journal of Computer Security (JCS)*, 2010.

Sumanth Dathathri, Krishnamurthy Dvijotham, Alexey Kurakin, Aditi Raghunathan, Jonathan Uesato, Rudy Bunel, Shreya Shankar, Jacob Steinhardt, Ian J. Goodfellow, Percy Liang, and Pushmeet Kohli. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring adult: New datasets for fair machine learning. *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, 2006.

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 2022.

Mahyar Fazlyab, Manfred Morari, and George J. Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, 2022.

Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. AI2: safety and robustness certification of neural networks with abstract interpretation. In *IEEE Symposium on Security and Privacy*, 2018.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision (LPCV)*. 2022.

Thomas A. Henzinger, Mathias Lechner, and Dorde Zikelic. Scalable verification of quantized neural networks. In *Conference on Artificial Intelligence (AAAI)*, 2021.

Hans Hofmann. Statlog (German Credit Data). UCI Machine Learning Repository, 1994.

Kai Jia and Martin C. Rinard. Efficient exact verification of binarized neural networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 2021.

Anan Kabaha and Dana Drachsler-Cohen. Verification of neural networks' global robustness. *Proceedings of the ACM on Programming Languages (PACMPL)*, 2024.

Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification (CAV)*, 2017.

Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification (CAV)*, 2019.

Haitham Khedr and Yasser Shoukry. Certifair: A framework for certified global fairness of neural networks. In *Conference on Artificial Intelligence (AAAI)*, 2023.

J. Larson. Propublica compas analysis. https://github.com/propublica/compas-analysis, 2017. Accessed: 2025-02-12.

Mathias Lechner, Dorde Zikelic, Krishnendu Chatterjee, Thomas A. Henzinger, and Daniela Rus. Quantization-aware interval bound propagation for training certifiably robust quantized neural networks. In *Conference on Artificial Intelligence (AAAI)*, 2023.

Changliu Liu, Tomer Arnon, Christopher Lazarus, Christopher A. Strong, Clark W. Barrett, and Mykel J. Kochenderfer. Algorithms for verifying deep neural networks. *Foundations and Trends in Optimization*, 2021.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Deep differentiable logic gate networks. *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Felix Petersen, Hilde Kuehne, Christian Borgelt, Julian Welzel, and Stefano Ermon. Convolutional differentiable logic gate networks. *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Luca Pulina and Armando Tacchella. Challenging SMT solvers to verify neural networks. *AI Communications*, 2012.

Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 2020.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Wenjie Ruan, Min Wu, Youcheng Sun, Xiaowei Huang, Daniel Kroening, and Marta Kwiatkowska. Global robustness evaluation of deep neural networks with provable guarantees for the hamming distance. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T. Vechev. Fast and effective robustness certification. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages (PACMPL)*, 2019.

Vincent Tjeng, Kai Yuanqing Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations (ICLR)*, 2019.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018a.

Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal security analysis of neural networks using symbolic intervals. In *USENIX Security Symposium*, 2018b.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Zhilu Wang, Chao Huang, and Qi Zhu. Efficient global robustness certification of neural networks via interleaving twin-network encoding. In *Design Automation and Test in Europe (DATE)*, 2022.

Linda F Wightman. LSAC national longitudinal bar passage study. LSAC research report series. 1998.

Haoze Wu, Omri Isac, Aleksandar Zeljic, Teruhiro Tagomori, Matthew L. Daggitt, Wen Kokke, Idan Refaeli, Guy Amir, Kyle Julian, Shahaf Bassan, Pei Huang, Ori Lahav, Min Wu, Min Zhang, Ekaterina Komendantskaya, Guy Katz, and Clark W. Barrett. Marabou 2.0: A versatile formal analyzer of neural networks. In *International Conference on Computer Aided Verification (CAV)*, 2024.