Truth Table Net: Scalable, Compact & Verifiable Neural Networks with a Dual Convolutional Small Boolean Circuit Networks Form

Adrien Benamira\textsuperscript{1}, Thomas Peyrin\textsuperscript{1}, Trevor Yap\textsuperscript{1}, Tristan Guérand \textsuperscript{1}, Bryan Hooi \textsuperscript{2}
\textsuperscript{1}Nanyang Technological University
\textsuperscript{2}National University of Singapore
{adrien.benamira, thomas.peyrin, trevor.yap}@ntu.edu.sg, guer0001@e.ntu.edu.sg, bhooi@comp.nus.edu.sg

Abstract

We introduce "Truth Table net" (TT\textsuperscript{net}), a novel Deep Neural Network (DNN) architecture designed to provide excellent scalability/compactness trade-offs among DNNs, allowing in turn to tackle the DNN challenge of fast formal verification. TT\textsuperscript{net} is constructed using Learning Truth Table (LTT) filters, analogous to how a Deep Convolutional Neural Network (DCNN) is built upon convolutional filters. The differentiable LTT filters are unique by their dual form: they are both a neural network-based function and a small-sized truth table that can be computed within a practical time frame. This characteristic guarantees, by design and independently of the overall architecture, the ability to practically extract an efficient (in terms of the number of logical gates) and functionally equivalent Conjunctive Normal Form (CNF) Boolean logic gate implementation. This CNF circuit is even optimal when the LTT truth table’s input bit size \(n\leq12\). In particular, TT\textsuperscript{net} architecture is the first differentiable DNN with as dual form a compact logic gate representation that can scale to datasets larger than CIFAR-10: we achieve an accuracy of 41% on the ImageNet dataset while ensuring that each LTT filter truth table is fully computable within \(2^{16}\) operations. We further compare the compactness and scalability performances of TT\textsuperscript{net} Boolean logic circuit representation to state-of-the-art differentiable logic DNNs across tabular, MNIST, and CIFAR-10 datasets. We emphasize that TT\textsuperscript{net} is the first solution to the open problem of designing differentiable convolutional neural networks with an exact dual logic gate circuit representation, bridging the gap between symbolic AI and trainable DCNNs. Finally, as improving DNNs compactness in Boolean logic circuit form reduces the complexity of their formal verification, we demonstrate TT\textsuperscript{net} effectiveness in exact sound and complete formal verification. Notably, our model achieves robustness verification in \(\approx 10ms\) vs \(\approx 100s\) for traditional state-of-the-art DNNs solvers.

1 Introduction

DNNs’ success in various machine learning tasks has been remarkable [Goodfellow et al., 2016], but their opacity has raised concerns about their security [AI, 2023; Commission, 2021], necessitating the development of formal verification processes to ensure safety and reliability [Driscoll, 2020]. Formal verification employs mathematical techniques to prove that a system satisfies predefined properties; however, the complexity of DNNs poses significant challenges for verification [Wang et al., 2021; Brix et al., 2023]. Traditional DNNs struggle to simultaneously achieve scalability, verifiability, and compactness in terms of logic gates [Petersen et al., 2022]. Ideally, a natural goal is to design a DNN that can be transformed into a compact Boolean logic circuit without sacrificing high accuracy on large-scale datasets. This approach holds the potential to enhance the compactness of DNNs and facilitate their verification processes [Jia and Rinard, 2020].

Our approach. DNNs, similar to ciphers, represent complex black-box functions that must be trustworthy. Drawing inspiration from a popular design principle of symmetric-key encryption algorithms (so-called Substitution-Permutation Networks or SPN [Daemen and Rijmen, 2002]), our paper proposes compact and small learnable filter Convolutional Neural Networks (CNNs) equivalent by design into small truth tables (from 6 to 16 bits of input). To achieve that goal, we adopt a divide-and-conquer strategy. In the divide phase, before training, we reduce the input size of each DNN building block function, enabling the computation of its complete distribution within a practical time, independent of its overall architecture. In the conquer phase, after training, we compute the complete distribution of each building function.

Therefore, our objective is to design DNNs based on functions that allow for full distribution computation within a practical timeframe, independent of the overall architecture, all while maintaining accurate performance on large datasets.

1.1 Our Contributions And Claims

The LTT filter. To address this challenge, our paper introduces a novel filter function called the Learning Truth Table (LTT) filter. The LTT filter serves as the fundamental building block function of TT\textsuperscript{net}, similar to how a CNN filter is utilized in a DCNN. The LTT filter is defined based on three key rules and presents six attractive properties, see Section 4.1.
The TTnet architecture. TTnet layers are assembled into a TTnet in the same way that CNN filters are assembled into DCNNs. Specifically, LTT filters are stacked together in LTT layers, and these LTT layers are stacked together in the overall TTnet. The final classification layer is linear.

Claim on scalability. In contrast to previous differentiable logic DNN solutions, TTnet demonstrates remarkable scalability, achieving performance comparable to other DNNs based on Boolean functions on the ImageNet classification task. For a fair comparison, we specifically evaluated TTnet against XnorNet [Rastegari et al., 2016] and the Binary Neural Networks (BNNs) original paper [Hubara et al., 2016], both of which are also based on Boolean functions. The results showcase that TTnet, which is a differentiable convolutional logic gate DCNN as well as a Boolean function, can effectively scale to large-scale datasets like ImageNet, achieving competitive performance (claim 1).

Claims on compactness. TTnet stands as the pioneering differentiable DCNN that can be transformed into a Boolean logic circuit by design (claim 2), which was already identified as an open problem [Petersen et al., 2022]. Empirical evaluation on small datasets like tabular datasets (Adult and Breast Cancer) and MNIST reveals that TTnet is equivalent to state-of-the-art differentiable logic gate DNN (Diff Logic Net [Petersen et al., 2022]) in terms of compactness and accuracy (claim 2A). Furthermore, at the cost of higher complexity, TTnet exhibits superior accuracy and training time scalability, achieving a notable accuracy of 70.75% on CIFAR-10 with less than an hour of training, marking an +8% improvement and ×90 faster training compared to Diff Logic Net [Petersen et al., 2022] (claim 2B).

Claims on formal verification. As a demonstration of its excellent scalability/compactness profile, we show that TTnet is very well suited to support fast property verification using generic SAT solvers (claim 3). This allows for the formal verification of the network’s robustness in dozens of milliseconds on MNIST and CIFAR-10, which is about 10^4 times faster than α – β-Crown [Xu et al., 2020; Wang et al., 2021], the state-of-the-art DNN verification solver and a recent winner of the Verification Neural Network (VNN) competition [Brix et al., 2023] (claim 3A). We also provide a comparison with BNNs verification [Jia and Rinard, 2020; Narodytska et al., 2019]: our method is more robust in low-noise cases and faster to verify in all scenarios, for MNIST/CIFAR-10 (claim 3B).

Outline. Section 2 presents a literature review on Boolean logic gates DNN and DNN formal verification. Section 3 provides a comprehensive overview of the background in the field of Boolean logic and formal verification. Section 4 describes the design principles and architecture of the TTnet model. In Section 5, we provide a thorough evaluation of the performance of our model on various datasets, carefully demonstrating the above claims. Finally, we discuss the limitations and opportunities in Section 6, and conclude in Section 7.

2 Related Works

Binary and Sparse Neural Networks. BNNs are DNN architectures where the activations and weights within a neural network use binary states, typically representing {−1, +1}. This innovative technique allows for the approximation of computationally expensive matrix multiplications through faster XOR and bitcount (popcount) operations. BNNs are characterized primarily by their weights. Sparse Neural Networks (SNNs) represent another approach to neural network design. In SNNs, only a subset of connections is present, in contrast to fully-connected layers. In the SNNs literature, the primary focus often revolves around distilling a sparse neural network from a dense one, with careful consideration given to the choice of connections. However, recent research has suggested the high efficacy of using randomized and fixed sparse connections from the outset. In our experiments, we include BNNs and SNNs as baseline models, due to their exceptional inference speed performance.

DCNNs into compact logic gate circuit representation. Chatterjee, 2018] proposed a DNN based on truth tables, but it did not scale beyond MNIST. Other works aimed to convert DNNs into compact Boolean logic circuit designs, which are essential for deploying DNNs on resource-constrained devices [Wang et al., 2019]. However, the challenge in learning logic gate networks is that they are typically non-differentiable, making them difficult to train with gradient descent [Rumelhart et al., 1986]. To date, there is no family of differentiable DCNNs logic gates, as stated by [Petersen et al., 2022] and our LTT filter function is proposed as a possible bridge to that gap between symbolic AI and trainable DCNNs.

Exact, complete, and sound formal verification. Formal verification is a critical aspect in ensuring the DNN’s soundness and completeness properties, particularly for safety-critical applications [Driscoll, 2020]. Conventional verifiers work with real-valued networks, but they face scalability challenges, e.g. hundreds of seconds to verify an MNIST image property [Müller et al., 2022], and provide no guarantees of correctness due to floating point errors [Jia and Rinard, 2021]. Therefore, developing an efficient DNNs verification process is an important topic, as evidenced by the α – β-Crown paper [Xu et al., 2020; Wang et al., 2021] and winner of the VNN competition [Brix et al., 2023]. In the present work, instead of designing a new general verification method for DNNs, we propose the first architecture that can be efficiently and correctly verified with any SAT solver [Roussel and Manquinho, 2009]. This research direction is attracting interest, as exemplified by [Jia and Rinard, 2020], who strives to show that BNNs [Hubara et al., 2016] are faster to verify if a specific well-crafted SAT solver is designed. Further discussion regarding incomplete methods is given in Appendix.

Local features for accurate DNN. Recent works by [Brendel and Bethge, 2018] and [Agarwal et al., 2021] experimentally showed that highly nonlinear functions (Resnet for the former, expanding autoencoder for the latter) can enable a DNN to learn high-quality local features (7 × 7 floating input pixels on image dataset for the former, 1 × 1 floating input on tabular dataset for the latter). In this work, we build upon the idea of using highly nonlinear functions to learn local features.
However, instead of processing local-dimensional floating-point inputs, TTNet reduces the local features to binary inputs ($n \leq 16$). These two approaches legitimize our use of TTNet on tabular and image datasets and highlight the importance of Rule 3 in Section 4.1.

3 Background

Boolean logic. A truth table is a mathematical representation of the output of a Boolean function for all possible input combinations. It is usable in practice when the corresponding Boolean function distribution is fully computable. Note that the Boolean function distribution of a DNN/BNN is not fully computable, as the input size of the function is too large. A truth table can be expressed in CNF1, being the standard form to represent a Boolean function, especially in formal verification. A Boolean logic circuit is a physical or mathematical representation of a Boolean function, made up of interconnected logic gates [Arora and Barak, 2009; Klir et al., 1997], typically AND/OR gates.

Formal verification. Our framework is built upon [Narodytska et al., 2019], a theoretical exact sound and complete framework to verify any property. Given a precondition $prec$ on inputs $x$, the property $prop$, on outputs $o$, and the SAT relations provided by a DNN between inputs and outputs denoted as $DN(x, o)$, we assess the validity of the statement $prec(x) \land DN(x, o) \Rightarrow prop(o)$. To show the existence of a counter-example to this robustness property, we search for a satisfying assignment of $prec(x) \land DN(x, o) \land prop(o)$ (Eq1) with a proven SAT solver. An “unsat” result from the SAT solver proves that there is one noise that satisfies (Eq1) to attack the DNN in Boolean logic gate format; and the floating-point operations counterpart denoted as FLOPs.

We define the computational efficiency in binary operations, denoted as OPs, as the total number of binary gates (NAND/NOR/AND/OR/XOR/XNOR) required to represent the entire DNN in Boolean logic gate format; and the floating-point operations counterpart denoted as FLOPs.

We now define LTT filters as below.

\[ o_{ij} = o_{ij}^1 \cup \cdots \cup o_{ij}^G \] (1)

with $\cup$ the concatenation operation and unit $o_{ij}^\gamma \in \mathbb{R}^{C_{out}/G}$ representing the output in group $\gamma$ at position $i, j$. Namely:

\[ o_{ij}^\gamma = \Psi_{w^{\gamma}}(x_{ij}^\gamma, \ldots, x_{ij+k-1,j+k-1}^\gamma) = \sum_{u,v=0}^{k-1} x_{[i+u,j+v]}^\gamma w_{u,v}^{\gamma} \] (2)

where $i \in \{0, \ldots, H\}$, $j \in \{0, \ldots, W\}$, $\gamma \in \{0, \ldots, G\}$, with the input $x_{[i+u,j+v]}^\gamma \in \mathbb{R}^{C_{in}/G}$ and the convolution filter weights $w_{u,v}^{\gamma} \in \mathbb{R}^{(C_{in}/G) \times (C_{out}/G)}$.

Property 2. Given that the set of inputs $(x_{i,j}^1, \ldots, x_{i+k-1,j+k-1}^1)$ is encoded on $q$ bits, the complete distribution of one 2D-CNN filter of function $\Psi_{w^{\gamma}}$ can be computed in $2^q \times k \times k \times (C_{in}/G) \times (C_{out}/G)$ operations. Furthermore, the weights $\omega^{\gamma}$ are real and learnable using gradient descent.

4 Truth Table Neural Networks

First, we define LTT filters, give their properties, and illustrate their use with an example in Section 4.1. We then explain how these LTT filters are integrated into TTNet in Section 4.2. We also provide a companion video in Supplementary Material.

4.1 Overall LTT Filter Design

General Design Criteria and Rules Of LTT Filters.

We aim to develop a novel filter function, called the Learning Truth Table (LTT) filter, that is comparable to traditional CNN filters in terms of computational efficiency but differs in complexity. Specifically, we design an LTT filter such that one of its main characteristics is to be a grouped CNN filter: it will be sparser in connectivity than a classical CNN filter. The LTT filter essential criteria are:

(A) The LTT filter distribution must be entirely computable in practical time, independently of the overall DNN architecture.

(B) Once LTT filters are assembled into a layer and layers into a DNN, the latter should be scalable, especially on large datasets such as ImageNet.

To achieve that, we define three LTT filter design rules:

Rule 1: Force the input bit size $n$ of the LTT filter to be $n \leq 16$, independently of the architecture.

Rule 2: Use binary inputs/outputs, but with real-valued weights and intermediate values.

Rule 3: Ensure that the LTT filter uses nonlinear functions in between the Heaviside activations.

As a result, each filter in our architecture becomes a truth table with a maximum input bit size of 16, leading to a compact DNN represented as a Boolean logic gate circuit that is easy to train and verify. We explore these benefits in Section 5.

Regular 2D-Convolution Filters.

We first define below a traditional 2D-Convolution filter.

Definition 1. Let an input $X \in \mathbb{R}^{C_{in} \times H \times W}$ where $C_{in}$, $H$, $W$ represent the number of channels, height, and width of a channel respectively. If a regular convolution filter is applied on $X$ with kernel size $k \times k$, group $G$, stride $s = 1$ padding $p = 0$, the output is denoted as $O \in \mathbb{R}^{C_{out} \times H \times W}$ where every output unit $o_{ij} \in \mathbb{R}^{C_{out}}$ is:

\[ o_{ij} = o_{ij}^1 \cup \cdots \cup o_{ij}^G \] (1)

with $\cup$ the concatenation operation and unit $o_{ij}^\gamma \in \mathbb{R}^{C_{out}/G}$ representing the output in group $\gamma$ at position $i, j$. Namely:

\[ o_{ij}^\gamma = \Psi_{w^{\gamma}}(x_{ij}^\gamma, \ldots, x_{ij+k-1,j+k-1}^\gamma) = \sum_{u,v=0}^{k-1} x_{[i+u,j+v]}^\gamma w_{u,v}^{\gamma} \] (2)

where $i \in \{0, \ldots, H\}$, $j \in \{0, \ldots, W\}$, $\gamma \in \{0, \ldots, G\}$. We define two convolutions filters $\Psi_{w^{\gamma}}$ and $\Psi_{w^{\gamma}}$ in $\mathbb{R}^{(C_{in}/G) \times (C_{out}/G)}$ operations. Furthermore, the weights $\omega^{\gamma}$ are real and learnable using gradient descent.
and such that \( k_1 + k_2 - 1 = k \) and \( a \) is the amplification ratio. We also define \( \Theta \) to be a nonlinear function. The LTT output is denoted as \( O \in \mathbb{R}^{C_{in} \times H \times W} \) where every output unit \( o_{i,j} \in \mathbb{R}^{C_{out}} \) is defined as in Equation (1). Unit \( o_{i,j}^{\gamma} = \Pi_{i,j}^{\gamma} \cdot \omega_2^{\gamma} \cdot x \), where the output unit in group \( \gamma \) at position \( i,j \) is given by
\[
o_{i,j}^{\gamma} = \Phi(o_{i,j}^{\gamma})(x_{i,k_1-1,j+k_1-1}) = binact(\Psi(o_{i,j}^{\gamma})(x_{i,k_1-1,j+k_1-1}))
\]
with \( \Psi(o_{i,j}^{\gamma}) \) defined as in Equation (2) and \( o_{i,j}^{\gamma} \in \mathbb{R}^{C_{in} \times C_{out}} \) as
\[
o_{i,j}^{\gamma} = \Theta(\Psi(o_{i,j}^{\gamma})(x_{i,k_1-1,j+k_1-1}))
\]
with \( i \in \{1, \ldots, H\}, j \in \{1, \ldots, W\}, \gamma \in \{1, \ldots, G\} \) and with the input \( x_{i,j} \in \mathbb{R}^{C_{in} \times C_{out}} \).

By definition, the described LTT filters validate the three rules given previously. In Figure 1a, we provide an example of an LTT computation in one dimension for \( k_1 = 3, k_2 = 2, k = k_1 + k_2 - 1 = 4, a = 4, C_{in} = 1, C_{out} = 1 \) and \( \Theta = \text{ReLU} \). The architecture is depicted in Figure 1b.

LTT Filters Properties.
We point six main LTT properties. Properties P1 and P2 of the proposed TTTnet architecture offers practical benefits, including fast and scalable filter training. Meanwhile, properties P3, P4, P6 lead to the development of compact LTT filters and efficient verification techniques. Figure 2 provides an example of the two forms of P5.

Example: From LTT Weights To Truth Table To CNF.
Consider a trained 1D-LTT \( \Phi_{\omega_1, \omega_2} \) with input size \( n = 4 \), a stride of size 1, and no padding. The architecture of \( \Phi_{\omega_1, \omega_2} \), given in Figure 1b, is composed of two CNN filter layers: the first one has parameters \( \omega_1 \) with (input channel, output channel, kernel size, stride) = (1, 4, 3, 1), while the second \( \omega_2 = (4, 1, 2, 1) \). The values of the weights \( (\omega_1, \omega_2) \) are given in Figure 2. The inputs and outputs of \( \Phi_{\omega_1, \omega_2} \) are binary, and we denote the inputs as \( [x_0, x_1, x_2, x_3] \). To compute the entire distribution of \( \Phi_{\omega_1, \omega_2} \), we generate all \( 2^4 = 16 \) possible input/output pairs, as shown in Figure 1a, and obtain the truth table in Figure 2. This truth table fully characterizes the behavior of \( \Phi_{\omega_1, \omega_2} \). We then transform the truth table into an optimal CNF using the Quine-McCluskey algorithm [Blake, 1938; Udovenko, 2023]. This optimal CNF fully characterizes the behaviour of \( \Phi_{\omega_1, \omega_2} \) as well and is exactly equivalent.

P1: The LTT filter weights \( (\omega_1, \omega_2) \) are trainable with gradient descent and the Straight-Through Estimator (STE) to handle the input/output binarisation.

P2: The LTT filter preserves real-valued weights \( (\omega_1, \omega_2) \) and intermediate values \( a \).

P3: The entire distribution of the LTT filter can be calculated in \( 2^n \leq 2^{10} = 65,536 \) operations (less than 1 ms on a standard PC). Truth table input bit size \( n \) is independent of the architecture.

P4: The Quine-McCluskey algorithm [Blake, 1938] can be used to compute the optimal CNF (in terms of Boolean logic gates) from the LTT truth table if \( n \leq 12 \). For \( 12 < n \leq 16 \) a compact CNF can be computed [Udovenko, 2023].

P5: The LTT filter has two forms: neural network weights \( (\omega_1, \omega_2) \), or a small Boolean circuit (as a truth table or as a Boolean logic gates).

P6: Under truth table or CNF form, an LTT evaluation does not need to compute any activation function.

4.2 Overall TTTnet Design
We integrated LTT filters into the neural network, just as CNN filters are integrated into a deep convolutional neural network: each LTT layer is composed of multiple LTT filters and there are multiple LTT layers in total (see Figure 2). Additionally,
We used 4 Nvidia GeForce RTX 3090 GPUs and 8 cores (experiment has been repeated three times with different seeds).

Table 1: Comparison of top 1 and top 5 natural accuracy on ImageNet with architecture/training information are given in Appendix.

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>TTnet$_{16—18}$</th>
<th>Original BNN</th>
<th>XnorNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>top 1</td>
<td>41.6% ± 0.6</td>
<td>27.9%</td>
<td>44.2%</td>
</tr>
<tr>
<td>top 5</td>
<td>65.1% ± 0.7</td>
<td>50.4%</td>
<td>69.2%</td>
</tr>
</tbody>
</table>

Table 2: Ablation study of TTnet$_{n—k}$ for different n on CIFAR-10. Acc. stands for accuracy.

5 Results

Experimental environment. The project code can be found in Supplementary Material and was coded in Python with PyTorch library [Paszke et al., 2019] for training, Numpy [Van Der Walt et al., 2011] for testing as we infer with truth tables. We used 4 Nvidia GeForce RTX 3090 GPUs and 8 cores Intel(R) Core(TM) i7-8650U CPU clocked at 1.90 GHz, 16 GB RAM. We note TTnet$_{n—k}$ & TTnet with truth tables of input bit size n, without pre-processing layer and a final linear layer with weights quantized on k bits; and TTnet$_{n—k}$, with a pre-processing layer of one CNN layer with floating weights.

5.1 Claim 1: Scalability

TTnet$_{16—18}$ shows an accuracy of 41.6% on ImageNet with truth tables of size n = 16. These results are comparable to those achieved by the original BNN paper [Hubara et al., 2016] and XnorNet [Rastegari et al., 2016], see Table 1. Our experimental results confirm that TTnet achieves high accuracy on a large dataset, which proves its non-trivial nature. We emphasize that other differentiable logic gate networks did not scale to ImageNet so far: the work done by [Petersen et al., 2022] does not scale to larger datasets than CIFAR-10. However, we believe there remains room for improvement on TTnet in terms of accuracy as BNNs now can achieve ≈ 70% – 75% [Liu et al., 2018] on ImageNet. Detailed architecture/training information are given in Appendix. Table 2 studies the influence of the input bit size n of the truth table on the scalability for CIFAR-10.

5.2 Claims 2: Compactness

We provide experimental evidence to support our claim of providing easy-to-train, accurate, and compact logic gate DCNNs. The results are presented in Table 3 for tabular datasets (Adult and Breast Cancer) and Table 4 for image datasets (MNIST and CIFAR-10) along with comparisons with state-of-the-art differentiable logic gate DNN [Petersen et al., 2022], BNNs [Umuroglu et al., 2017; Hirtzlin et al., 2019] and SNNs [Molchanov et al., 2017; Mocanu et al., 2018; Han et al., 2016; Zeng and Urtasun, 2018; Zhou et al., 2021]. More information on training conditions, architectures, and additional comparisons can be found in Appendix.

A FLOP (Floating point OPerations) is typically composed of multiple binary operations (OPs). Float32 adders/multipliers require around 1,000 logic gates or lookup tables and have a significant delay. They are commonly implemented in CPU and GPU hardware due to their importance. However, they are much more expensive than simple calculations or bitwise logical operations on int64 data types. Processors can perform 3 to 10 int64 bitwise operations per cycle, while floating-point operations usually take a full clock cycle. Converting a non-sparse model assumes a conservative estimate of 1000 OPs per 1 FLOP. Neural networks’ speeds are theoretical, with sparse execution often being 10 to 100 times slower. In practice, 1,000 binary OPs for 1 clear FLOP (float32) is a conservative estimate for SNNs. It’s also a cautious estimate for sparse float32 models. Theoretical estimates assume no density cost and no hardware acceleration for floating-point operations [Petersen et al., 2022].

Performances discussion - small datasets. We present a comparison between two methods, TTnet$_{6—4}$ and Diff Logic Net, with respect to accuracy and compactness on small
datasets. Compactness is defined in terms of the number of Boolean logic gates (NAND/NOR/AND/OR/XOR/XNOR) used by the model, as noted as OPs. Our results in Table 3 demonstrate that TTnet outperforms Diff Logic Net in terms of accuracy for both tabular datasets, while using much fewer gates. On the MNIST dataset, as indicated in Table 4 (top table), both models achieve comparable accuracies (10% maximum relative error rate), but TTnet utilizes fewer operations. In summary, our findings show that TTnet and Diff Logic Net have similar performance on small datasets, with TTnet offering the advantage of compactness in terms of number of logical gates.

<table>
<thead>
<tr>
<th>MNIST</th>
<th>Acc.</th>
<th># Param.</th>
<th>OPs</th>
<th>FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional models</td>
<td>Linear Regression</td>
<td>91.60%</td>
<td>4K (4M)</td>
<td>4K</td>
</tr>
<tr>
<td></td>
<td>Neural Network</td>
<td>98.40%</td>
<td>22.6M (45G)</td>
<td>45M</td>
</tr>
<tr>
<td>Boolean DNNs</td>
<td>Diff Logic Net (small)</td>
<td>98.07%</td>
<td>48K</td>
<td>48K</td>
</tr>
<tr>
<td></td>
<td>Diff Logic Net</td>
<td>98.47%</td>
<td>341K</td>
<td>341K</td>
</tr>
<tr>
<td></td>
<td>TTnet₆-⁴ (small)</td>
<td>97.34%</td>
<td>37K</td>
<td>34K</td>
</tr>
<tr>
<td></td>
<td>TTnet₆-⁴ (big)</td>
<td>98.32%</td>
<td>203K</td>
<td>188K</td>
</tr>
<tr>
<td>BNNs</td>
<td>FINN</td>
<td>98.40%</td>
<td>-</td>
<td>5.28M</td>
</tr>
<tr>
<td>SNNs</td>
<td>M₁</td>
<td>98.08%</td>
<td>4K</td>
<td>8K</td>
</tr>
<tr>
<td></td>
<td>SET-MLP</td>
<td>98.74%</td>
<td>89.8K (180M)</td>
<td>180K</td>
</tr>
</tbody>
</table>

Table 3: Comparison of TTnet with state-of-the-art Diff Logic Net [Petersen et al., 2022] on Adult and Breast Cancer tabular datasets for two models. Acc. stands for accuracy, OPs for the number of logical gates.

5.3 Claim 3: Complete, Sound Formal Verification

Greater Boolean logic compactness in circuit design can aid formal verification by reducing the complexity of the verification process. Therefore, we applied TTnet to formal verification, where two strategies are common: either using a specific solver to verify ReLU based-DNN like β-Crown [Wang et al., 2021], or using a specific architecture that allows a generic verification method. We compared both methods based on natural accuracy, verified accuracy for \( \ell_{∞} \)-norm bounded input perturbations (formal definition in Appendix). For a fair comparison, we note that our pre-processing layer and training configuration are the same as [Jia and Rinard, 2020]. More results and discussions can be found in Appendix.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>General DNN + α-β-Crown</th>
<th>TTnetα-1 + General SAT verification pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>96</td>
<td>13</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>175</td>
<td>27</td>
</tr>
</tbody>
</table>

Table 5: Comparison of verification strategies: usage of a general DNN to verify with \( \alpha - \beta \)-Crown [Xu et al., 2020; Wang et al., 2021] or using specific TTnet with a general SAT verification method. The comparison is based on the 7 benchmarks from the VNN competition, and the results are presented as an average, full results are given in Appendix. TTnet has no pre-processing, \( n = 9, k = 1 \).

Strategy 1: Comparison with general DNN solvers. In Table 5, we present a comparison of our proposed verification strategy, which utilizes the TTnet architecture and classical verification tools, against the state-of-the-art \( \alpha - \beta \)-Crown...
method, the winner of the VNN competition 2021. The comparison is based on the 7 benchmarks from the VNN competition, and the results are presented as an average. Our approach demonstrates a significant improvement in verification time, with an average speed-up of 1250x for CIFAR-10 and 1600x for MNIST, at the same noise level. Additionally, a higher verified accuracy (+4% and +7%) was observed on the cifar_10_resnet benchmark in Appendix. Also, our approach did not encounter any timeouts, whereas $\alpha - \beta$ Crown had an average of more than 10% of timeouts.

It should be noted that our strategy cannot be directly compared to the VNN competition as the competition focuses on novel DNN verification algorithms/pipelines, whereas we propose a new DNN family (TTnet) that can be easily verified using classical verification tools. However, the results presented demonstrate the competitiveness of our approach. One can use our strategy to verify the robustness property on CIFAR-10, 1K images, standard DNN with $\alpha - \beta$-Crown which takes 2 days, whereas it takes 14 seconds if one chooses to verify TTnet.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Complete method</th>
<th>Accuracy</th>
<th>Verif.</th>
<th>Nat.</th>
<th>Verif. time (s)</th>
<th>Timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST ($\epsilon_{\text{test}} = 0.1)$</td>
<td>TTnet$_{g=1}$</td>
<td>95.12%</td>
<td>98.33%</td>
<td>0.012</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>JR20</td>
<td>95.68%</td>
<td>97.46%</td>
<td>0.1115</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N+19</td>
<td>20.00%</td>
<td>96.00%</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MNIST ($\epsilon_{\text{test}} = 0.3)$</td>
<td>TTnet$_{g=1}$</td>
<td>66.24%</td>
<td>97.43%</td>
<td>0.005</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>JR20</td>
<td>77.59%</td>
<td>96.36%</td>
<td>0.1179</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIFAR-10 ($\epsilon_{\text{test}} = 2/255$)</td>
<td>TTnet$_{g=1}$</td>
<td>32.32%</td>
<td>49.23%</td>
<td>0.06</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>JR20</td>
<td>30.49%</td>
<td>47.35%</td>
<td>0.1750</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIFAR-10 ($\epsilon_{\text{test}} = 8/255$)</td>
<td>TTnet$_{g=1}$</td>
<td>21.08%</td>
<td>31.13%</td>
<td>0.04</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>JR20</td>
<td>22.55%</td>
<td>35.00%</td>
<td>0.1781</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* results given on the first 1K images of the test set. Moreover, the authors only authorize a maximum of 20 pixels to switch.

Table 6: Application of TTnet to complete adversarial robustness verification for low and high noise bounded by $l_{\infty}$. We tabulate results of verified accuracy, natural accuracy, and mean verification time on MNIST and CIFAR-10 datasets in comparison to state-of-the-art SAT methods (JR20 stands for [Jia and Rinard, 2020] and N+19 for [Narodytska et al., 2019]). The best verified accuracy and verification time are displayed in bold.

6 Limitations And Future Works

Overall limitations. The main limitation of TTnet is closely related to the maximum input bit size $n$, so that the truth table can be fully enumerated practically. Additionally, users cannot pre-set the final number of gates or complexity before starting the training process. Also, as mentioned in Section 2, it has not yet been shown that the aggregation of local features learned by non-linear functions can provide good performances for another dataset than images and tabular.

Limitations on scaling. While TTnet has shown promising results, its performance falls behind state-of-the-art computer vision models on larger datasets. Improving the natural accuracy of the model without increasing $n$ is an area for future investigation [Liu et al., 2018; Bello et al., 2021].

Limitations on compactness. TTnet achieves better compactness than previous methods on tabular and MNIST datasets, but its compactness on CIFAR-10 and accuracy on MNIST could be improved. Further research is needed to address this limitation and explore the use of CNNs with Boolean logic circuits for larger datasets such as ImageNet.

Limitations on formal verification. While TTnet has demonstrated promising results in terms of verification, it may be less effective in the presence of high noise. Further research is needed to increase the model’s verified accuracy, especially in noisy environments. This includes exploring the use of dedicated robustness training techniques.

Future works. The TTnet architecture presents several opportunities for future research. These include exploring its application on time series or graph-based datasets, developing better LTT architectures and training methods to achieve higher accuracy on ImageNet, and creating heuristics for converting truth tables to optimized CNF for larger $n$, which would in turn help the scaling. TTnet could also be a good candidate for the VNN competition.

7 Conclusion

This work is a step towards more scalable, compact, and verifiable DNNs while linking symbolic AI and learning AI. There is room for improving TTnet, and we hope that it will inspire further explorations into the use of truth tables as a tool for applying DNNs to critical applications. We provide more discussions on TTnet in Appendix.

References


