Hardware Trojan Detection via BiGCN-TIF and Multi-Level Feature Fusion

*2025 CAD Contest Problem A

Tzu-Chi Huang
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
zchuang0203@gmail.com

Chi-Lun Chen
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
chilunchen28@gmail.com

I. Proposed Methodology

Our approach, BiGCN-TIF (Bidirectional Graph Convolutional Network with Topology and Information Fusion), aims to detect and localize hardware Trojans in gate-level netlists by integrating structural graph analysis and GNNs. The system includes graph construction, model training, and inference post-processing.

A. Graph Construction and Features

The Verilog netlist is converted into a directed graph where each gate is a node, and signal propagation forms directed edges.

1) Bidirectional Graph: To enable bidirectional message passing, two edge index tensors are generated: Forward Edges (`edge_index_fw`, driver to load) and Backward Edges (`edge_index_bw`, load to driver).

2) Multi-Level Feature Extraction: We extract multi-level features for each node, resulting in a **41-dimensional feature vector** $\mathbf{x}$:

- **Gate-Level:** Gate type one-hot encoding, IO flags.
- **Structural/Topological:** Fanin/fanout counts within two hops, local depth, DFF presence, and distance features (to PI, PO, DFF).
- **Simulation Statistics:** Logic-1 probability and toggle frequency from Monte-Carlo simulation.

We also introduce **subgraph-level contextual features** $\mathbf{Z}_s$ by grouping gates into Weakly Connected Components (WCCs) or DFF-based segments.

B. Dual-Branch GNN Architecture

The architecture combines node-level and subgraph-level representations.

1) Node Branch (BiGCN-TIF): This branch consists of three stacked BiGCN_TIF_Layer modules. Each layer uses a pair of GCNConv operators on the forward and backward edge indices. The layer outputs are concatenated to form a comprehensive 192-dimensional node structural representation $\mathbf{h}_{\text{node}}$.

2) Subgraph Branch: This branch uses a 2-layer Feed-Forward Network (MLP) to encode the statistical subgraph features $\mathbf{Z}_s$ into a 64-dimensional latent space. These embeddings are then aligned to all nodes to form the contextual representation $\mathbf{z}_{\text{node}}$.

3) Fusion and Classification: The 192-dim $\mathbf{h}_{\text{node}}$ and 64-dim $\mathbf{z}_{\text{node}}$ are **concatenated into a 256-dimensional vector**. This fused vector is classified by a two-layer MLP head to produce outputs for the $\{\text{Free, Trojan}\}$ classes.

C. Inference and Post-Processing

The model's outputs undergo topology-aware filters to enhance spatial consistency and robustness.

- **Neighbor Consistency Filter:** Resets a predicted Trojan node to non-Trojan if it has no other Trojan neighbors within one hop.
- **Small-Group Pruning:** Connected groups of predicted Trojan gates with size < 5 are discarded.
- **Trojan Count Enforcement:** If the total predicted Trojan count is below **10**, all predictions are reset to zero, classifying the design as Trojan-Free.

II. Experiment Configuration

A. Dataset Generation

The final dataset consists of approximately **2,180 netlists** ($\approx 2,080$ Trojan-inserted and 100 Trojan-free). Two strategies were used for Trojaned data generation:

1) **RTL-Level Injection:** Adjusting parameters on official RTL Trojan templates and synthesizing them using commercial EDA tools, Cadence GENUS and Synopsys Design Compiler (DC).
2) **Logic-Level Augmentation:** Performing structural transformations on public benchmark cases to enhance diversity.

B. Training Procedure

Given the severe class imbalance, **Focal Loss** is employed:

$$\text{FL}(p_t) = -\alpha(1-p_t)^{\gamma}\log(p_t)$$

where $\alpha \in [0.5, 0.8]$ and $\gamma = 2$.

- **Optimizer:** Adam.
- **Learning Rate:** 8×10^{-5}.
- **Early Stopping:** Patience of 100 epochs.
- **Batch Size:** Single-graph batch size (batch size $= 1$).

C. Evaluation Metrics

The final evaluation employs the comprehensive contest-style metric:

$$\text{Total Score} = \text{Classification Score} + \text{F1 Bonus}$$

The F1 Score, Precision, and Recall are used for localization, while Accuracy is used for overall classification.

III. Result Summary

The total score achieved was $\mathbf{121.8891}$ across the cases, demonstrating robust performance on both classification and localization tasks.

References